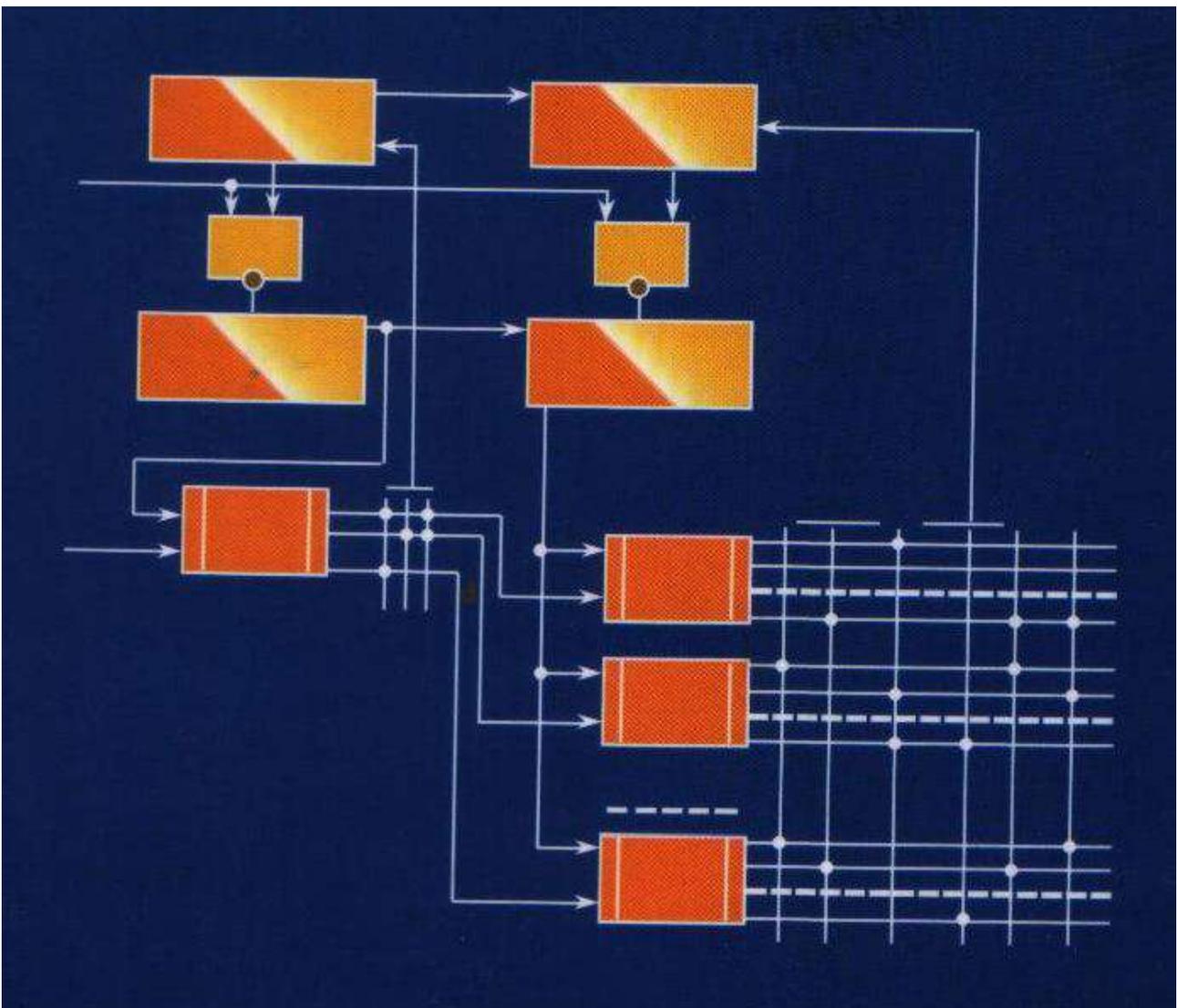


**L.F.MARAKHOVSKY**

**HANDBOOK ON DIGITAL COMPUTER ENGINEERING**

**(INFORMATION TECHNOLOGY FOR PROCESSING THE  
HIERARCHIC INFORMATION IN DIGITAL COMPUTER  
TECHNOLOGY)**



**Kyiv -2018**

УДК 004.258:252 + 519.716:713.1

**Marakhovsky L.F.**

Handbook of digital computing: (information technology processing hierarchical information in digital computers) - K.: Interdisciplinary Academy of Sciences of Ukraine, 2018. - 375 p.

The handbook sets out the basics of new knowledge in information technology. The technology is based on the use of automatic memory circuits and is oriented to the processing of hierarchical information. In the handbook are considered: the principle of hierarchical program control, the theory of multifunctional automata of Marakhovsky (1st, 2nd and 3rd kind) and multi-level automata.

Methods of synthesis of multifunctional and multilevel new elementary memory circuits. Automatic continuous time, within which it became possible to consider the functioning of multifunctional automata, new memory circuits and the construction on their basis of typical reconfigurable devices, such as: registers, counters, control devices, microprocessors, computers and other devices and computer systems are considered. The methods of hierarchical software and the foundations for constructing neurons and neural networks based on automatic memory circuits are described.

The guide material will allow developers of computer devices to create competitive reconfigurable processors, computers and computer networks.

The handbook can be useful to students of technical universities, postgraduate students, teachers and scientists specializing in the field of computer technology.

Reviewers:

**Stakhov A.P.** - Doctor of Technical Sciences, Professor, Academician of the Academy of Engineering Sciences of Ukraine, Honorary Professor of Taganrog Radio Engineering University, President of the International Club of the Golden Section.

**Borisenko A.P.** - Doctor of Technical Sciences, Professor, Head of the Department of Electronics and Computer Technology, Sumy State University of MES of Youth and Sports of Ukraine.

**Verbitsky V.G.** - Doctor of Technical Sciences, Professor, Director of IMP NASU

© Marakhovsky L.F.

© Interdisciplinary Academy of Sciences of Ukraine, 2018

## CONTENTS

List of Abbreviations	8
Foreword	9
Section 1. THEORY OF ABSTRACT HIERARCHICAL AUTOMATES	11
Chapter 1. Multifunctional and multilevel abstract machines	11
1.1. Statement of the problem	11
1.2. Automatic continuous time	13
1.3. Multifunctional deterministic automata	19
1.4. Reconfigurable multifunction machines	30
1.5. Structures of multi-level automatic memory devices	31
1.6. Abstract probabilistic automata of the third kind	36
1.7. Abstract fuzzy automata of the third kind	43
1.8. Classification of deterministic abstract automata	47
1.9. Classification of nondeterministic abstract automata	48
1.10. Conclusion to the 1st chapter	5p
Chapter 2. Multilevel Abstract Automata	51
2.1. Principle of hierarchical program management	51
2.2. The meaning of probabilistic and fuzzy transitions in real devices of computer equipment	53
2.3. Multilevel Abstract Automatic Machines	55
2.4. Conclusion to the 2nd chapter	59
Chapter 3. Methods for setting automata	61
3.1. Tabular methods for specifying multifunctional abstract determinate automata	61
3.2. Tabular methods for specifying abstract probability automata of the third kind	65
3.3. The way to specify a hierarchical abstract automaton with a multifunctional system of organizing memory using a polygraph	69
3.4. Formulation of a polygram	76
3.5. The automaton of the 4th kind that controls the operability of basic memory circuits	79
3.6. Conclusion to the third chapter	82
Conclusion to section 1	82
Section 2. STRUCTURAL ORGANIZATION OF MULTIFUNCTIONAL AND MULTI-LEVEL MEMORY SCHEMES	84
Chapter 4. Multifunctional memory circuits	84
4.1. Basic concepts	84
4.2. Method of microstructural synthesis of elementary multifunctional memory circuits	86
4.3. Symbolic language for describing elementary multifunctional memory circuits	90

4.4. Possible variants of multifunctional memory circuits in symbolic number	93
4.5. Synthesis of multifunctional memory circuits by symbolic description	95
4.6. The definition of the input words of the automatic memory circuits	101
4.6.1. Definition of single-valued elementary input words	101
4.6.2. Determination of enlarged elementary input words	108
4.7. Reliability issues of multifunctional memory circuits	112
4.7.1. Issues of improving the reliability of memory circuits	113
4.7.2. Issues of increasing the survivability of memory circuits	118
4.7.3. Issues of monitoring the efficiency of memory circuits	119
4.8. Characteristics of the parameters of multifunction memory circuits	121
4.9. Conclusion to chapter 4	131
Chapter 5. Multi-level memory circuits	132
5.1. Basic concepts	132
5.2. Symbolic language for describing multilevel memory schemes	133
5.3. Defining the parameters of multi-level memory circuits by symbolic description	135
5.4. Method of synthesis of multilevel memory circuits by symbolic description	137
5.5. Principle of structural organization of elementary multi-level memory circuits	141
5.6. The method of designing a general strategy automaton	142
For the entire multi-level memory circuit	
5.7. The method of logical design of a multi-level memory scheme of a class with a single automaton of strategy	146
5.8. The method of logical design of a multilevel class memory scheme with the strategy automaton for each group of multilevel memory schemes	150
5.9. Classification of basic elementary memory circuits	156
5.10. Comparison of the parameters of the basic memory circuits	157
5.10.1. Determination of the number of logical elements necessary for constructing memory circuits	157
5.10.2. Determination of the maximum possible load The ability to output memory circuits	158
5.10.3. Determining the number of internal links of memory circuits	159
5.10.4. Determining the number of external links of memory circuits	159
5.10.5. Determination of the number of elements per one state of the memory circuits	160
5.10.6. Comparison of the operating switching frequency and the functionality of the memory circuits	160
5.11. Issues of constructing reliable devices on multi-level memory circuits	161
5.11.1. Reliability issues for multi-level memory circuits	161
5.11.2. The questions of the vitality of multilevel memory circuits	166
5.11.3. Monitoring the performance of the MUSP	169
5.11.4. Increasing the reliability of devices using MFIS in the MUSP	172
5.12. Conclusion 5 chapters	175

Conclusion to section 2	176
Section 3. TYPICAL COMPUTER DEVICES ON AUTOMATIC MEMORY SCHEMES	178
Chapter 6. Methods for constructing reconfigurable registers	178
6.1. Basic concepts	178
6.2. Methods for constructing reconfigured registers on multilevel memory circuits	180
6.3. Analysis of parameters of reconfigured parallel registers in multi-level memory circuits	190
6.4. Methods for constructing the reconfigured shift registers in multi-level memory circuits	194
6.5. Conclusion to chapter 6	196
Chapter 7. Methods for constructing reconfigurable counters on multi-level memory circuits	197
7.1. Basic concepts	197
7.2. Methods for constructing reconfigured counters on multi-level memory circuits	199
7.3. Conclusion to chapter 7	208
Chapter 8. Control device on multi-level memory circuits	209
8.1. Basic concepts	209
8.2. Methods for constructing a reconfigured control device on multi-level memory circuits	210
8.3. Conclusion to the 8th chapter	213
Chapter 9. Summers on the Matrix Schemes of MUSP	214
9.1. Basic concepts	214
9.2. Adder on matrix structures of MUSP	216
9.3. Conclusion to the 9th chapter	218
Chapter 10. Methods for building reconfigured processors and computers on automatic memory circuits	219
10.1. Basic concepts	219
10.2. Methods for constructing a reconfigured architecture and processor structure on the MFIS and MUSP	219
10.3. Methods for describing the polygram	226
10.4. Principles for building reconfigured processors and computers that simultaneously process general and private information	231
10.5. Methods for building a reconfigured computer, taking into account the "elemental" level.	234
10.6. Accelerating the execution of algorithms in reconfigured computer systems on multi-level memory circuits	237
10.7. Conclusion to 10 chapter	238
Conclusion of 3 section	239
Section 4. NEURONS AND NEURAL NETWORKS	240
Chapter 11. Some concepts about neurons	240

11.1. Introduction	240
11.2. Preliminary concepts of the neuron model	241
11.3. Properties of human thinking	242
11.4. Fundamentals of short-term memory of the human brain	242
11.5. The problems of creating a model of the human brain	245
11.6. Information characteristics of the neuron of the human brain	249
11.7. Principles of constructing an artificial intelligence system	251
11.8. IBM's achievements in building a supercomputer on artificial neurons	252
11.9. Conclusion to the 11th chapter	254
Chapter 12. Artificial Elementary Neuron	255
12.1. Design of an artificial elementary neuron	255
12.1.1. IBM has created a chip with artificial neurons	255
12.1.2. Formulation of problem	256
12.1.3. General principles of designing a single category of an artificial neuron	257
on multi-level memory circuits	
12.2. Methods for designing a single category of an artificial neuron on multi-	264
level memory circuits	
12.3. Conclusion to 12 chapter	277
Chapter 13. Neural networks	279
13.1. The history of research in the field of neural networks	279
13.2. Basic concepts	280
13.3. Analysis of the structural organization of the neuron	283
13.4. Symbols for multifunctional and multi-level memory circuits	291
13.5. Structural organization of a neuron on automatic memory circuits	296
13.5.1. The main functional properties of the brain and neuron	296
13.5.2. Models of an artificial neuron on automatic circuit Memory	298
13.5.3. Structural organization of an elementary two-stage neuron on automat-	299
ic memory circuits	
13.5.4. Symbols of an elementary three-stage neuron on automatic memory	302
circuits	
13.6. Control schemes for an artificial neuron and communication between	304
neurons	
13.6.1. Introduction	304
13.6.2. Neuron structures with fixation of disability function	307
13.6.3. Existing functions of the neuron	310
13.6.4. Self-Improving Artificial Neuron Systems at the MUSP	315
13.7. Structures of axons of an artificial neuron with the ability to self-	319
reconstruct for the organization of connections between neurons	
13.7.1. Introduction	319
13.7.2. Structures of axons of an artificial neuron	321
13.8. Structure of short-term and long-term memory of artificial intellect and	323
determination of their functions	
13.8.1. Introduction	323

13.8.2. Characteristics of information and entropy	324
13.8.3. The model of the elementary structure of human existence	327
13.8.4. Number of hierarchical levels of short-term human memory	331
13.9. Structures of neural networks	331
13.10. Structural diagram of an artificial brain in multi-level memory circuits	334
13.10.1. Introduction	334
13.10.2. Description of the hierarchical structure of specialized	336
Blocks of neurons	
13.11. Blocks of hierarchical neurons	339
13.11.1. Introduction	339
13.11.2. Analysis unit	340
13.11.3. Typical Link Library	342
13.12. Conclusion to 13 chapter	343
14. CONCLUSION On a new scientific direction in the field of digital computer technology	345
14.1. Introduction	345
14.2. System of existing knowledge of digital computers	346
14.3. Representations of hierarchical information	348
14.4. A new system of knowledge in the field of computer technology	349
15. CONCLUSIONS	353
LIST OF REFERENCES	354

## LIST OF ACRONYMS

CT - computer technology;

AI - artificial intelligence;

SMP – multi-stable memory scheme;

MFIS - multifunctional memory circuits;

MUSP - multi-level memory schemes;

EA - is an elementary automaton;

$x(t)$  - setting the input signal;

$e(\Delta)$  - preserving the input signal;

$p(T)$  - is the input word;

$M$  - is the number of stable memorized states of the memory scheme;

$r_x$  - is the number of  $x(t)$  input signals in the memory scheme;

$r_e$  - is the number of  $e(\Delta)$  input signals stored in the memory scheme;

$F_p$  - limiting working frequency of switching;

$P_Q$  - load capacity on outputs;

$S_{cb}$  - number of internal connections;

$S_{vc}$  - is the number of external connections;

$L$  - is the number of elements per state;

*IEA* is a multifunctional elementary automaton;

*BA* - basic automaton;

$Y_1(t)$  - is the output signal of an automaton of the first kind;

$Y_2(T)$  - is the output signal of an automaton of the second kind;

$Y_3(\Delta)$  - is the output signal of the automaton of the third kind;

$p_0(T)$  is a single-valued input word;

$p_y(T)$  is an enlarged input word;

VLSI - (Very-large-scale integration);

CIN - is a digital artificial neuron.

## FOREWORD

A computer has been developing since 40-ies of the XX century and has proved a powerful factor in scientific and technological progress in all areas of human activity.

During this period, it has grown into a whole interdisciplinary field of industries, such as: large-scale integrated circuits with a billion components, production calculators, personal computers, supercomputers and neural computers, software, modern programming languages. All these industries of computer production implement new and new achievements of science in this direction.

For integrated circuits acted Moore's Law, which states that every two years the integration of large integrated circuits will increase. Since 2015 the law has ceased to act as the VLSI technology has reached its limit. A new era of new search features, departures from the traditional von Neumann structure and the revision of fundamentals of computer technology has begun.

Whole paradigms of scientific directions to the development of the foundations of computer science is being created. For example, American scientists claimed a large-scale simulation of the brain - a new interdisciplinary field that brings together computational neuroscience, supercomputers and simulation methodology. In the area of cognitive computer that works similarly to the brain, the cortex simulation is an extremely important technology for testing hypotheses about its structure, dynamics and function. Creating a computer-based cognitive neurons - a rejection of the paradigm of von Neumann computing.

Such fundamentals of computer technology, as the theory of sequential automata of Mealy and Moore, automatic discrete time, theorem on the functional completeness of the elementary automata proposed by academician V.M. Glushkov, became the foundation for the construction of new elements, which are designed on the basis of modern integrated circuits, the binary memory circuits, all of them need to be expended.

In 2015, Professor V.K. Promonenkov proved that the information is in the same part of the universe as matter and motion and it is hierarchical. It made us to change the way of using the computers, which focuses on single-level information processing using only one input signal into discrete automaton time  $x(t)$ .

The reference book was made up of experiences in the scientific school of developing methods for designing of computer devices based on the elements of an automaton memory, which has ongoing seminar at the department "Automation and Computer-Integrated Technologies of Transport" in the State Economy and Technology University of Transport. The handbook examines the foundations of a new para-

digm of information technology based on schemes automaton memory - new interdisciplinary direction for processing of hierarchical information.

This direction combines the theory of multi-function automats, the theory of the synthesis of an automaton memory circuits, basic principles of analysis of memory circuits after catastrophic rejection, methods of logic designing of reconfigurable computer devices and digital designing principles of artificial neuron. The authors hope that the foundations of the new information technologies based on the automaton memory schemes will help to design the competitive devices. It will expand the functionality of computer devices, increase their reliability and survivability, and allow the identification of a catastrophic failure in the basic memory circuits of reconfigurable computer devices which is implemented with memory for triggers, from the beginning of their industrial use.

Due to the novelty of the theoretical approach of reference material, before each section is given a paragraph where all disadvantages of computer devices with memory based on triggers are listed. The following section of this chapter highlights new scientific achievements and new reconfigurable computer digital devices. At the end of this section it is shown the comparative characteristics of these reference materials and the well-known computing devices.

The novelty of the material that was presented is certainly arouse the interest of developers of new computer technology, will give impetus to new research work on the improvement of fundamentals of computer technology, and will be useful to researchers, teachers and students of higher educational institutions.

# **SECTION 1 THEORY OF ABSTRACT HIERARCHICAL AUTOMATES**

## **Chapter 1 MULTIFUNCTIONAL AND MULTILEVEL ABSTRACT AUTOMATAS**

### **1.1. Formulation of the problem**

The theory of finite automata is characterized by a wide use of discrete techniques in various fields of application. This theory was first developed on the basis of Boolean algebra and the model of a discrete device in the form of a so-called finite automaton. It is based on the development of methods for the logical design of discrete devices and methods for constructing tests to test the latter, ensuring the reliability and stability of their work, solving the problems of "designing" discrete devices. There were separate branches of the theory of finite automata in the form of the theory of probability and fuzzy automata, the collective behavior of automata, experiments, etc. [1-16].

The theory of automata is a section of the theory of control machines, which studies mathematical models of converters of discrete information, called automata (from the Greek *atutómatos* - self-acting). From a theoretical point of view, converters are both real devices (computers, automata, living organisms, etc.), and abstract systems (mathematical machines, axiomatic theories, etc.). A characteristic feature of these converters is the discreteness of the functioning and the finiteness of the ranges of the values of the parameters describing them.

The fundamental foundations of the theory of sequential automata mathematically describing real devices with memory on the triggers are:

1. The theory of sequential abstract automata Miley and Moore [6].
2. Discrete automaton time, where the "empty word of zero length" is not used [7].

3. A theorem on the structural completeness of elementary automata, which substantiated the elemental base from the first computers to modern integrated circuits of modern computers [6].

At the time his theory has a number of fundamental limitations:

1. The automates of Miley and Moore are not able to reconstruct the structure of the memorized states [17].

2. The discrete automatic time, in which the "empty word of zero length" is not used, allows a single unique transition from one state to another during the information input signal  $x(t)$ , which limits the possibilities of describing hierarchical information in devices [17].

3. The theorem on structural completeness uses the structure of Moore's automaton with non-trivial memory (trigger), which has zero information redundancy, low reliability and has no ability to store hierarchical information [17].

Thus, in this chapter will be considered theories of multifunctional and multi-level automata that describe computer devices in automatic continuous time which is capable to process hierarchical information unlike the Miley and Moore automata that are operating in automatic discrete time.

To solve the expansion of the theory of successive automata of Miley and Moore, the following problems were posed and solved:

1. Expansion of the automatic discrete time to the level of uninterrupted.

2. Development of the theory of multifunctional automata of the 1st, 2nd and 3rd kind using new types of transitions in two variables, using an "empty word of zero length" of automatic continuous time.

3. Development of the theory of multilevel parallel automata on the basis of multifunctional automata.

4. Give comparative characteristics of the proposed theories with the theory of Miley and Moore automata and show their advantages.

In this case, deterministic enlarged transitions, probabilistic and fuzzy transitions will be considered. The classification of the considered automata is given. These new fundamental foundations have made it possible to increase the reliability, ca-

capacity of reconfigurable hierarchical devices, and also to raise the computer intelligence of computing devices capable of simultaneously processing hierarchical information.

## 1.2. Automatic continuous time

In modern discrete automata it is customary to identify the letters of the standard alphabet used with the digits of one or one number system (most often binary or decimal). Therefore, discrete automata are usually called digital automata.

The basic quality that separates discrete automata from among all other information converters is the presence of a discrete (in this case in real devices always finite) set of internal states and the properties of a jump-like transition from one state to another during the time  $t$  of the machine clock  $T$  (Figure 1.1), Which takes no more than two or three delays  $\tau_{\text{э}}$  of one logical element. A jumble transition means the ability to treat it as an instantaneous (to a certain degree of abstraction). Such an abstraction describes quite well the basic properties of real digital automatic devices (primarily computer devices and computers with memory on triggers) and therefore such an abstraction is adopted for constructing the theory of digital automata.

The second assumption is that the transition to the next state is possible after the transient process in the automaton memory circuits that have a spread of parameters, but for a given automaton it is some fixed time interval, which is usually chosen equal to four delays  $\tau_{\text{э}}$  of one element. The interval between cycles  $t$  in the theory of automata is usually called an "empty word of zero length", because under the influence of the word  $e$  the automaton is not able to go from one state to another, but only performs the function  $\delta e$  of maintaining the established state [6-7]. Such an assumption makes it possible to consider the operation of digital automata with memory based on triggers in the discrete automaton time  $t_i$  (Fig. 1.1, *a*) [1; 3-6; 12; 14-15].

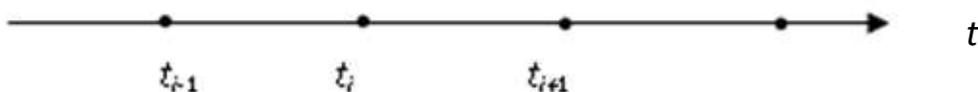
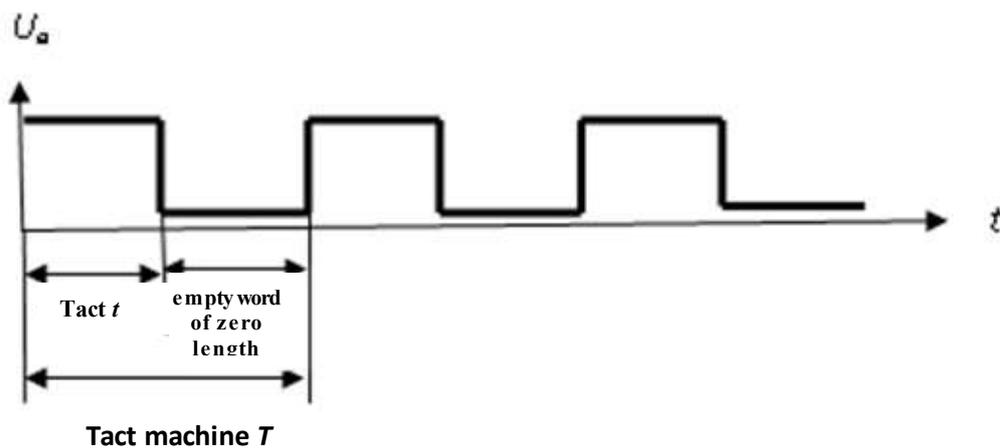


Fig. 1.1, *a*. Points on the digital axis



**Fig. 1.1, b.** Machine tact

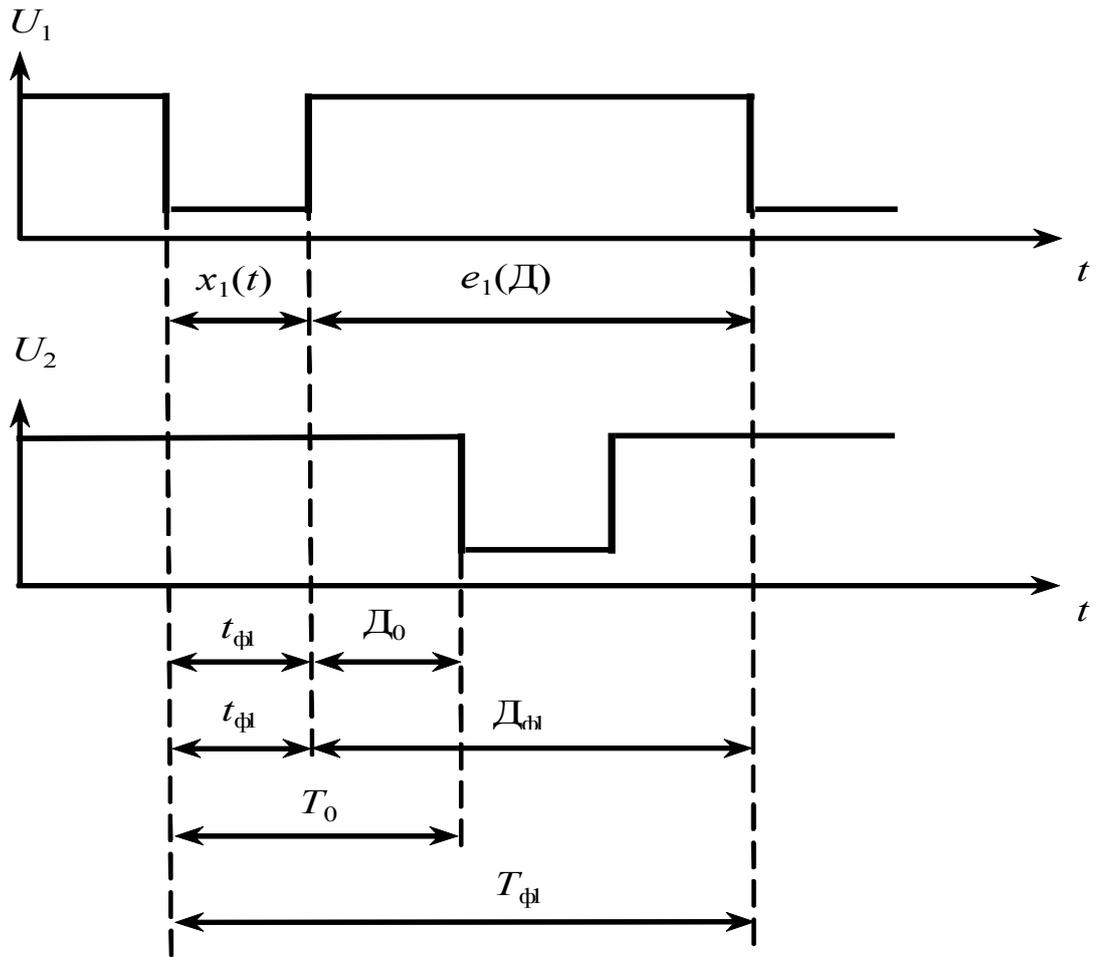
The theory of asynchronous automata differs significantly from the theory of synchronous automata in that it considers not only the moments of actual transitions, but also those transitions that are possible at the given moment, but have not yet occurred. Among such moments, the arrival times of the input signals  $x(t)$  of the impulse type (instantaneous) and the voltage level of the signals of the potential type operating in the time interval  $t$  of the clock cycle (figure 1.1, *b*) are reckoned. It is believed that the interval of discreteness of the automaton limits the minimum possible distance between the additionally introduced moments of the automaton time. With this assumption, the theory of asynchronous automata can in some cases be reduced to a synchronous case, since in fact the lengths of intervals between successive moments of discrete automaton time in the idealized theory of automata (without taking into account transient processes) have no significance, since the automaton at that moment receives an empty Word  $e$  of length zero. With this in mind, the abstract discrete automaton time is used in the operation of automata with memory on triggers, taking non-negative integer values:  $t = 0, 1, 2, \dots, n, \dots$ , and constructing a theory like the theory of consecutive synchronous automata, Although in reality it covers to a large extent the theory of asynchronous automata.

Transitions from one  $a_i$  state of the automaton to another  $a_k$  are caused by the information input signals  $x(t)$  arising outside the automaton and transmitted to the automaton over a finite number of input channels. Two assumptions are made with respect to  $x(t)$  of the input signals of digital automata: first, for any digital machine, the number of different  $x(t)$  input signals is of course finite, and, secondly,  $x(t)$  From one state to another and refer to the moments of time determined by the corresponding transitions. Especially it should be emphasized that the real physical input signal that causes the state of the automaton to change at time  $t_i$  may end before this moment, however, nevertheless, it refers specifically to the time  $t_i$ , not  $t_{i-1}$ .

The result of the operation of the digital machine is the output of the output signals  $y$  transmitted from the machine to external circuits over a finite number of output channels. With respect to the output signals, we introduce assumptions analogous to those that were made for the case of the input signals  $x$ . First, for any digital machine, the number of different output signals  $y$  is necessarily finite, and, secondly, to each time different from zero, the automatic time is the corresponding output signal  $y$ .

In multifunction automatic machines of Marakhovskiy, which are named after the name of the author who proposed them, automatic memory is used, automatic continuous time is introduced and used (Fig. 1.2) [18-20].

In automatic continuous time, in addition to the clock  $t_i$ , the interval between the  $t_i$  ticks is used, the interval of which is denoted by the symbol " $\Delta$ ". This is explained by the fact that the input signal  $e(\Delta)$  is used in the interval, which is called in the automata with memory on the flip-flops "empty word of zero length" and is not taken into account in the synthesis of automata. In multipurpose automatons, the input signal  $e(\Delta)$  can be used to implement the enlarged transitions in the automaton memory of the machine.



**Fig. 1.2.** Time relations of automatic continuous time and synchronized input signals

Let us consider an automatic continuous time with allowance for the synchronous signals  $\tau_j$  ( $j = 1, 2$ ) and give a number of definitions.

*Definition 1.1.* This  $t_{\tau_i}$  is the time interval during which an arbitrary synchronizing signal  $\tau_j$  can be fed to the machine

*Definition 1.2.* The inner clock  $\Delta_{\tau_j}$  will be the smaller open time interval between the appearances of the synchronizing signals  $\tau_j$ .

*Definition 1.3.* An outer clock  $T_{\tau_j}$  will be the smaller right-open time interval, which corresponds to the period of the synchronized signal  $\tau_j$ .

The measures that are defined can be expressed by the following formula:

$$T_{\tau_j} = t_{\tau_j} + \Delta_{\tau_j}; \quad j = 0, 1, 2, \dots, n, \dots \quad (1.1)$$

The information input signals  $x(t)$  are fed to the input channels of the complex automaton and synchronized by the signal  $\tau_j$ . These input signals act on the automaton in the automatic continuous time of the measure  $t_{\tau_j}$ .

*Definition 1.4.* The inner unit clock cycle  $\Delta_0$  of the automatic continuous time is the smaller open time interval between the appearance of two arbitrary and consecutive synchronized signals  $\tau_p$  and  $\tau_{p+1}$ .

*Definition 1.5.* The outer unit clock cycle  $T_0$  is the smaller right-open time interval, which corresponds to the time interval between the appearance of two arbitrary and consecutive synchronizing signals,  $\tau_p$  and  $\tau_{p+1}$ .

The external unit cycle  $T_0$  in accordance with the time is equal to the sum of the duration of the cycle  $t_{\tau_j}$  and the internal unit cycle  $\Delta_0$ . This relationship is considered by this equation:

$$T_0 = t_{\tau_j} + \Delta_0. \quad (1.2)$$

The duration of the outer measure  $T_{\tau_j}$  is equal to the duration of the outer clock cycle  $T_a$  of the automaton. For one external clock cycle  $T_a$ , all synchronizing signals  $\tau_j$  affect the automaton once.

*Definition 1.6.* The outer clock cycle  $T_a$  of an automaton is the smaller right-open time interval, which corresponds to the period of action of all the synchronizing signals  $\tau_j$  ( $j = 1, 2, \dots, C$ ) on the input channels of the automaton.

An automaton, which describes the real device of a computer, must satisfy the simplest requirements for the reliability and stability of the states of the  $a_i$  and output signals  $y_i$ .

It is necessary to take into account that signals of a certain duration  $\tau_j$  enter the input channels of elementary automata.

Under the influence of the input signal  $x(t)$  during the clock  $t_{\tau_j}$ , an elementary automaton with memory can go into a definite state when implementing a single-valued transition. Under the influence of the input signal  $e(\Delta)$  during the internal unit cycle  $\Delta_0$ , an elementary multifunctional automaton with memory can go into a new state when implementing the function of an enlarged, probabilistic or fuzzy transition.

The new established state of an elementary automaton can be preserved after the end of a cycle  $t_{\tau_j}$  for a single transition or after an internal unit cycle  $\Delta_0$  for the realization of the functions of an aggregated or probabilistic transition.

The timing relations of the synchronizing and input signals are shown in Fig. 1.2.

The output signal  $y_j$  of the elementary automaton fixes a new stable state in the automaton, which remains unchanged over the entire time interval  $T_{\tau_j}$ . The segment  $T_{\tau_j}$  is determined according to the following formula:

$$T_{\tau_j} = \Delta_{\tau_j} - \Delta_0. \quad (1.3)$$

The outer cycle  $T_a$  of the machine with its duration equal to the sum of the outer unit cycles  $T_0$ :

$$T_a = \sum_{j=1}^c (T_0)_j. \quad (1.4)$$

The automatic continuous time gives a more complete possibility to investigate the law of the functioning of the Mealy and Moore automata realized on triggers than the discrete automaton time  $t_i$ . However, for the study of the multifunctional automat-

ic machines of Marakhovsky implemented on automatic memory circuits, the possibilities of automatic discrete time are insufficient. In connection with this, an automatic continuous time is introduced, in which the input word  $p(T)$  consisting of two successive input signals  $x(t)$  and  $e(\Delta)$  is taken into account [17].

The input signals  $x(t)$  and  $e(\Delta)$  of the input word  $p(T)$  have one important property: when they intersect, the input signal  $x(t)$  absorbs the signal  $e(\Delta)$ . In other words, the formula works:

$$x(t) \cup e(t) = x(t) \quad (1.5)$$

Summing up the analysis of automatic continuous time, it can be stated that it expands the concept of automatic discrete time, which is oriented to determining the operation of automata with memory on the triggers, and besides it allows us to consider the work of Marakhovsky automata on automatic memory circuits [17-20].

### 1.3. Multifunctional deterministic automata

The functioning of abstract multifunctional automata of Marakhovsky built on automatic memory circuits is considered in automatic continuous time (Figure 1.2) [18-20].

For a better understanding of the use of automatic continuous time, we consider in it the work of successive monofunctional Mealy and Moore automata.

*Definition 1.7.* A monofunctional abstract automaton that includes, in the form of components, an empty word  $e$  and a state conservation function  $\delta_e$  is described by an eight-component vector:

$$A = (X, Y, Q, \delta, \lambda, \delta_e, a_0, e), \quad (1.6)$$

which one:

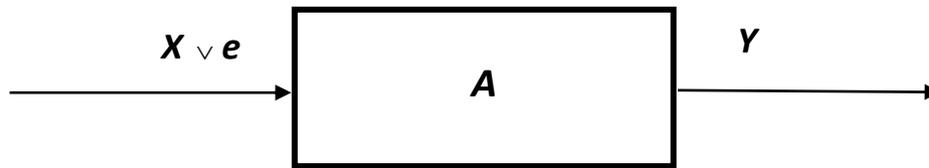
- the components  $X, Y, Q, \delta, \lambda$  and  $a_0$  have the same meanings and the same meaning as in the description of the six-component vector of Mealy and Moore automata [6];

- $e (e \notin X)$  - preserving the input signal;

- $\delta_e: Q \times e \rightarrow Q$  is a state conservation function that realizes a map  $D_{\delta_e} \subseteq Q \times e$  on  $Q$ , under which an arbitrary state  $a_i (a_i \in Q)$  retains its value.

The empty word  $e(\Delta)$ , called the conserved input signal, acts on its input nodes in the automaton  $A$  during the absence of input signals  $x(t)$ , where  $x \in X$ . When the input signals  $x(t)$  and  $e(t)$  is absorbed by the signal  $x(t)$ , as reflected in formula (1.8):

The structure of an eight-component monofunctional abstract automaton  $A$  in a



**Fig. 1.3.** The structure of the automaton  $A$  with memory on the triggers

generalized form is shown in Fig. 1.3.

In real devices, a successive pair of input signals  $x(t)$  and  $e(\Delta)$  arrive at the input nodes, which determine the elementary input word  $p(T) = x(t), e(\Delta)$ . When using as triggers for memory, all the states of the memory elements are remembered only with one saving  $e(\Delta)$  input signal. This preserving  $e(\Delta)$  input signal is not able to change the state of the automaton. This explains the fact that in the automata in which triggers are used as memory, the input signal  $e(\Delta)$  is omitted, and the functioning of the abstract automaton itself is considered in automatic discrete time.

However, at the level of the abstract theory of the Marakhovsky automata, which we will henceforth simply call the automaton  $M$ , with a multifunctional memory organization, we use transitions in two variables  $x(t)$  and  $e(\Delta)$  in automatic memory circuits [18-20], the value of the preserving  $e(\Delta)$  of the input signal must be

taken into account and used to consider the operation of automata in automatic continuous time [17; 21].

The automatic memory scheme can be conditionally represented in the form of a matrix in which the columns are subsets of the  $\mu_i (\cup_j \mu_j = Q)$  states of the automaton, and the rows are the subsets  $\pi_j (\cup_j \pi_j = Q)$  of the states of the automaton (Table 1.1).

Table 1.1

The state matrix automatic memory circuits

	$\mu_1$	$\mu_2$	.....	$\mu_n$
$\pi_0$	$a_{10}$	$a_{20}$	...	$a_{n0}$
$\pi_1$	$a_{11}$	$a_{21}$	...	$a_{n1}$
$\pi_2$	$a_{12}$	$a_{22}$	...	$a_{n2}$
...	...	...	...	...
$\pi_m$	$a_{1m}$	$a_{2m}$	...	$a_{nm}$

Triggers and memory memory circuits (SMPs) can be represented as a string  $\pi_0$  automata matrix (Table 1.1) because all its  $a_i$  states are preserved with one saving  $e(\Delta)$  input signal in one set of all its states. These circuits have the completeness of the transition functions, when from each  $a_k$  state it is possible, under the influence of a certain input signal  $x(t)$ , to go to any, preassigned,  $a_i$  state of the memory scheme; As well as the completeness of the output system, when each  $a_i$  state is identified with the output  $y_i$  signal, which is different from all others.

In the multifunctional automatic memory circuit [17], transitions at the instant  $t$  under the influence of the input  $x(t)$  signals can occur from one state to another in a certain subset  $\pi_i$ , and at times  $\Delta$  of the automatic continuous time  $T$  under the influence of the input  $e(\Delta)$  of the signal, transitions from one state to another occur in a certain subset  $\mu_i$  of states of the automaton. Thus, in the matrix scheme of automata

memory, transitions are possible with respect to two variables  $x(t)$  and  $e(\Delta)$  in one machine cycle  $T$  of automatic continuous time [21].

*Definition 1.8.* A mathematical model of a discrete device with a multifunctional memory organization on automatic memory circuits is a multifunctional abstract automaton  $M$  defined as a fifteen-component cortege or vector [21]:

$$A = (X, E, Y_I, Y_{II}, Y_{III}, Q, \pi, e_0, a_0, \delta_o, \delta_e, \delta_y, \lambda_1, \lambda_2, \lambda_3), \quad (1.7)$$

which one:

- $X = \{x_0, x_1, \dots, x_{N-1}\}$  is the set of information input signals (input information alphabet);
- $E = \{e_0, e_1, \dots, e_{R-1}\}$  is the set of conserved input signals (input preserving the alphabet);
- $Y_I = \{Y_0^1, Y_1^1, \dots, Y_{K_1-1}^1\}$  - set of output signals of type 1 (output alphabet of type 1);
- $Y_{II} = \{Y_0^2, Y_1^2, \dots, Y_{K_2-1}^2\}$  - set of output signals of type 2 (output alphabet of type 2);
- $Y_{III} = \{Y_0^3, Y_1^3, \dots, Y_{K_3-1}^3\}$  - set of output signals of type 3 (output alphabet of type 3);
- $Q = \{a_0, a_1, \dots, a_{m-1}\}$  is an arbitrary set of states (the alphabet of states);
- $\pi = \{\pi_0, \pi_1, \dots, \pi_{R-1}\}$  is the set of blocks of  $\pi_j$  states (the alphabet of state blocks);
- $e_0 \in E$  is the initial storing input signal;
- $a_0 \in \pi_0$  ( $\bigcup_j \pi_j = Q$ ) is the initial state of the automaton;
- $\delta_o: Q \times X \rightarrow Q$  is a single-valued transition function that realizes a map  $D_{\delta_o} \subseteq Q \times X$  in  $Q$ . In other words, the function  $\delta_o: (a(\Delta-1), x(t))$ , to some pairs "state - information input  $a(t) = \delta_o(a(\Delta-1), x(t))$ , where  $a \in Q$ ;
- $\delta_e$ : the conservation function of the states of the block  $\pi_j$  at  $a \in \pi_j$ , realizing the map  $D_{\delta_e} \subseteq Q \times e_j$  in  $\pi_j$ . In other words, the function  $\delta_e: (a(t), e(\Delta))$  to some pairs

"state - preserving the input signal" puts in correspondence the state of the automaton  $a(\Delta) = \delta_e(a(t), e(\Delta))$ , where  $a \in \pi_j$ ,  $a(t) = a(\Delta)$  ( $\pi_j \in Q$ );

- $\delta_y: Q \times E \rightarrow \pi_j$  is the function of aggregated transitions realizing the map  $D_{\delta_y} \subseteq Q \times E$  in  $\pi_j$ . In other words, the function  $\delta_y: (a(t), e(\Delta))$  for some pairs "state-preserving the input signal" assigns the state of the automaton  $a(\Delta) = \delta_y(a(t), e(\Delta))$ , where  $a(t) \in \pi_p$ ,  $a(\Delta) \in \pi_j$ ,  $a(t) \neq a(\Delta)$  ( $a(t), a(\Delta) \in Q$ );

- $\lambda_1: Q \times X \rightarrow Y_I$  - output function of type 1, realizing the mapping  $D_{\lambda_1} \subseteq Q \times X$  in  $Y_I$ . In other words, the function  $\lambda_1: (a(\Delta-1), x(t))$  to some pairs "state - information input signal" puts in correspondence the output signal of the automaton  $y(t) = \lambda_1(a(\Delta-1), x(t))$ , where  $y \in Y_I$ ;

- $\lambda_2: Q \rightarrow Y_{II}$  is a function of outputs of type 2 that implements the mapping  $D_{\lambda_2} \subseteq Q$  in  $Y_{II}$ . In other words, the function  $\lambda_2: (a(t), a(\Delta))$ , to some pairs "state-state", which are equal to each other  $a(t) = a(\Delta)$ , corresponds to the output signal of the automaton  $y(T) = \lambda_2(a(t), a(\Delta))$ , where  $y \in Y_{II}$ ;

- $\lambda_3: Q \times E \rightarrow Y_{III}$  - output function of type 3, realizing the mapping  $D_{\lambda_3} \subseteq Q \times E$  in  $Y_{III}$ . In other words, the function  $\lambda_3: (a(\Delta), e(\Delta))$  to some pairs "state - preserving the input signal" corresponds to the output signal of the  $y(\Delta) = \lambda_3(a(\Delta), e(\Delta))$ , where  $y \in Y_{III}$ .

Abstract multifunctional automata  $M$  represent the union of automata of the 1st, 2nd and 3rd kind, function in the automatic continuous time  $T_i$ , which consists of two segments  $t_i$  and  $\Delta_i$  ( $T_i = t_i + \Delta_i$ ), described earlier (see Fig. 1.1).

At each time  $\Delta_i$ , the automaton is able to perceive the input  $e(\Delta)$  as a co-decreasing signal - an arbitrary letter of the input preserving alphabet  $E$  and is in a certain state  $a(\Delta)$  of the subset  $\pi_j$  from the set  $Q$  ( $\pi_j \in Q$ ) of states of the automaton, and at the initial instant of time  $\Delta = 0$ , the initial automaton is always in its initial state  $a_0$ , that is,  $a(0) = a_0$ .

At each instant  $t_i$ , the multifunctional automaton of the first kind is able to perceive the input  $x(t)$  information signal - an arbitrary letter of the input information

alphabet  $X$  and carry out a transition to a new state  $a(t)$ , that belongs to a certain subset of states  $\pi_j$ , which is conserved at the input signal  $e_j(\Delta)$  and the  $Q$  ( $\pi_j \in Q$ ), entering the set of states, and also output the corresponding output signal  $y_1(t)$  - some letter of the output alphabet  $Y_I$ .

The law of the functioning of a multifunctional abstract automaton in automaton continuous time in the case of an automaton of the first kind is given by the equations:

$$\begin{cases} a(t) = \delta_0(a(\Delta - 1), x(t)); \\ a(\Delta) = \delta_e(a(t), e(\Delta)); \\ y_L^1(t) = \lambda_1(a(\Delta - 1), x(t)), \\ a(t), a(\Delta) \in \pi_j; \quad i = 0, 1, 2, \dots; \quad \Delta = 0, 1, 2, \dots \end{cases} \quad (1.8)$$

At each instant  $t_i$ , the multifunctional automaton of the second kind is able to perceive the input  $x(t)$  information signal - an arbitrary letter of the input information alphabet  $X$  and carry out a transition to a new state  $a(t)$ , which belongs to a certain subset of states  $\pi_j$ , which persists for the input signal  $e_j(\Delta)$  and the  $Q$  ( $\pi_j \in Q$ ), entering the set of states, and also output the corresponding output signal  $y_2(T)$  - some letter of the output alphabet  $Y_{II}$ .

The law of the functioning of a multifunctional abstract automaton in automaton continuous time in the case of an automaton of the second kind is given by the equations:

$$\begin{cases} a(t) = \delta_0(a(\Delta - 1), x(t)); \\ a(\Delta) = \delta_e(a(t), e(\Delta)); \\ y_L^2(T) = \lambda_2(a(t), a(\Delta)), \\ a(t), a(\Delta) \in \pi_j; \quad i = 0, 1, 2, \dots; \quad \Delta = 0, 1, 2, \dots \end{cases} \quad (1.9)$$

At each time  $t_i$  multifunctional machine of the 3rd kind is able to perceive the input  $x(t)$  data signal - an arbitrary letter input information alphabet  $X$  and make the transition to a new state  $a(t)$ , and then is able to perceive the continued input signal

$e_j(\Delta)$  - preserving arbitrary letter alphabet input  $E$  and carry at each moment  $\Delta_i$  transition to a new state  $a(\Delta)$  included in the subset  $\mu_i$  of the set of states  $Q$  ( $\mu_i \in Q$ ), and issue a corresponding output signal  $y_3(\Delta)$  - some starting weekend about alphabet  $Y_{III}$ .

Law functioning multipurpose abstract machine in automatic uninterrupted time in the case of automatic third kind given by the equations:

$$\begin{cases} a(t) = \delta_0(a(\Delta - 1), x(t)); \\ a(\Delta) = \delta_y(a(t), e(\Delta)); \\ y_L^3(\Delta) = \lambda_3(a(\Delta), e(\Delta)), \\ a(t) \notin \pi_j, a(\Delta) \in \pi_j; \quad i = 0, 1, 2, \dots; \quad \Delta = 0, 1, 2, \dots \end{cases} \quad (1.10)$$

By the establishment of the laws of the functioning of multifunctional abstract automata of the 1st, 2nd, and 3rd kinds of a generalized abstract automaton  $M$ , the definition of an abstract automaton terminates.

The meaning of the concept of a multifunctional abstract automaton of the first kind consists in the realization of a map  $\varphi_1$  of a consecutive set of elementary  $p$  words consisting of the letters of the input information alphabet  $X$  and the letters of the input conserved alphabet  $E$  into the set of words of the output alphabet  $Y_I$ .

Each elementary input word  $p(p = x, e)$  is sequentially fed to the input of a given multifunctional abstract automaton  $M$ , previously set to the initial state. At the time  $t$ , under the influence of the input  $x(t)$  of the input information alphabet  $X$ , the multifunctional automaton  $M$  is able to go into the new state of the subset of states  $\pi_j$  that is conserved under the input letter  $e_j(\Delta)$  of the input preserving alphabet  $E$  of the automaton  $M$ , and the state  $Q$  ( $\pi_j \in Q$ ), and also output the corresponding output signal  $y^I(t)$  - some letter of the output alphabet  $Y_I$ .

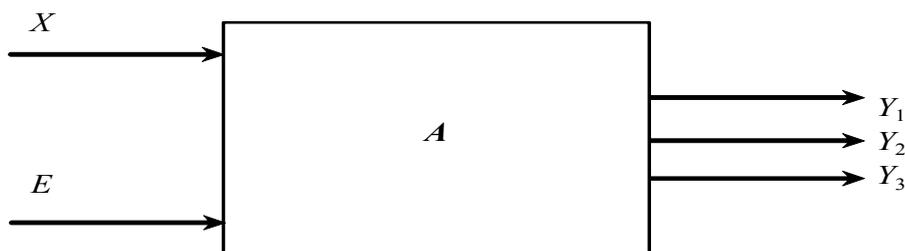
The meaning of the concept of a multifunctional abstract automaton of the second kind consists in the realization of a map  $\varphi_2$  of a consecutive set of elementary  $p$  words consisting of the letters of the input information alphabet  $X$  and the letters of the input preserving alphabet  $E$ , and also output the corresponding output signal  $y^2(T)$  - a letter of the output alphabet  $Y_{II}$ .

The meaning of the notion of an abstract automaton of the third kind consists in the realization of a map  $\varphi_3$  of a consecutive set of elementary  $p$  words consisting of the letters of the input information alphabet  $X$  and the letters of the input preserving alphabet  $E$  into the set of words of the output alphabet  $Y_{III}$ .

The finite sequences of input elementary words  $p(T) = x(t), e(\Delta)$  arising in this way, on the basis of the laws of automaton operation in the automaton continuous time  $T$ , respectively, cause respectively definite sequences  $q_i = \varphi_i(p)$  of output letters from the corresponding input alphabets  $Y_I, Y_{II}, Y_{III}$ . This sequence  $q_i$  is called the output word of the automaton of the 1-st, 2-nd or 3-rd kind of the generalized multifunctional automaton  $M$ .

Thus, each sequence of input elementary words  $p$  has the corresponding sequence of the output word  $q_i$ , we obtain the required mapping  $\varphi_i$ , which is called the mapping induced by the abstract automaton  $M$ , which is able to work as an automaton of the 1-st, 2-nd or 3-rd kind.

The abstract  $M$ -automaton  $A$ , which is described by the vector (1.10), has two inputs  $X$  and  $E$  and three outputs  $Y_I, Y_{II}, Y_{III}$ . (Fig. 1.4). The functioning of the automaton is investigated in the automaton continuous time  $T$  [17; 21].



**Fig. 1.4.** The block diagram of an abstract  $M$ -automaton  $A$

An abstract machine describing a real computer device must satisfy the elementary requirements of reliability, stability of states and output symbols (words). Here it is necessary to depart from the purely abstract concept of the symbols of the input

alphabet and take into account that the input channels (nodes) of the real device receive setting and preserving signals of a very definite duration. Under the influence of any setting input signal  $x(t)$ , the device goes into a completely defined state  $a(t)$  and must remain in it until the effect of this input signal ends. At the level of the abstract model, this means that if the automaton has entered the state  $a(t)$  when the input signal  $x(t)$  arrives, then it must remain in it while the duration of the input signal  $x(t)$  is preserved (repeated exposure of the symbol  $x$  for a period of time  $T$  in state  $a$ ).

During the period of action of the preserving input signal  $e(\Delta)$ , the state  $a(t)$  is conserved ( $a(t) = a(\Delta)$ ) in the first and second kind automata, or an enlarged transition to the new state  $a(\Delta)$  occurs, then Is the state  $a(t) \neq a(\Delta)$ , in automata of the third kind. The meaning of multifunctionality lies in the fact that automata of the first and second kind can change the algorithm of its functioning when the input signal is changed  $e(\Delta)$ . The fulfillment of this condition is also necessary for orientation to a definite realization of the automaton in the given elemental basis [21].

An abstract machine describing a real computer device must satisfy the elementary requirements of reliability, stability of states and output symbols (words). Here it is necessary to depart from the purely abstract concept of the symbols of the input alphabet and take into account that the input channels (nodes) of the real device receive setting and preserving signals of a very definite duration. Under the influence of any setting input signal  $x(t)$ , the device goes into a completely defined state  $a(t)$  and must remain in it until the effect of this input signal ends. At the level of the abstract model, this means that if the automaton has entered the state  $a(t)$  when the input signal  $x(t)$  arrives, then it must remain in it while the duration of the input signal  $x(t)$  is preserved (repeated exposure of the symbol  $x$  for a period of time  $T$  in state  $a$ ).

During the period of action of the preserving input signal  $e(\Delta)$ , the state  $a(t)$  is conserved ( $a(t) = a(\Delta)$ ) in the first and second kind automata, or an enlarged transition to the new state  $a(\Delta)$  occurs, then Is the state  $a(t) \neq a(\Delta)$ , in automata of the third kind. The meaning of multifunctionality lies in the fact that automata of the first and second kind can change the algorithm of its functioning when the input signal is

changed  $e(\Delta)$ . The fulfillment of this condition is also necessary for orientation to a definite realization of the automaton in the given elemental basis [21].

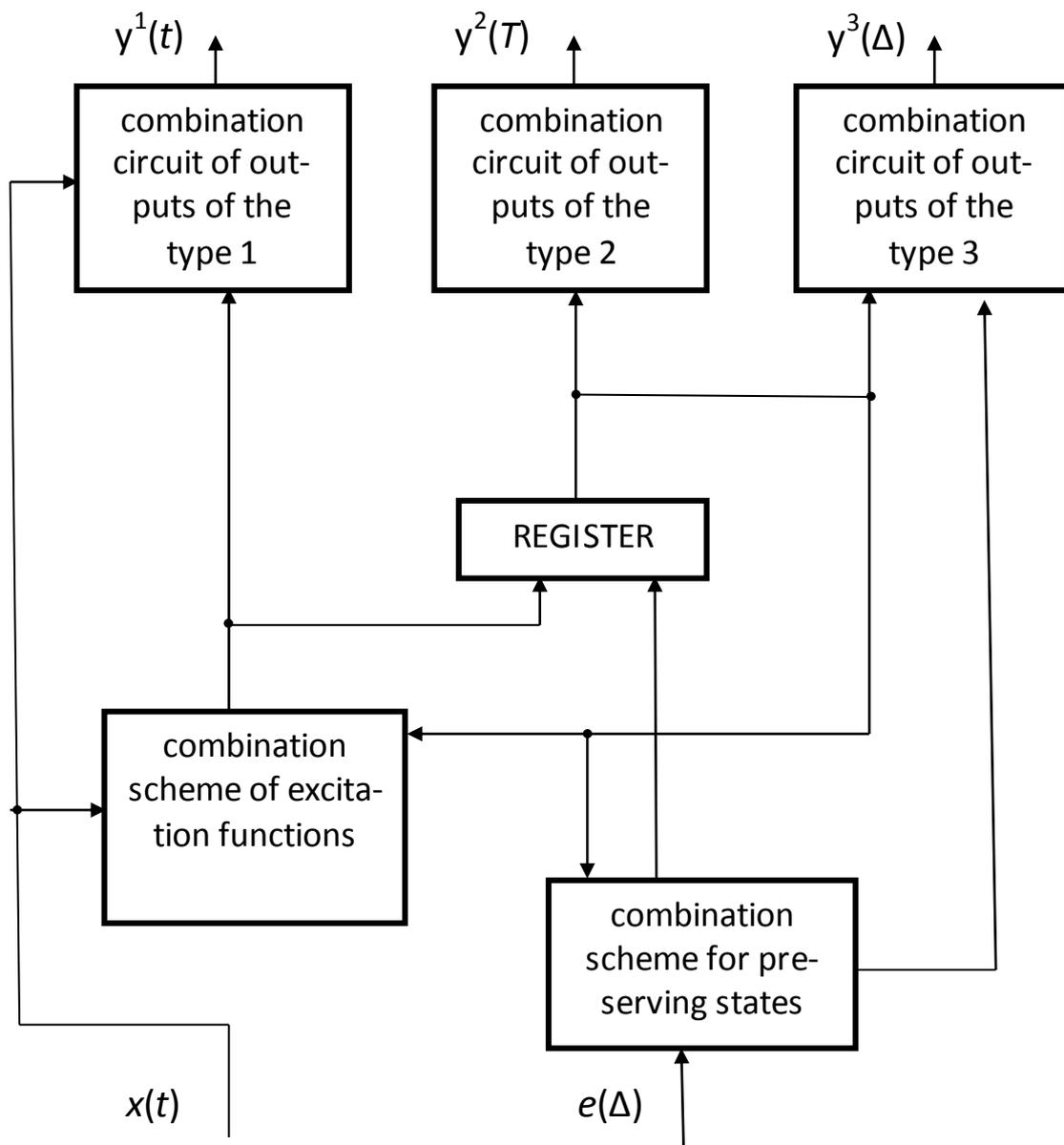
For the correct (functionally reliable) operation of the automaton  $A$ , it is impossible to allow the input nodes of the memory circuit to receive signals that took part in the creation of the output signals and through the feedback loop would be fed at the same time  $t$  to the input nodes of the same circuits Memory. In this regard, it is reasonable to use delays in the memory of the automaton for the appearance of feedback input signals at the next instant of time  $t + 1$ . An example of using a delay line in memory circuits can be two-stage triggers or registers, which use the second memory stage as a delay. Each stage in the register is clocked by the synchronous pulses  $\tau_i$  ( $i = 1, 2$ ) of only one series, or separate groups of memory circuits, each of which has input signals that establish  $x(t)$ , which are clocked by the synchronous pulses  $\tau_i$  ( $i = 1, 2$ ) of only one series. When using several groups of memory circuits, you can use the outputs of other groups to create feedbacks, which at the moment  $t$  have stable output signals.

The block diagram of the multifunction machine  $M$  is shown in Fig. 1.5.

Thus, the block diagram of the multifunction machine  $M$  consists of a circuit for automatic memory (register) and five combinational circuits. It is represented by a combination scheme of the memory excitation functions of the first stage of the register, which preserves the combinational memory scheme of the first stage of the register, the combinational circuit of the outputs of type 1, symbolizing the automaton of the first kind, the combination circuit of the outputs of type 2, symbolizing the automaton of the second kind, and the combination circuit of outputs of the type 3, symbolizing an automaton of the third kind

Summarizing the results of the consideration of multifunctional automata of the 1st, 2nd, and 3rd kind, it can be noted that the automata of the 1st and 2nd kind are capable of rearranging the subsets of the  $\pi_j$  remembered states under the influence of the input e-signals that preserve  $e(\Delta)$ . The automata of the third kind are able to carry out transitions from one state  $a_i$  to another state  $a_k$  under the influence of the input word  $p(T) = x(t), e(\Delta)$ . The disadvantage of these automata is the need for generation

of  $e(\Delta)$  preserving input signals. This drawback is eliminated in the construction of multi-level automata, which will be considered later.

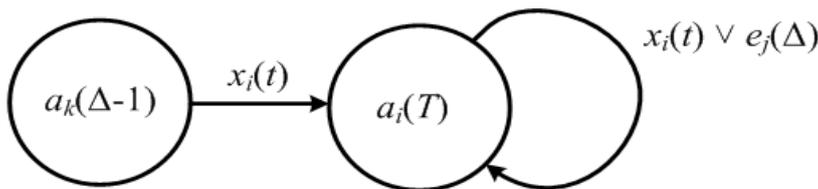


**Fig. 1.5.** The structure of the structural scheme of an abstract multi-functional  $M$ -automaton  $A$

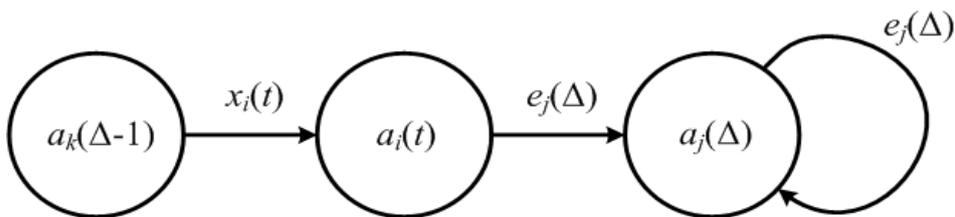
## 1.4. Reconfigurable multifunction machines

Multifunctional automata on automatic memory circuits have the ability to perform a mutation of the excitation functions and outputs with the help of the tuning  $U$ , as was done in the implementation of the automata with memory on the triggers [12], and also with the help of the  $g$ -step automaton  $A$  having  $(g-1)$  An  $A_M$  strategy automaton that generates  $e(\Delta)$ -containing input signals for the abstract  $M$ -automaton  $A$  and performs the reconstruction (regeneration) of the functions for storing the states of the multifunctional memory of the automaton  $A_y$ , which is new in hierarchical theory Multifunctional automata.

The considered multifunctional memory circuits have the ability to implement determinate single-valued transitions (Figure 1.6) with one variable  $x(t)$  and large transitions in two variables (Figure 1.7).

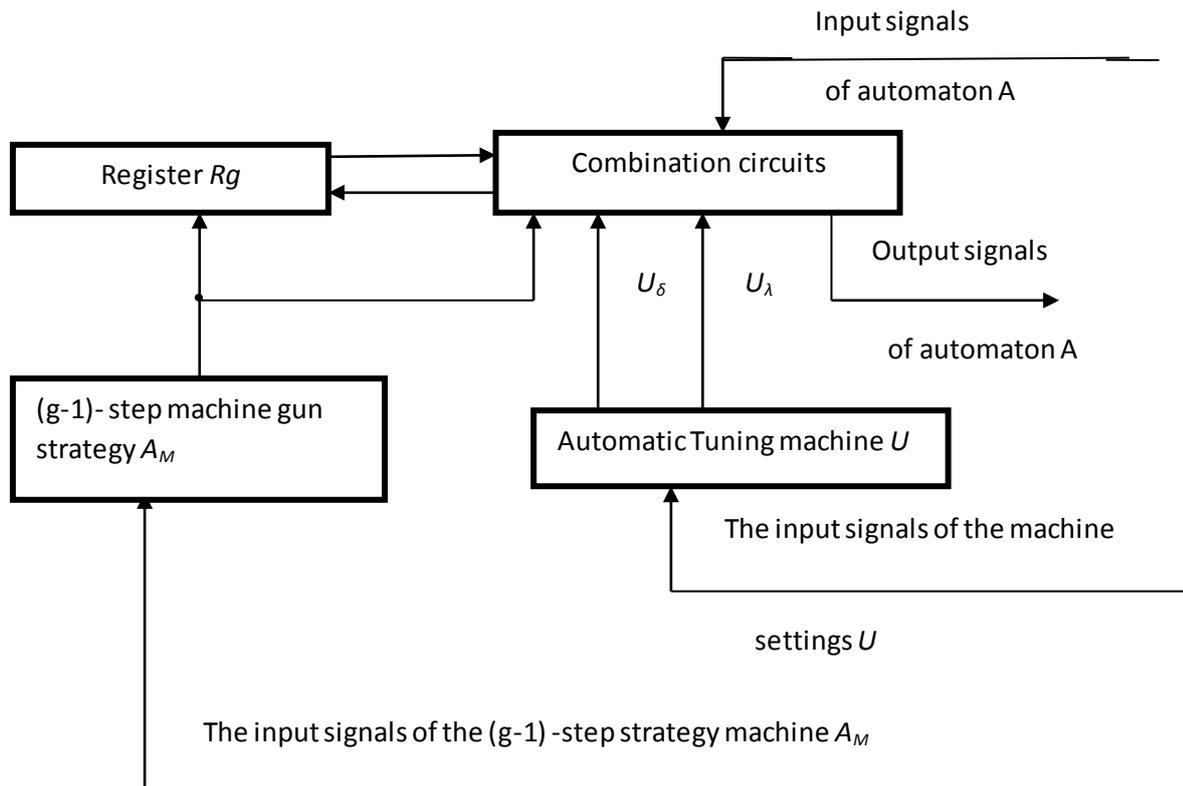


**Fig. 1.6.** A single-valued transition



**Fig. 1.7.** Enlarged transition

A generalized block diagram of a multifunctional automaton  $A$  on automatic memory circuits that make up the memory register  $R_g$ , combinational circuits (CS), tuning automata  $U$  and  $(g-1)$ -step automatic strategy  $A_M$ , is illustrated in Fig. 1.8.



**Fig. 1.8.** A generalized block diagram of a reconfigurable multifunctional automaton A on automatic memory circuits

## 1.5. Structures of multi-level automatic memory devices

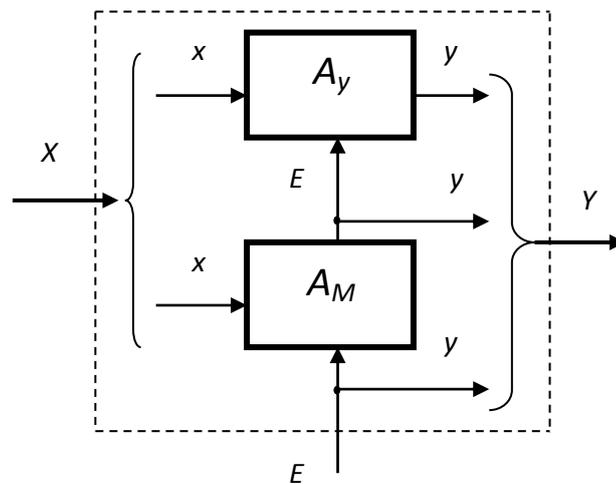
Consider open and semi-closed structures of memory devices.

*Definition 1.8.* An open multi-level structure of the automatic memory device (MUSP) with a multifunctional organization system is called an automatic device, which consists of two devices: a manageable multifunctional memory circuit (MFIS)  $A_y$  and a multifunction machine strategy  $A_M$ , which have combined states, input  $X$  and output  $Y$  alphabets and preserving the input alphabet  $E_H$  of the  $A_M$  strategy automaton.

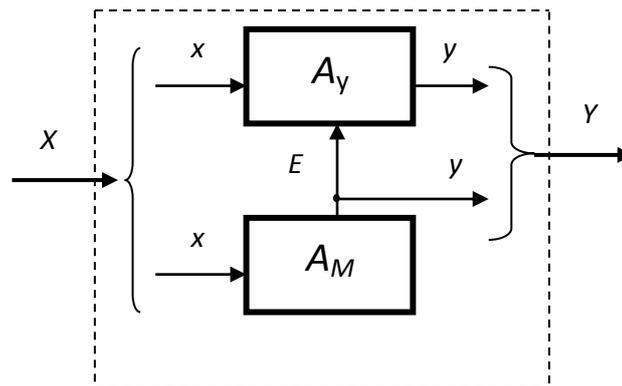
An open multilevel structure of the memory device (MUSP) with a multifunctional organization is illustrated in Fig. 1.9. This structure allows you to expand the multifunctionality, using an additional automatic strategy.

*Definition 1.9.* A semi closed multilevel structure of a memory device (MUSP) with a multifunctional system of organization is called an automaton, which consists of two devices: a managed SMP  $A_y$  and a multifunctional (or monofunctional)  $A_M$  strategy machine that have combined state sets, input  $X$  and output  $Y$  alphabets.

A semi-closed multi-level memory device structure (MUSP) with a multifunctional organization system is illustrated in Fig. 1.10. Triggers and registers on triggers are closed memory structures, due to the fact that they are not able to change the structure of stored states.

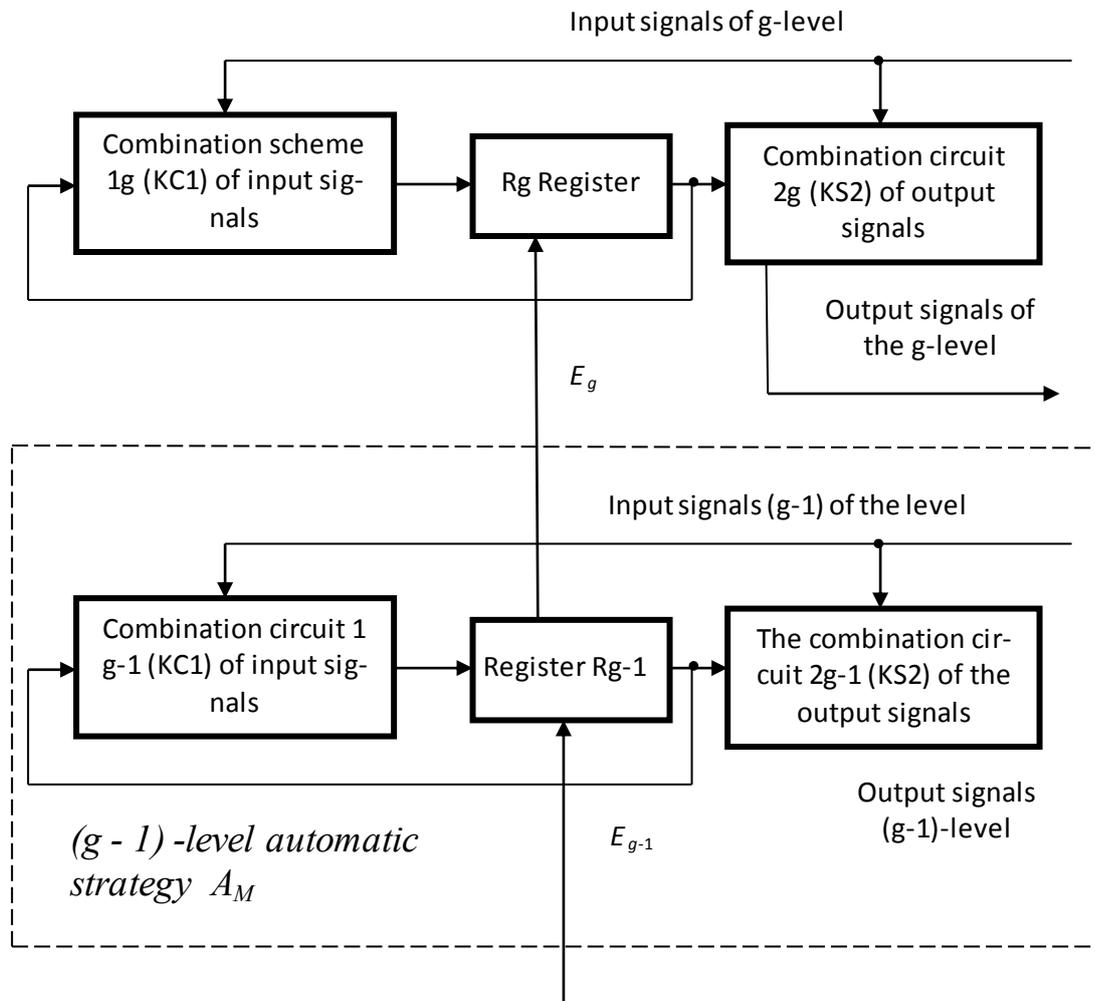


**Fig. 1.9.** Open multilevel structure



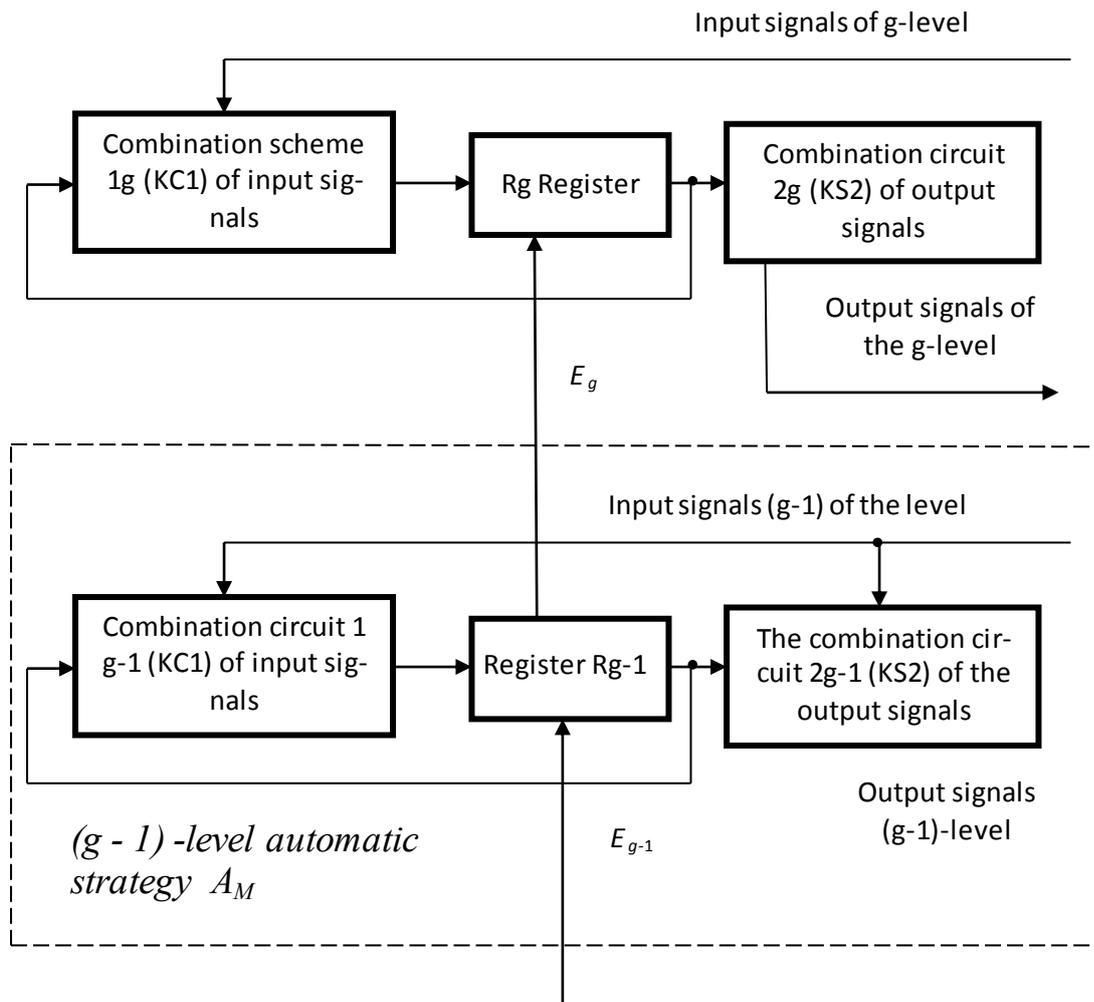
**Fig. 1.10.** Semi-closed multilevel structure

The structural diagram of an automaton of the first kind (an open g-level structure)The MFIS, together with the strategy automaton  $A_M$ , can create semi-open or open multilevel structures of memory devices with a multifunctional organization system.



**Fig. 1.11.** The structural diagram of an automaton of the first kind (an open g-level structure)

A hierarchical abstract automaton  $A$  with memory on the MFIS is considered as a  $g$ -step ( $g > 1$ ) device consisting of registers  $R_j$  ( $j = 1, 2, \dots, g$ ) that interact simultaneously, and two combinational circuits  $KS_{1i}, KS_{2i}$  ( $i = 1, 2, \dots, g$ ) at each step. At level  $g$  (upper), the hierarchical  $g$ -step automaton  $A$  is able to function as a multifunctional automaton of the 1st, 2nd, and also the third kind, which has a  $(g-1)$ -step  $A_M$  strategy



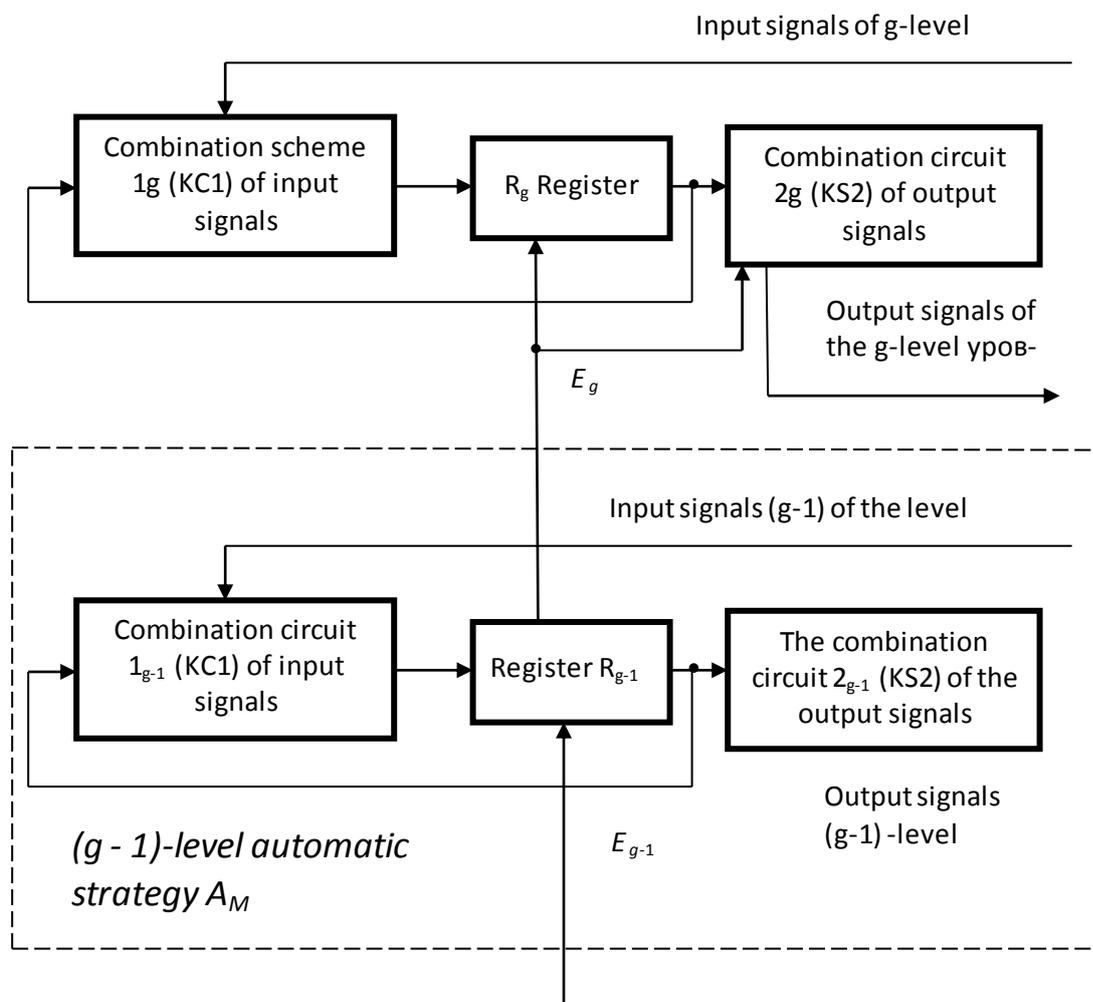
**Fig. 1.12.** The structural diagram of an automaton of the second kind (an open  $g$ -level structure)

machine, used to store a specific block  $\pi_i$  states of the corresponding automaton at level  $g$ . The canonical structure of the  $g$ -step abstract automaton of the 1st, 2nd, and 3rd kind on the MFIS simultaneously with the  $(g-1)$ -step strategy automaton is shown in Fig. 1.11 -1.13.

Unlike the Mealy (1st kind) and Moore (2nd kind) automata with memory on the triggers, whose operation is considered in automatic discrete time, Marakhovsky

automatic machines on the MFIS, whose functioning is considered in automatic continuous time [17; 21] are able to memorize the states of certain blocks (subsets)  $\pi_j(\pi_j \in Q)$  for various input signals  $e_j(\Delta)$ . In accordance with this ability, it is possible to realize various mono-functional Mealy and Moore automata in each of the blocks  $\pi_j(\pi_j \in Q)$  and also using different blocks  $\pi_j(\pi_j \in Q)$  of the  $g$ -level MFP memory and realize multifunctional automata of the 1st and 2nd Kind (Figure 1.11-1.12).

The  $R_g$  register on the MFIS and the combination schemes (KS) are devices that are used to process the local information of the upper  $g$ -level. Each such device can be described as an abstract automaton that processes input information and stores input signals  $e_j(\Delta)$  belonging to the set  $E_g$  ( $e_j(\Delta) \in E_g$ ), from the  $(g-1)$ -level strategy automaton (Figure 1.11).



**Fig. 1.13.** The structural scheme of the automaton of the third kind (open  $g$ -level structure)

Thus, we come to the conclusion that the Mealy and Moore automata, which use memory on flip-flops, are a special case of  $M$ -automata that implement their memory on the MFIS.

It is shown that, the automata  $M$  of the  $g$ -level stage on the MFIS simultaneously with the  $(g-1)$ -level  $A_M$  strategy automata have a unique ability to process local and general information in one clock cycle of computer time.

It is shown that the automata of the  $M$   $g$ -level stage on the MFIS have qualitatively new transitions from one state to another under the influence of  $e_j(\Delta)$  of the input signals compared to the automata in triggers: enlarged, probabilistic and fuzzy.

## **1.6. Abstract probabilistic automata of the third kind**

Differences between discrete automata that describe the functioning of computer devices and even the computer itself, and models that the human brain uses, is that ability that allows a person to think and make conclusions in inaccurate, non-quantitative, fuzzy, vague terms, taking into account parallel considerations Both general and concomitant, local character [8-10].

The talented mathematician Frank Plumpton Ramsay proved that complete disorder is impossible. Each sufficiently large set of numbers, points or objects necessarily contains a highly ordered structure [22].

The main element of the human brain is the neuron, the most important role is played by the restructuring of supramolecular structures in various parts of the nerve cell, as well as structural adjustment waves that accompany the transfer of information within the given neuron and from the neuron to another neuron [23]. Neural memory is represented in a matrix form, driven by two coordinates by exciting and inhibiting input signals and capable of performing structural rearrangement of information transmission in hierarchical ensembles of neurons [24-25].

The considered multifunctional automata of the third kind are capable of transitioning to a new state in two coordinates (variables): informational

(establishing) and preserving input signals and performing structural rearrangement of subsets of memorized states, which can be compared with the work of a neuron [24-25]. In connection with this, it is of interest to study the functioning of multifunctional automata of the third kind when they receive probabilistic or fuzzy input words.

Probabilistic transitions in automata of the third kind occur under the influence of one of two types of probabilistic elementary  $p_{\epsilon 1}(T)$  or  $p_{\epsilon 2}(T)$  input words of the automaton of continuous time (1.11) or (1.12):

$$p_{\epsilon 1}(T) = x_p(t), e_j(\Delta), \quad (1.11)$$

where  $x_p(t)$  – is the information input signal ( $x_p \in X$ ), which unambiguously establishes the memory state of the automaton, which is not stored for any one storing the  $e_j(\Delta)$  input signal;

and

$$p_{\epsilon 2}(T) = x_i(t), e_j^*(\Delta), \quad (1.12)$$

where  $e_j^*(\Delta)$  – is the probabilistic conserved input signal ( $e_j^* \in E$ ).

The information  $x_p(t)$  signal ( $x_p \in X$ ) in the deterministic automata is forbidden, since it sets an output signal on the output nodes of the memory circuits, which is not stored for any one storing the  $e_j(\Delta)$  input signal and leads to probability transitions, which is not allowed (It's simply forbidden).

A probabilistic word of the first type (1.11) makes it possible to carry out a probabilistic transition in the clock internal moment  $\Delta$  from the  $a_p(t)$  state that is not remembered to the state  $a_k(\Delta)$  of the completely determined  $\pi_j$  state block of the automaton under the influence of the input signal preserving  $e_j(\Delta)$

A probabilistic word of the second type (1.12) makes it possible to carry out a probabilistic transition to the clock internal time  $\Delta$  from a definite  $a_i(t)$  state to an

undefined state of a completely defined  $\mu_i$  state block under the influence of a probability preserving  $e_j(\Delta)$  input signal.

Thus, probabilistic transitions in automata of the third kind are able to carry out transitions from one state to a state of a certain state block with a certain measure of probability

We consider two types of probabilistic abstract automata of the third kind, depending on the types of elementary input words (1.11 -1.12).

A probability word of the first type (1.11) determines the operation of a probabilistic abstract automaton of the third kind of the first type.

A probabilistic abstract automaton of the third kind of the first type is given by the following definition.

*Definition 1.10.* A mathematical model of a probabilistic device with a multifunctional memory organization system on automatic memory circuits is the abstract probability automaton  $A_{B1}$  defined as a ten-component corтеge or vector [17]:

$$A_{B1} = (x_p, E, Y_{III}^B, Q, \pi, e_0, a_0, a_p, \delta_o, \delta_y^B, \lambda_3^B, P_e), \quad (1.13)$$

which one:

- $x_p$  - information input signal (input information alphabet);
- $E = \{e_0, e_1, \dots, e_{R-1}\}$  – is the set of conserved input signals (input preserving the alphabet
- $Y_{III}^{B1} = \{Y_0^{B1}, Y_1^{B1}, \dots, Y_{K_3-1}^{B1}\}$  – A set of probabilistic output signals of type 3 (output probabilistic alphabet of type 3);
- $Q = \{a_0, a_1, \dots, a_{m1}\}$  – An arbitrary set of memorized states of the automaton (state alphabet);
- $\pi = \{\pi_0, \pi_1, \dots, \pi_{R-1}\}$  – Set of blocks of  $\pi_j$  states (alphabet of state blocks);
- $e_0 \in E$  – Initial preserving the input signal;
- $a_0 \in \pi_0 (\bigcup_j \pi_j = Q)$  – Initial preserving the input signal;

- $\delta_o: Q \times x_p \rightarrow a_p$  – A single-valued transition function realizing the map  $D_{\delta_o} \subseteq Q \times x_p$  in  $a_p$ . In other words, the function  $\delta_o$ : to some pairs "state - information input signal"  $(a(\Delta-1), x_p(t))$  uniquely corresponds to the non-conserved state of the automaton  $a_p(t) = \delta_o(a(\Delta-1), x_p(t))$ , where  $a_p \notin Q$ ;
- $\delta_y^{B1}: a_p \times E \rightarrow \pi_j$  – Function of the probabilistic large-scale transition, realizing the map  $D_{\delta_y} \subseteq a_p \times e_j$  in  $\pi_j$ . In other words, the function for some pairs "non-conserved state - preserving the input signal"  $(a_p(t), e(\Delta))$  associates the state of the automaton  $a(\Delta) = \delta_y^{B1}(a_p(t), e(\Delta))$ , where  $a_p(t) \notin \pi_j$ ,  $a(\Delta) \in \pi_j$ ,  $a_p(t) \neq a(\Delta)$ ;
- $\lambda_3^{B1}: Q \times E \rightarrow Y_{III}^{B1} \rightarrow$  – Probability function of outputs of type 3 realizing the mapping in  $D_{\lambda_3} \subseteq a_p \times E \rightarrow Y_{III}^{B1}$ . In other words, the function  $\lambda_3^{B1}$ : for some pairs, the "non-conserved state - preserving the input signal"  $(a_p(t), e(\Delta))$  matches the output signal of the automaton  $y^{\delta 1}(\Delta) = \lambda_3^{B1}(a_p(t), e(\Delta))$ , where  $y \in Y_{III}^{B1}$ ;
- $P_e(a_p e_j)$  – System of conditional probabilities.

The system of conditional probabilities  $P_e$ , given for each pair  $(a_p(t), e_j(\Delta))$  on the set  $\pi_j \times Y_{III}^{B1}$  ( $\pi_j \in Q$ ;  $Y_{III}^{B1} \in Y_{III}$ , where  $Y_{III}$  is the output signal of the automaton of the third kind) characterizes the probability of the automaton transition to the state  $a_s(\Delta)$  with the issuance the signal  $y_k(\Delta)$ , which displays the state  $a_s(\Delta)$  of the automaton, provided that the automaton was previously set to the state  $a_p(t)$ , which is not remembered, and its inputs are fed with a saving  $e_j(\Delta)$  input signal capable of storing The entire set of states of the state block  $\pi_j$  ( $a_s \in \pi_j$ ).

The law of operation of a probabilistic abstract automaton of the third kind of the first type is given by equations (1.14):

$$\begin{cases} a_p(t) = \delta_0(a(\Delta - 1), x_p(t)); \\ a(\Delta) = \delta_{B_1}(a_p(t), e(\Delta), P_e); \\ y_L^{B_1}(\Delta) = \lambda_3(a(\Delta), e(\Delta)), \\ a(t) \notin \pi_j, a(\Delta) \in \pi_j; \\ i = 0, 1, 2, \dots; \Delta = 0, 1, 2, \dots, 0 < P_e \leq 1. \end{cases} \quad (1.14)$$

A probabilistic word of the second type (1.12) defines a probabilistic abstract automaton of the third kind of the second kind.

A probabilistic abstract automaton of the third kind of the second type is given by the following definition.

*Definition 1.11.* A mathematical model of a probabilistic device with a multifunctional system of memory organization on automatic memory circuits is the abstract probability automaton  $A_{B2}$  defined as an eleven component cortege or vector [17]:

$$A_{B2} = (X, E_{\epsilon 2}, Y_{III}^{B2}, Q, \pi, e_0, a_0, \delta_0, \delta_y^{B2}, \lambda_3^{B2}, P^{(0)}, P_e), \quad (1.15)$$

which one:

- $X = \{x_1, x_2, \dots, x_{N-1}\}$  – A set of information input signals (input information alphabet);
- $E_{\epsilon 2} = \{e_1^{B2}, e_2^{B2}, \dots, e_{R-1}^{B2}\}$  – A set of probabilistic conserved input signals (input probabilistic preserving alphabet);
- $Y_{III}^{B2} = \{Y_0^{B2}, Y_1^{B2}, \dots, Y_{K_3-1}^{B2}\}$  – Set of probabilistic output signals of type 2 (output probabilistic alphabet of type 2);
- $Q = \{a_0, a_1, \dots, a_{m1}\}$  – An arbitrary set of memorized states (the alphabet of states);
- $\pi = \{\pi_0, \pi_1, \dots, \pi_{R-1}\}$  – Set of blocks of  $\pi_j$  states (alphabet of state blocks);
- $e_0 \in E$  – Initial preserving the input signal;
- $a_0 \in \pi_0 (\bigcup_j \pi_j = Q)$  – Initial state of the machine

- $\delta_o: Q \times X \rightarrow Q$  – A single-valued transition function realizing the mapping in  $D_{\delta_o} \subseteq Q \times X \rightarrow B(a(\Delta-1), x(t))$ . In other words, the function  $\delta_o$ : to some pairs "state - information input signal"  $(a(\Delta-1), x(t))$  associates the state of the automaton  $a(t) = \delta_o(a(\Delta-1), x(t))$ , where  $a \in Q$ ;
- $\delta_y^{B2}: Q \times E_{B2} \rightarrow \pi_j$  – Function of the probabilistic large-scale transition, realizing the map  $D_{\delta_y} \subseteq Q \times E_{B2} \rightarrow \pi_j$ . In other words, the function for some pairs "state-probability-preserving input signal"  $(a(t), e^{B2}(\Delta))$  associates the state of the automaton  $a(\Delta) = \delta_y^{B2}(a(t), e^{B2}(\Delta))$ , where  $a(t) \in \pi_b$ ,  $a(\Delta) \in \pi_j$ ,  $a(t) \neq a(\Delta)$  ( $a(t), a(\Delta) \in Q$ );
- $\lambda_3^{B2}: Q \times E_{B2} \rightarrow Y_{III}^{B2}$  – Probability function of outputs of type 3 realizing the mapping  $D_{\lambda_3} \subseteq Q \times E_{B2} \rightarrow Y_{III}^{B2}$ . In other words, the function  $\delta_y^{B2}$  to some pairs "state-probability-preserving input signal"  $(a(t), e^{B2}(\Delta))$  matches the output signal of the automaton  $y^{B2}(\Delta) = \lambda_3^{B2}(a(t), e^{B2}(\Delta))$ , where  $y \in Y_{III}^{B2}$ ;
- $P^{(0)} = \{P_0, P_1, P_2, \dots, P_{k-1}\}$  – The initial probability distribution of the state block  $P_e(a e^{B2}_j)$  of the automaton;
- $P_e(a e^{B2}_j)$  – System of conditional probabilities.

The system of conditional probabilities  $P_e$ , given for each pair  $(a(t), e^{B2}_j(\Delta))$  on the set  $\pi_j \times Y_{III}^{B2}$  ( $\pi_j \in Q$ ;  $Y_{III}^{B2} \in Y_{III}$ , where  $Y_{III}$  is the output signal of the automaton of the third kind) characterizes the transition probability of the automaton to the state  $a_s(\Delta)$  with a signal  $y^{B2}(\Delta)$ , which displays the state  $a_s(\Delta)$  of the automaton, provided that the automaton was previously set to the state  $a(t)$  and its inputs are fed with a probabilistic saving  $e^{B2}_j(\Delta)$  input signal capable of storing the entire set of states of the block  $\pi_j$  states ( $a_s \in \pi_j$ ).

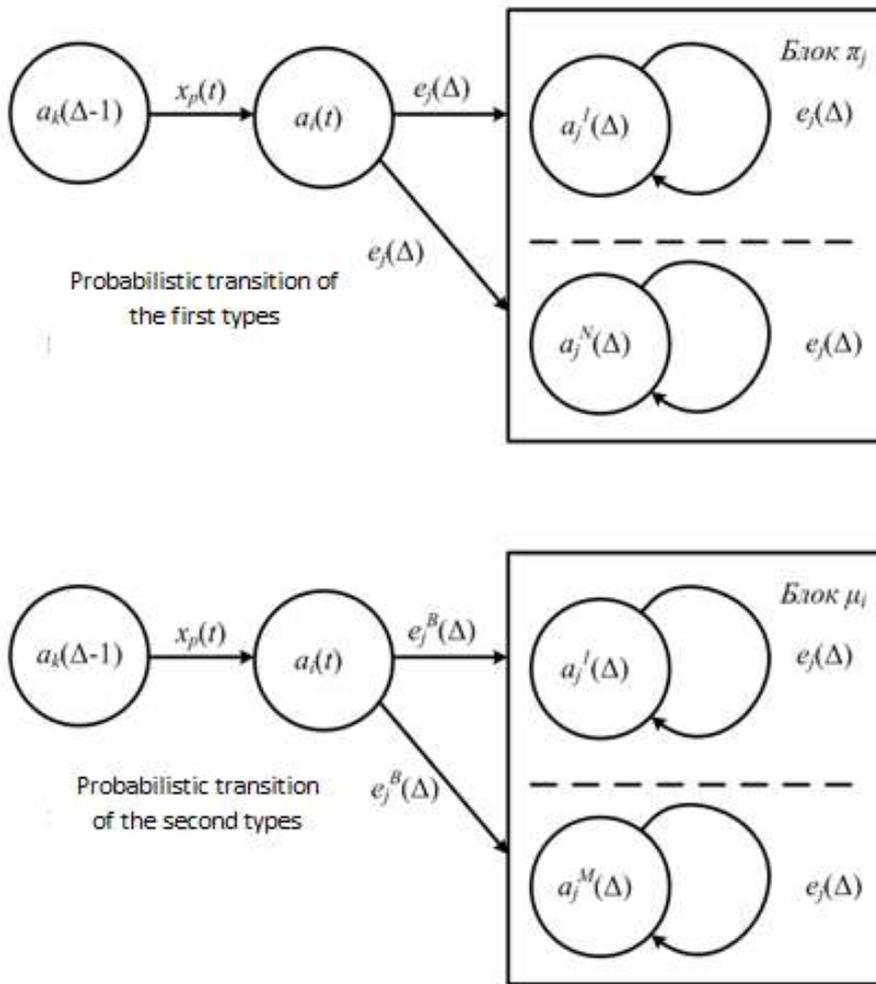
The law of operation of a probabilistic abstract automaton of the third kind of the second type is given by equations (1.16):

$$\begin{cases}
a(t) = \delta_0(a(\Delta - 1), x(t)); \\
a(\Delta) = \delta_{B_2}(a(t), e_j^{B_2}(\Delta), P_0, P_e); \\
y_L^{B_2}(\Delta) = \lambda_3^{e_2}(a(\Delta), e(\Delta)), \\
a(t) \notin \pi_j, a(\Delta) \in \pi_j; \\
i = 0, 1, 2, \dots; \Delta = 0, 1, 2, \dots, 0 < P_0 \leq 1; 0 < P_e \leq 1.
\end{cases} \quad (1.16)$$

The meaning of the notion of an abstract probability automaton of the third kind consists in the realization of a probability mapping  $\psi_e$  in the set of probabilistic words  $p_e(T)$  of input alphabets into the set of words  $y^e(\Delta)$  of the output alphabet whose symbols belong to completely defined subsets  $Y_{III}$ . The appearance of each character of the output sequence depends and is determined by a system of conditional probabilities in accordance with the type of an abstract probabilistic automaton of the third kind.

Abstract probability automata of the third kind extend the possibilities of deterministic automata of the third kind due to probabilistic transitions to certain subsets of states of the automaton.

Intelligence consists in the fact that there is the possibility of eliminating unnecessary options. In this sense, probabilistic transitions provide an opportunity to choose from a set of subsets  $\pi_j$  a completely definite subset and, thus, to shorten the time of search. Such transitions are more general than transitions from one state to another.



**Fig. 1.14.** Probabilistic transitions of the first and second types

### 1.17. Abstract fuzzy automata of the 3rd kind

Abstract fuzzy automata of the third kind are considered when input  $p_H(T)$  input words ( $p_H(T) = x_p(t), e^s(\Delta)$ ) arrive at their input nodes in the automaton continuous time  $T$ .

*Definition 1.12.* A mathematical model of a fuzzy device with a multifunctional memory management system on automatic memory circuits is an abstract fuzzy automaton  $M$  defined as a fourteen-component cortege or vector [17]:

$$A_H = (x_p, E, E_\omega, Y_{III}^H, Q, Q_H, \pi, e_0, a_0, \delta_o, \delta_y^H, \lambda_3^H, P^{(0)}, P_e), \quad (1.17)$$

which one:

- $x_p$  – Information input (input information alphabet);
- $E$  – A set of preserving input signals (input preserving the alphabet);
- $E_H = \{e_1^H, e_2^H, \dots, e_{R-1}^H\}$  – A set of fuzzy conserved input signals (input fuzzy conserved alphabet, where  $(\bigcup_j E_H = E)$ );
- $Y_{III}^H = \{Y_0^H, Y_1^H, \dots, Y_{K_3-1}^H\}$  – Set of fuzzy output signals of the 3rd kind automaton (output fuzzy alphabet);
- $Q$  – An arbitrary set of memorized states of the automaton (state alphabet);
- $Q_H = \{a_0, a_1, \dots, a_{m1}\}$  – An arbitrary fuzzy subset of states (the alphabet of fuzzy subsets of states, where  $(\bigcup_j Q_H = Q)$ );
- $\pi = \{\pi_0, \pi_1, \dots, \pi_{R-1}\}$  – Set of blocks of  $\pi_j$  states (alphabet of state blocks);
- $e_{0 \in E}$  – Initial preserving the input signal;
- $a_0 \in \pi_0 (\bigcup_j \pi_j = Q)$  – Initial state of the machine
- $\delta_o: Q \times x_p \rightarrow a_p$  – A single-valued function of the transition to an unremembered  $a_p$  state, realizing the map  $D_{\delta_o} \subseteq Q \times x_p$  in  $a_p$ . In other words, the function  $\delta_o$ : to some pairs "state - information input"  $(a(\Delta-1), x_p(t))$  puts into correspondence the unserved state of the automaton  $a_p(t) = \delta_o(a(\Delta-1), x_p(t))$ , where  $a_p \notin Q$ ;
- $\delta_y^H: Q \times E_H \rightarrow \pi_j$  – Function of fuzzy integration, realizing the map  $D_{\delta_y} \subseteq Q \times E_H$  in  $\pi_j$ . In other words, the function  $\delta_y^H$  for some pairs of "non-conserved state - fuzzy conserved input signal"  $(a_p(t), e^H(\Delta))$  associates the state of the automaton  $a(\Delta) = \delta_y^H(a_p(t), e^H(\Delta))$ , where  $a_p(t) \notin \pi_j, a(\Delta) \in \pi_j, a_p(t) \neq a(\Delta)$ ;
- $\lambda_3^H: Q_H \times E_H \rightarrow Y_{III}^H$  – Fuzzy output function that implements the mapping  $D_{\lambda_3} \subseteq Q_H \times E_H$  in  $Y_{III}^H$ . In other words, the function  $\lambda_3^H$ : for some pairs, the

"non-conserved state-fuzzy conserved input signal"  $(a_p(t), e^H(\Delta))$  associates the output signal of the automaton  $y^H(\Delta) = \lambda_3^H(a_p(t), e^H(\Delta))$ , where  $y^H \in Y_{III}^H$ ;

- $P^{(0)} = \{P_0, P_1, P_2, \dots, P_{k-1}\}$  – The initial distribution of the fuzzy subset  $Q_H$  of the states of the automaton;
- $P_H(a_p e^H_j)$  – System of conditional probabilities.

Conditional probabilities system  $P_H$ , assigned for each pair  $(a_p(t), e^H(\Delta))$  in the set  $Q_H \times Y_{III}^H$  ( $Q_H \in Q$ ;  $Y_{III}^H \in Y_{III}$ , wherein  $Y_{II}$  output automaton signals 3rd kind represents the transition probability automaton to the state  $a_s(\Delta)$  with  $y_k(\Delta)$ , which displays the state  $a_s(\Delta)$  of the automaton, provided that the automaton was previously set to the state  $a_p(t)$  and its inputs are fed with a fuzzy  $e^H_j(\Delta)$  - containing input signal capable of translating the automaton from  $a_p(t)$  state to  $a(\Delta)$  block status  $\pi_j$  states, which belongs to a fuzzy subset  $Q_H$  states ( $\pi_j \in Q_H$ ).

The law of functioning fuzzy abstract automaton third kind is given by equations (1.18):

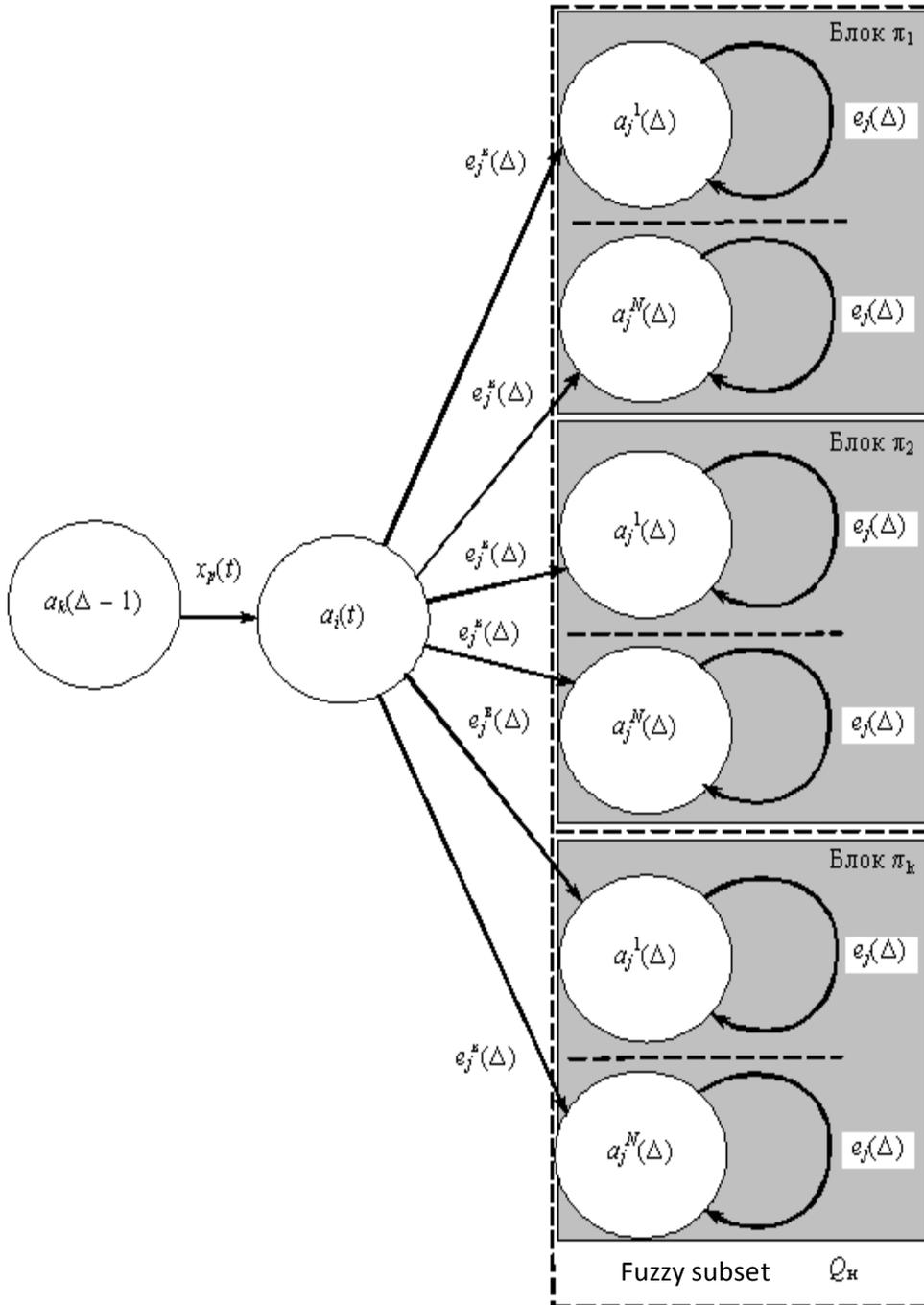
$$\begin{cases} a_p(t) = \delta_0(a(\Delta-1), x_p(t)); \\ a(\Delta) = \delta_n(a_p(t), e^H_j(\Delta), P_0, P_n); \\ y^H(\Delta) = \lambda_3^H(a(\Delta), e^H(\Delta)), \\ a_p(t) \notin \pi_j, a(\Delta) \in \pi_j; \\ i = 0, 1, 2, \dots; \Delta = 0, 1, 2, \dots, 0 < P_0 \leq 1; 0 < P_n \leq 1. \end{cases} \quad (1.18)$$

The meaning of the notion of an abstract fuzzy automaton of the third kind consists in the realization of some fuzzy map  $\psi_H$  of the set of elementary fuzzy words  $p_\delta(T)$  of input alphabets into the set of words  $y^H(\Delta)$  of the output alphabet whose symbols belong to completely defined subsets  $Y_{III}^H$ . The appearance of each character of the output sequence depends and is determined by the system of conditional probabilities in accordance with the system of conditional probabilities  $P_H$  of an abstract fuzzy auto of the third kind.

Fuzzy subsets  $Q_H$  consist of several well-defined subsets  $\pi_j$  ( $\pi_j \in Q_H$ ). The transition to this fuzzy subset  $Q_H$  from a certain state  $a_i(\Delta-1)$  occurs under the

influence of the fuzzy input word  $p_n(T)$ , which probabilistically converts the automaton to one of the states  $a_k(\Delta)$  of one of the subsets  $\pi_j$  of the fuzzy subset  $Q_n$ .

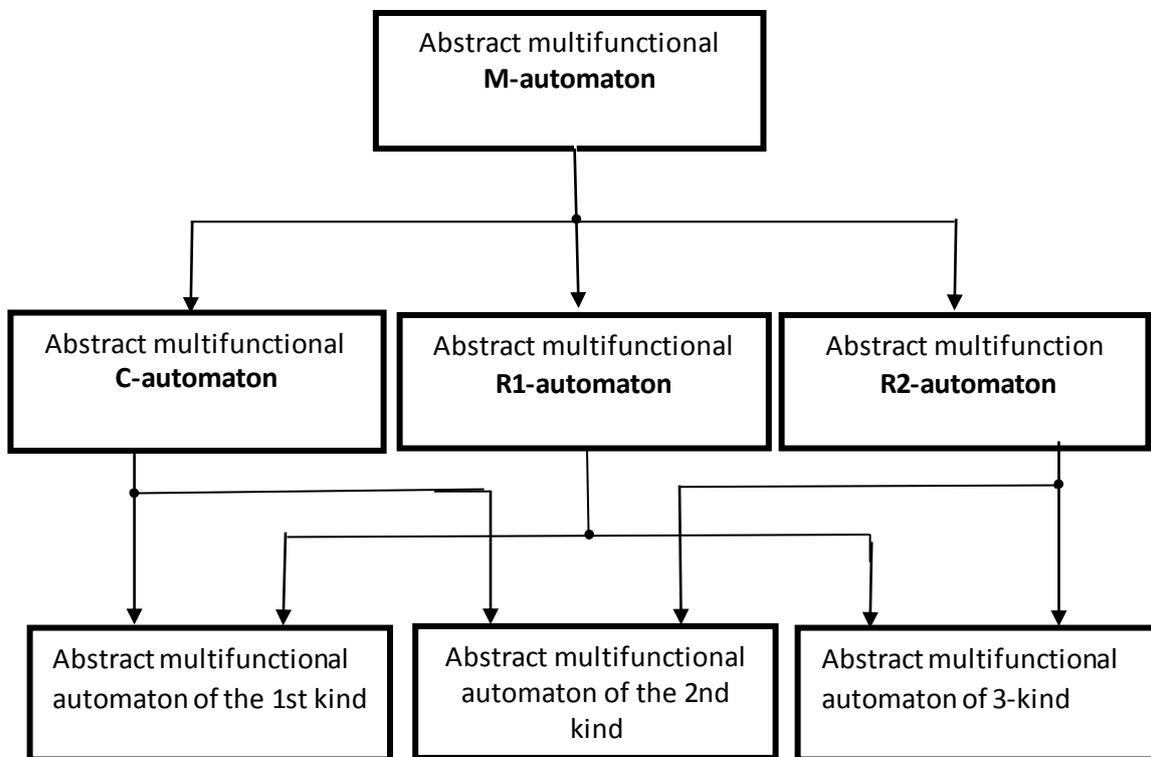
The fuzzy transition is shown in Fig. 1.15.



**Fig. 1.15.** Fuzzy transition in the automaton of the third kind

## 1.8. Classification of deterministic abstract automata

Let us consider the classification of deterministic abstract automata using the types of their output signals, as is done in classical Mealy abstract machines generating output  $y^1(t)$  signals of type 1 and Moore, generating output  $y^2(T)$  signals of type 2. The combined abstract C-automaton is characterized by Generation of two types (type 1 and type 2) of the output signals  $y^1(t)$  and  $y^2(T)$ . Memory circuits of these machines use triggers, which, by their limitations, determine the monofunctional nature of their operation.



**Fig. 1.16.** Classification of abstract multifunctional deterministic automata on automatic memory circuits

The functioning of the Marakhovsky multifunctional automata is investigated in automatic continuous time, and the abstract automata whose memory is realized on automatic memory circuits determine the multifunctional nature of their operation and can also be classified according to their types of output signals, as is done in the classical Mealy and Moore automata. Thus, the combined abstract multifunction  $M$ -

automaton has the ability to generate three types (output 1, type 2 and type 3) of the output signals  $y^1(t)$ ,  $y^2(T)$  and  $y^3(\Delta)$ . The combination of the output signals  $y^1(t)$  and  $y^2(T)$  in the abstract multifunctional automaton of only two types (type 1 and type 2) can also be called a multifunctional *C*-automaton, by analogy with the combined Mealy and Moore automaton. The combination of only two types (type 1 and type 3) of output signals  $y^1(t)$  and  $y^3(\Delta)$  in the abstract multifunctional automaton is called a multifunctional *R1*-automaton, and the combination in the abstract multifunction machine of only two types (type 2 and type 3) of output signals we call  $y^2(T)$  and  $y^3(\Delta)$  a multifunctional *R2*-automaton. Abstract multifunctional automata using only one type of output signals  $y^1(t)$ ,  $y^2(T)$  or  $y^3(\Delta)$  will be called respectively abstract multifunctional automata of the 1st, 2nd or 3rd kind.

We note that abstract monofunctional automata Mealy (of the first kind) and Moore (of the second kind) are a special case of abstract multifunctional automata of the first and second kind.

Such a classification of automata is rather conditional, since the type of transition (deterministic, probabilistic, fuzzy), type of memory used in the machine (one-level, multi-level, hierarchical), etc., can be taken as a classification criterion.

## **1.9. Classification of nondeterministic abstract machines**

The considered non-deterministic (probabilistic and fuzzy) abstract automata of the third kind are characterized by the implementation of transitions in two variables in one cycle  $a(\Delta-1)$  of the automatic continuous time under the influence of elementary probabilistic and fuzzy input words.

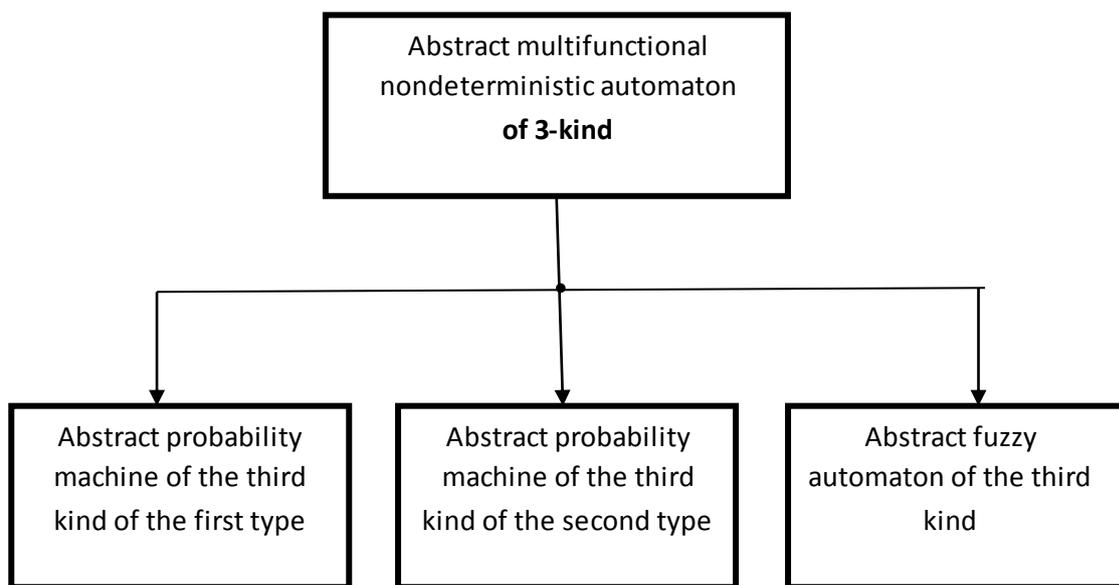
Probabilistic abstract automata of the third kind describe probabilistic transitions from a completely defined  $a(\Delta-1)$  state to the state  $a(\Delta)$  of an entire subset of states of a completely defined block of  $\pi_j$  states in accordance with a definite probability  $P_e$ . Such transitions can be constructive if one considers that they are performed in the apparatus of computer devices and can be determined by testing the equipment. Such transitions can be very useful when using them in program algorithms due to the fact

that eliminating unnecessary options increases the "machine" intelligence of the program.

Fuzzy abstract automata of the third kind describe fuzzy transitions from a completely defined  $a(\Delta-1)$  state to the state  $a(\Delta)$  of an entire fuzzy  $Q_h$  subset of states, which consists of completely defined subsets of  $\pi_j$  states in accordance with a certain probability  $P_h$ .

Such transitions can also be constructive if one considers that they are performed in the apparatus of computer devices and can be determined by testing the equipment. Such transitions can also be very useful when using them in program algorithms due to the fact that eliminating unnecessary options increases the "machine" intelligence of the program.

The classification of abstract probabilistic and fuzzy automata of the third kind is shown in Fig. 1.17



**Fig. 1.17.** Classification of abstract nondeterministic automata of the third kind

An abstract nondeterministic automaton of the third kind can be general when it uses elementary probabilistic input words of the first and second type, as well as elementary fuzzy input words.

The combined abstract nondeterministic automaton can be considered together with a deterministic abstract automaton of the third kind. In this case, the

corresponding elementary input words are used: single-valued, enlarged, probabilistic first and second types or fuzzy.

### **1.10. Conclusion to the 1st chapter**

The theory of multifunctional automata of Marakhovsky is introduced in the development of experimental digital devices using memory on automatic memory circuits. Perhaps this theory will find its worthy application in the development of reconfigurable promising and competitive digital devices and artificial intelligence devices.

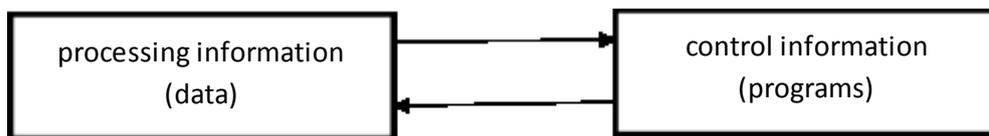
## Chapter 2

### MULTILEVEL ABSTRACT AUTOMATICS

#### 2.1. The principle of hierarchical program management

The principle of program management assumes the use of numerical methods for computing arithmetic and logical operations performed in a sequence determined by the algorithm of the problems being solved.

For the first time, the principle of program management was proposed by the English scientist Charles Babbage in the construction of an analytical machine in the 30s of the 19th century. It consisted in the fact that information was divided into two types: processing (data) and control (program).



**Fig. 2.1.** The principle of program management

In 1945, the American scientist J. von Neumann supplemented it with the principle of a program stored in memory, thus completing the theoretical justification for the possibility of creating universal technical means for automating computational work [1].

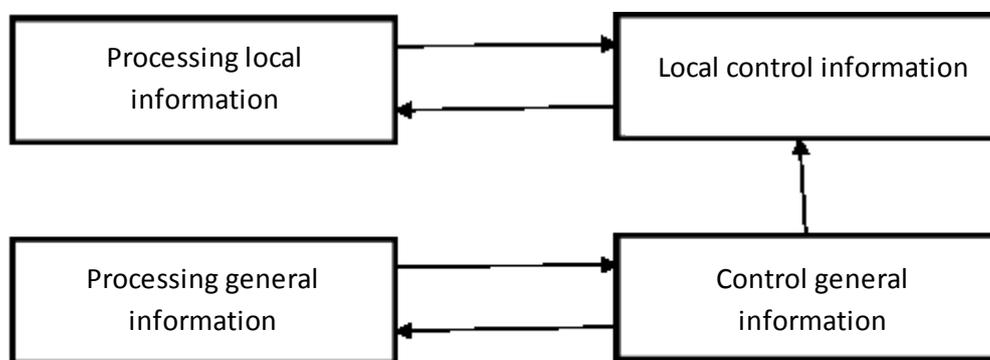
It is important to note that the calculations made by the computer's processor are determined by the program's commands. It is the program commands that "configure" the computer to obtain the necessary results. Replacement of commands in the program can lead to changes in the functions of the computer. Consequently, the variety of subsets of commands and programs executed in the computer determines a class of functions that are capable of effectively implementing algorithms for a particular class of tasks.

At the present time, the capabilities of computers are growing [2]. They strive to find new capabilities of the element base for integrated circuits based on multi-

functional memory circuits and analogs of a biological neuron [3-5]. They try to move away from the Neumann architecture for the purpose of compatibility of operations (pipeline processing of information), parallelization of algorithms (multiprocessor computers), development of interpretation systems (introduction of non-traditional means of addressing and operations over information), etc. [4]. One of the characteristic fundamental properties of these modern studies in new directions of their development is the preservation of automatic discrete time, which limits the possibility of the appearance of new transitions in elementary memory.

One of the proposals for expanding the possibilities for a new basis for information processing was the principle of hierarchical program management, proposed by Prof. Marakhovsky L.F. in 1996 [6].

The principle of hierarchical program management is that the information processed and managed is divided into a private and a general, where the management information is vertically linked from the general information to the private information and processed simultaneously with it (in parallel).



1 **Fig. 2.2.** The principle of hierarchical program management

One of the main temporal characteristics of processing hierarchical information in this case is faster processing of private information in relation to the general, and one of the functional characteristics is the change in the algorithm for processing the multifunctional private information, depending on the simultaneous change in the processing of general information.

General information can also be presented as local and general, increasing the number of levels of processing hierarchical information. Such a hierarchical separation of information is finite and possible up to a certain minimum amount of general information.

Examples of hierarchical information are often found in our lives. For example, the structure of subordination in the army: from the commander to the soldier, who should execute the commands of higher-ranking officers.

Local information can be processed unambiguously, superficially (probability) or fuzzy under the principle of hierarchical program management. General ("root") information should be processed unambiguously and determine the mode of processing of local information.

In a two-level processing, local information can be processed in deterministic or probabilistic ways, and for three-level processing and higher local information can be processed in deterministic, probabilistic or fuzzy ways.

The use of probabilistic and fuzzy computational results along with the emerging possibility of deterministic multifunctional computing processing results extends the capabilities of computing devices and creates prerequisites for increasing computer intelligence [7-9]. To process the hierarchical memory, we had to abandon the limitations of the automatic discrete time and proceed to automatic continuous time, which allowed us to expand the range of transitions in multi-level automata.

## **2.2. The meaning of probabilistic and fuzzy transitions in real devices of computer technology**

Advances in the development of the theory of algorithms and the theory of deterministic automata have placed on the agenda the task of identifying new fundamental opportunities that involve the use of random acts in the process of discrete processing of information. The meaning of the theory of probability automata is to obtain a positive randomness value in discrete information converters. A probabilistic automaton essentially appears only when the probabilities are not postulated com-

putable. A probabilistic automaton is defined as a non-constructive element because of the assumption that computability of probability transitions is not computable, except in the special case when a rational probability automaton is considered [10].

Another modification of the finite automaton is considered: fuzzy automata. The theory of fuzzy subsets best allows you to structure everything that is divided by not very precise boundaries, for example, thought, language, people's perceptions. Merit of Professor L.A. Zadeh is introducing the notion of weighted membership, which gives the concept of a fuzzy (fuzzy) subset [7-8].

Fuzzy automata are obtained as a result of replacing transition functions and outputs with fuzzy relationships. A fuzzy subset  $M$  is given by a function mapping in the interval  $[0, 1]$ . Thus, the functions of the transition function and the output in the fuzzy automaton are functions that map the sets  $Q \times X \times Q$  and  $Q \times X \times Y$  in the interval  $[0, 1]$ , where  $Q$  is the set of states,  $X$  is the input alphabet,  $Y$  is the output alphabet. The non-fuzzy automata naturally generalize the basic concepts and problems characteristic of fuzzy automata [6-8; 11].

Probabilistic and fuzzy automata in this handbook will not be considered in detail. This is due to the fact that they are usually not used in practice in the form of structural elements.

However, in the opinion of the authors this is not quite a true statement.

Mathematics offers a much simpler explanation. In 1928, Frank Plumpton Ramsey, an English mathematician, philosopher and economist, has shown that such ordered configurations are inevitably present in any large structure, be it a group of stars, a collection of randomly scattered pebbles, or a sequence of numbers obtained by throwing a dice. If we are talking about a sufficiently large number of stars, we can always find a group that with very high accuracy forms a given configuration: a straight line, a rectangle or, if we are talking about the stars, a large bucket. In fact, Ramsey's theory asserts that any structure necessarily contains an ordered substructure. As the American mathematician Theodor S. Motzkin first proclaimed about a quarter of a century ago, Ramsey's theory implies that complete confusion is impossible [12].

Multifunctional automata in automatic memory circuits have the ability to perform a mutation of the excitation functions and outputs using the tuning  $U$ , as was done in the implementation of the automata with memory on the triggers. This can also be done with the help of a  $g$ -step multifunctional automaton  $A_y$  having a  $(g-1)$ -step strategy  $A_M$  machine that generates  $e$ -preserving input signals for the automaton  $A_u$  and performs the reconstruction (regeneration) of the multifunction memory functions of the automaton  $A_y$ , which is new in the theory of hierarchical automata [6].

The meaning of probabilistic and fuzzy transitions in real devices of computer technology from the mathematical point of view is equally probable.

Based on their Ramsey theory and practical observations in real computer devices, probabilistic and fuzzy transitions largely depend on electrical parameters in memory circuits and with a certain degree of probability are realized in the concrete state of a certain subset of states. In this connection, transitions can be used in the computational practice of real devices with a certain degree of probability.

### **2.3. Multilevel Abstract Automata**

Sequential multifunctional automata of Marakhovsky 1 st, 2 nd and 3 rd kind, having two sets of input alphabets:  $X$  - information input alphabet and  $E$  - preserving the input alphabet, can be tuned by another multifunction machine through the set of letters  $e_j$  ( $e_j \in E$ ) of the conserved input the alphabet to function in a certain block (subset) of its states. In this case, a multilevel (hierarchical) structure of the  $F$ -automaton from the successive multifunctional  $S_i$  subautomata ( $i=1, 2, \dots, n$ ) is formed. The organization of such a combined structure of the hierarchical automaton (IA) is shown in Fig. 2.3.

Each multifunctional  $S_i$  ( $i=1, 2, \dots, n$ ) multifunctional abstract  $F$ -automaton can pass from one state to another in parallel with other subautomata of IA (Figure 2.3). The functioning of the  $S_i$  subautomata in a particular subset of states can be changed by the effects of the input letters of the information alphabet  $X$  at the clock moment  $t$  and the effects of the results of the operation of other subautomata  $S_j$  IA by the letters

of the preserving input alphabet  $E$  at the moments of the internal cycle  $\Delta$  of the automatic continuous time  $T$ .

Let us define a multi-level (hierarchical) abstract  $F$ -automaton.

*Definition.* The mathematical model of a hierarchical discrete device with a multifunctional system of memory organization is an abstract hierarchical  $F$ -automaton, defined as an  $N$ -component vector

$$F_u = (S_1, S_2, \dots, S_N), \quad (2.1)$$

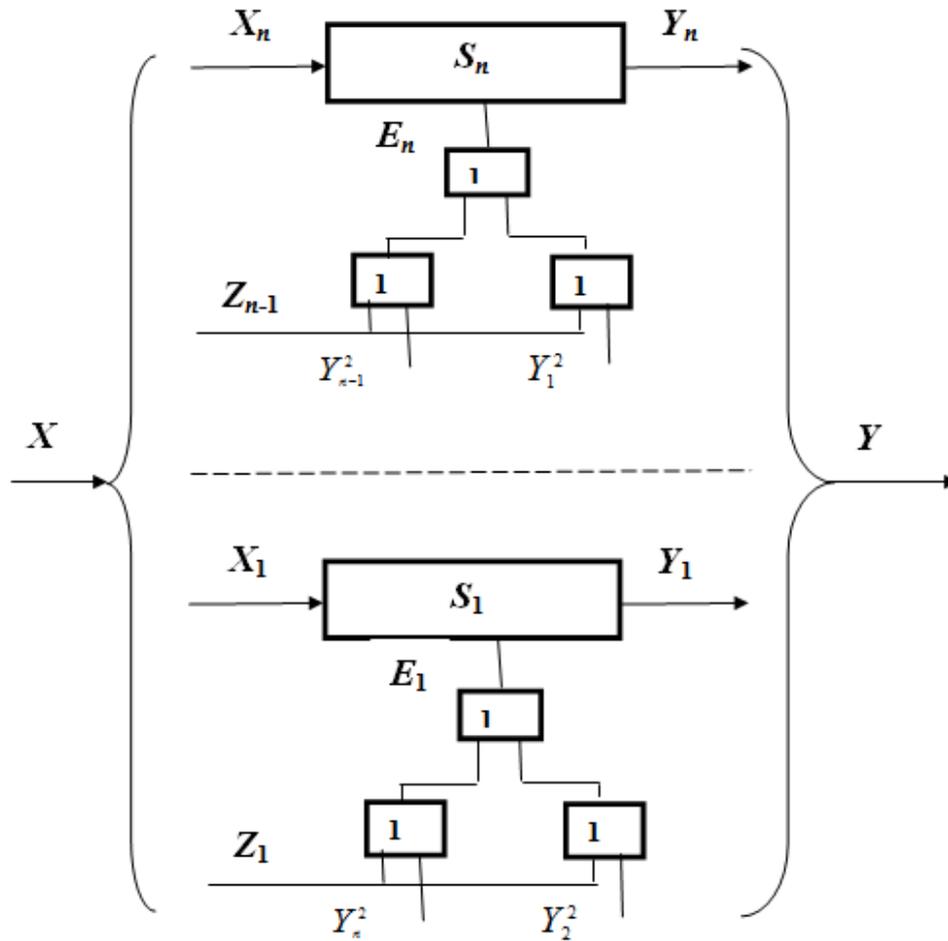
whose  $S_i$  components are given by the sixteenth component vector

$$S_i = (X_i, E_i, Z_i, Y_i^1, Y_i^2, Y_i^3, Q_i, \pi_i, e_{i_0}, a_{i_0}, \delta_{i_0}, \delta_{i_e}, \delta_{i_y}, \lambda_{i_1}, \lambda_{i_2}, \lambda_{i_3}) \quad (2.2)$$

which one

- $X_i$  - a set of information input signals;
- $E_i$  - the set of input signals that they preserve states;
- $Z_i$  - the set of resolving input signals;
- $Y_i^1$  - a set of output signals of type 1;
- $Y_i^2$  - a set of output signals of type 2;
- $Y_i^3$  - a set of output signals of type 3;
- $Q_i$  is an arbitrary set of states;
- $\pi_i$  - the set of blocks  $\pi_{i_j}$  of states of the subautomata  $S_i$ ;
- $e_{i_0}$  - initial storing the input signal;
- $a_{i_0}$  is the initial state of the  $S_i$  subautomaton;
- $\delta_{i_0} : Q_i \times X_i \rightarrow Q_i$  is a single-valued transition function;
- $\delta_{i_e} : Q_i \times e_{i_j} \rightarrow \pi_{i_j}$  - state blocks saving function;
- $\delta_{i_y} : Q_i \times E_i \rightarrow \pi_{i_j}$  - the function of the enlarged transition;

- $\lambda_{i_1} : Q_i \times X_i \rightarrow Y_{i_1}'$  - function of outputs type 1;
- $\lambda_{i_2} : \pi_{i_j} \rightarrow Y_{i_2}''$  -- function of outputs type 2;
- $\lambda_{i_3} : Q_i \times E_i \rightarrow Y_{i_3}'''$  is a function of outputs of type 3 and defined functionally, as well as components of the structure of  $S_i$ , by a hexadecomponent vector



**Fig. 2.3.** Hierarchical abstract  $F_i$ -automaton

$$F_A = (X, E, Z, Y', Y'', Y''', Q, \pi, E_0, Q_0, F_1, F_2, F_3, \lambda_1, \lambda_2, \lambda_3) \quad (2.3)$$

which one

- $X = \{X_1, X_2, \dots, X_N\}$  - the set of information input signals;
- $E = \{E_1, E_2, \dots, E_N\}$  - is the set of input signals that conserve;
- $Z_i = \{Z_1, Z_2, \dots, Z_N\}$  - the set of resolving input signals;
- $Y' = \{Y_1', Y_2', \dots, Y_N'\}$  - set of output signals of type 1;
- $Y'' = \{Y_1'', Y_2'', \dots, Y_N''\}$  - set of output signals of type 2;
- $Y''' = \{Y_1''', Y_2''', \dots, Y_N'''\}$  - set of output signals of type 3;

- $Q = \{ Q_1, Q_2, \dots, Q_N \}$  – is an arbitrary set of states;
- $\pi = \{ \pi_1, \pi_2, \dots, \pi_N \}$  – the set of blocks of states of the subautomata  $S_i$ ;
- $E_0 = \{ e_{1_0}, e_{2_0}, \dots, e_{N_0} \}$  – initial saving the input signal;
- $a_0 = \{ a_{1_0}, a_{2_0}, \dots, a_{N_0} \}$  – the initial state of the  $S_i$  subautomaton;
- $F_1: Q \times X \rightarrow Q$  – is a single-valued transition function realizing the mapping  $D_{F1} \subseteq Q \times X$  in  $Q$ ;
- $F_2: Q \times e_j \rightarrow \pi_j$  – is the conservation function of the state blocks, realizing the map  $D_{F2} \subseteq Q \times e_j$  to  $\pi_j$ ;
- $F_3: Q \times E \rightarrow \pi_j$  – is the coarse transition function realizing the mapping  $D_{F3} \subseteq Q \times E$  to  $\pi_j$ ;
- $\lambda_1: Q \times X \rightarrow Y'$  – is a function of outputs of type 1 that implements the mapping  $D_{\lambda1} \subseteq Q \times X$  to  $Y'$ ;
- $\lambda_2: \pi_j \rightarrow Y''$  – is a function of outputs of type 2, realizing the map  $D_{\lambda2} \subseteq \pi_j$  onto  $Y''$ ;
- $\lambda_3: Q \times E \rightarrow Y'''$  – is a function of outputs of type 3 that implements the mapping  $D_{\lambda3} \subseteq Q \times E$  onto  $Y'''$ .

Subautomata  $S_i$ , realizing their automaton memory on open-type registers, operate in automatic continuous time  $T$ .

At the initial clock time  $t_0$ , all the  $S_i$  subautomata are set to the initial state  $a_0$  by the corresponding input signal  $N_{i_0}(t_0) \in X_i$ . During the subsequent internal cycle  $\Delta_0$ , the initial state  $a_0$  remains under the influence of the initial preserving  $e_{i_0}(\Delta_0)$  of the input signals. The union of the states of the  $S_i$  subautomata determines the states  $a_i$  of the hierarchical automaton  $a_i$  a given clock time  $t_i$  or  $\Delta_i$  of the automatic continuous time  $T_i$ . In the hierarchical automaton  $A$ , at the time  $t_i$ , all or only some  $S_i$  subautomata can perform single-pass functions, realizing the common function  $\delta_{i_0}$  of single-valued transitions  $F_1$  of the hierarchical automaton  $A$  to the new state  $a_s(t) = \bigcup_i a_{i_j}(t)$ . During the internal cycle  $\Delta_i$ , the  $S_i$  subautomata can perform the

coarse transitions  $\delta_{i_y}$  realizing the common transition function  $F_3$  of the hierarchical automaton  $A$  to the new state  $a_k(\Delta) = \bigcup_i a_{i_j}(\Delta)$ . If the  $S_i$  subautomata do not perform transitions to the new state during the entire outer clock cycle  $T$ , then, consequently, they realize the function  $\delta_{i_e}$  is stored states in implementing the IA and the combined function of storing  $F_2$  states.

Each of the  $S_i$  subautomata operates in a certain state block  $\pi_{i_j}$  of the whole set of its states  $Q_i$ . The blocks  $\pi_{i_j}$  of states of  $S_i$  subautomata form a definite block  $\pi_{i_j}$  of the set of state blocks  $\pi_{i_j}$  in IA  $A$ , in which the IA  $A$  is functioning at a given time.

A characteristic feature of IA is the possibility of interaction of  $S_i$  subautomata not only during the clock  $t_i$ , but also during the internal cycle  $\Delta_i$  of the automatic continuous time  $T_i$ . The memory of the  $S_i$  subautomata is a matrix structure in which, during the time  $t_i$ , under the influence of the information signals  $x_i(t)$  of the input signals, the  $S_i$  subautomaton is able to go from one state to another in one state block  $\pi_{i_j}$  (the matrix structure line of the automatic memory circuit). And during the internal cycle  $\Delta_i$ , under the influence of the input signals retaining  $e_j(\Delta)$ , the  $S_i$  subautomaton is able to go from one state to another in one block  $\mu_i$  (the column of the matrix structure of the automatic memory circuit), that is, from a certain state of one block  $\pi_{i_j}$  to a certain state of another block  $\pi_{i_j}$  states.

Thus, the mathematical model of a multilevel hierarchical automaton  $A$  is able to describe the functioning of not only parallel-running  $S_i$  subautomata, but also their interfacial interaction by using the letters of the input preserving  $E$  alphabet of the  $S_i$  subautomata.

## 2.4. Conclusion to the 2nd chapter

The second chapter discusses the principle of hierarchical program management, which allows you to present multi-level control information with a vertical relation-

ship between levels and the ability to process this information simultaneously. The meaning of probabilistic and fuzzy transitions in real devices is shown on the basis of the theory of F.P. Ramsey, which is different from the mathematical representation. Management of multi-level abstract automata is described on the basis of the principle of hierarchical program control.

## Chapter 3

### METHODS OF ASSIGNING AUTOMATICS

#### 3.1. Tabular methods for specifying multifunctional abstract determinate automata

The assignment of multifunction abstract automata of the 1st, 2nd, and 3rd kinds is reduced to the specification of the corresponding single-valued transition functions  $\delta_o$ , the output functions  $\lambda_1, \lambda_2, \lambda_3$ , the conservation function  $\delta_e$ , and the function of the integrated  $\delta_y$  transitions. Let us discuss in more detail the methods for specifying automata using tables and graphs [1].

The description of the multifunctional automaton of the first kind is given by sets (at least two) of the transition tables, the sets of output tables and the table of conservation of states. Let us illustrate the task of multifunctional automaton  $A_I$  of the first kind in Tables 3.1 to 3.5.

The rows in Tables 3.1 to 3.5 correspond to the input signals  $x_i$ , and the columns to the blocks of  $\pi_I$  states that conserve the  $e_j$  of the input signals. Usually the last column in the tables is denoted by the initial state  $a_0$ . The state  $a(t) = \delta_o^t(a(\Delta-1), x(t))$ , is placed on the intersection of rows and columns of single-valued transition tables, in which an automaton of the first kind is transferred from the state  $a(\Delta-1)$  under the influence of the information  $x(t)$  of the input signal to the state  $a(t)$  at the time  $t$  of the automatic continuous time  $T$  [2]. In the output tables of an automaton of the first kind, the output signal  $y(t) = \lambda_1(a(\Delta-1), x(t))$  of type 1 corresponding to the transition.

The rows of the state preservation table (Table 3.5) correspond to the retaining  $e(\Delta)$  input signals, and the columns to  $a(\Delta)$  to the states of the automaton. In each row at the intersection with the columns in the table of conservation of states, the corresponding state  $a(\Delta) = \delta_e(a(t), e(\Delta))$ , where  $a \in \pi_j$ ,  $a(t) = a(\Delta)$  ( $\pi_j \in Q$ ), stored

under the influence of the retaining  $e(\Delta)$  input signal during the internal cycle  $\Delta$  of the automatic continuous time  $T$ , and in the remaining cases dashes are placed.

Table of Transitions 3.1

$$\delta_0^1: Q(\Delta-1) \times X(t) \rightarrow Q(t)$$

$\pi_1$	$a_0$	$a_1$	$a_2$	$a_3$
$x_i$				
$x_1$	$a_1$	$a_2$	$a_1$	$a_0$
$x_2$	$a_2$	$a_3$	$a_0$	$a_2$

Transition Table 3.2

$$\delta_0^2: Q(\Delta-1) \times X(t) \rightarrow Q(t)$$

$\pi_2$	$a_0$	$a_4$	$a_5$	$a_6$
$x_i$				
$x_1$	$a_4$	$a_0$	$a_6$	$a_5$
$x_2$	$a_5$	$a_6$	$a_4$	$a_0$

Table of Transitions 3.3

$$\lambda_1^1: Q(\Delta-1) \times X(t) \rightarrow Y_1(t)$$

$\pi_1$	$a_0$	$a_1$	$a_2$	$a_3$
$x_i$				
$x_1$	$y_1$	$y_2$	$y_3$	$y_4$
$x_2$	$y_2$	$y_1$	$y_4$	$y_3$

Transition Table 3.4

$$\lambda_1^2: Q(\Delta-1) \times X(t) \rightarrow Y_1(t)$$

$\pi_2$	$a_0$	$a_4$	$a_5$	$a_6$
$x_i$				
$x_1$	$y_5$	$y_5$	$y_6$	$y_7$
$x_2$	$y_6$	$y_7$	$y_5$	$y_6$

Table of Transitions 3.5

$$\delta_e: Q \times e_j \rightarrow \pi_j$$

$Q$	$a_0$	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$
$e_j$							
$e_1$	$a_0$	$a_1$	$a_2$	$a_3$	—	—	—
$e_2$	$a_0$	—	—	—	$a_4$	$a_5$	$a_6$

In describing the operation of the multifunctional automatic machine  $A$  of the second kind, the output signal depends only on the memorized states of the automaton unit  $\pi_j$ , which are set at the time moment  $t$  and are preserved during the realiza-

tion of the state conservation functions during the internal time interval  $\Delta$  of the automatic continuous time  $T$ .

Automata of the 1st and 2nd kind are able to perform the transition functions at the time  $t$  of the automatic continuous time  $T$  under the influence of information  $x(t)$  input signals.

The operation of an automaton of the second kind can be specified with the help of the marked tables of functions  $\delta_o$  of single-valued transitions together with the function  $\lambda_2$  of the output table of type 2 and the table of conservation of states (Table 3.5). An example of a tabular description of an automaton of the 2nd kind of  $A_3$  is given in Tables 3.5-3.7.

Automata of the third kind differ qualitatively from automata of the first and second kind in that they are able to carry out transitions from one state to another not only at the time moment  $t$  under the influence of information  $x(t)$  input signals, but also during the internal  $e(\Delta)$  of the input signals of the automatic continuous time  $T$ . This fundamental difference of the automata of the third kind allows them to carry out transitions in the matrix structure of the automaton memory circuits (MFIS) that occur not only in the states  $\pi_j$  blocks into the clock the moment  $t$  representing the MFIS state line, but also in blocks  $\mu_i$  during the internal cycle  $\Delta$  representing the MFIS state column, for one external cycle  $T$  of the automatic continuous time.

Marked Transition Table 3.6

$$\delta_o^1: Q(\Delta-1) \times X(t) \rightarrow Q(T)$$

$y_i$	$y_0$	$y_1$	$y_2$	$y_3$
$\pi_1$	$a_0$	$a_1$	$a_2$	$a_3$
$x_i$				
$x_1$	$a_1$	$a_2$	$a_1$	$a_0$
$x_2$	$a_2$	$a_3$	$a_0$	$a_2$

Marked Transition Table 3.7

$$\delta_o^2: Q(\Delta-1) \times X(t) \rightarrow Q(T)$$

$y_i$	$y_0$	$y_4$	$y_5$	$y_6$
$\pi_2$	$a_0$	$a_4$	$a_5$	$a_6$
$x_i$				
$x_1$	$a_4$	$a_0$	$a_6$	$a_5$
$x_2$	$a_5$	$a_6$	$a_4$	$a_0$

The automata of the 1st, 2nd and 3rd kind also differ by their functions  $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$  of the output signals considered in the automaton continuous time  $T$ .

The output function  $\lambda_1$  of automata of the first kind is realized only at the time moment  $t$  under the influence of information  $x(t)$  input signals and states of the automaton  $a(\Delta-1)$ , that is,  $y(t) = \lambda_1(a(\Delta-1), x(t))$ , where  $y \in Y_I$ .

The output function  $\lambda_2$  of the automata of the second kind is realized at the time instant  $t$  after the transition of the automaton to the new states under the influence of the information  $x(t)$  of the input signals and the state of the automaton  $a(\Delta-1)$  that was earlier in time. The output function  $\lambda_2$ , in the first place, is by this shifted in comparison with the beginning of the clock moment  $t$ , and also depends only on the newly established state  $a(t)$  and the conserved state  $a(\Delta)$ , where  $a(t) = a(\Delta)$  and  $y(T) = \lambda_2(a(t), a(\Delta))$ , where  $y \in Y_{II}$ .

The output function  $\lambda_3$  of the automata of the third kind is realized during the internal time moment  $\Delta$  in the new states under the influence of the input signals and the states of the automaton  $a(\Delta)$  that preserve  $e(\Delta)$ . The output function  $\lambda_3$ , firstly, is shifted in comparison with the beginning of the internal clock moment  $\Delta$ , and also depends only on the newly established state  $a(\Delta)$  and on the input signal retained by  $e(\Delta)$ , that is,  $y(\Delta) = \lambda_3(a(\Delta), e(\Delta))$ , where  $y \in Y_{III}$ .

The operation of the automaton of the third kind can be specified using the tables of functions  $\delta_0$  of single-valued and distinguished tables of functions  $\delta_y$  of the enlarged transitions together with the function  $\lambda_3$  of the output table of type 3. An example of a tabular description of an automaton of the third kind of  $A_3$  is given in Tables 3.8-3.11 and tables of conservation of states (Table 3.5).

Table of Transitions 3.8

$$\delta_0^1: Q(\Delta-1) \times X(t) \rightarrow Q(t)$$

$\pi_1$	$a_0$	$a_1$	$a_2$	$a_3$
$x_i$				
$x_1$	$a_1$	$a_2$	$a_1$	$a_0$
$x_2$	$a_2$	$a_3$	$a_0$	$a_2$

Transition Table 3.9

$$\delta_0^2: Q(\Delta-1) \times X(t) \rightarrow Q(t)$$

$\pi_2$	$a_0$	$a_4$	$a_5$	$a_6$
$x_i$				
$x_1$	$a_4$	$a_0$	$a_6$	$a_5$
$x_2$	$a_5$	$a_6$	$a_4$	$a_0$

Marked Transition Table 3.10

$$\delta_0^1: Q(t) \times E(\Delta) \rightarrow Q(\Delta)$$

$y_i$	$y_0$	$y_1$	$y_2$	$y_3$
$\pi_1$ $e_i$	$a_0$	$a_1$	$a_2$	$a_3$
$e_1$	$a_1$	$a_2$	$a_1$	$a_0$
$e_2$	$a_2$	$a_3$	$a_0$	$a_2$

Marked Transition Table 3.11

$$\delta_0^2: Q(t) \times E(\Delta) \rightarrow Q(\Delta)$$

$y_i$	$y_0$	$y_4$	$y_5$	$y_6$
$\pi_2$ $e_i$	$a_0$	$a_4$	$a_5$	$a_6$
$e_1$	$a_4$	$a_0$	$a_6$	$a_5$
$e_2$	$a_5$	$a_6$	$a_4$	$a_0$

### 3.2. Tabular methods for specifying abstract probability automata of the third kind

In contrast to deterministic automata in which only one-to-one transition from one state to another is possible under the influence of elementary single-valued  $p_0(T)$  and enlarged  $p_y(T)$  input words, the probabilistic automata transition to the state of a certain block  $\pi_j$  of states of the automaton with a certain probability  $P_e$  transition under the influence of elementary probabilistic  $p_b(T)$  input words.

A probabilistic abstract automaton of the third kind of the first type is given by means of a table of conservation of states, as in deterministic automata (Table 3.5), tables of single-valued transition to the  $a_p(t)$  state (Table 3.12-3.13), a marked transition table in  $a(\Delta)$  the state of the  $\pi_j$  state block (Table 3.14) with a certain transition probability  $P_e$ .

Table of Transitions 3.12

$$\delta_0^1: Q(\Delta-1) \times x_p(t) \rightarrow a_p(t)$$

$\pi_1$ $x_i$	$a_0$	$a_1$	$a_2$	$a_3$
$x_p$	$a_p$	$a_p$	$a_p$	$a_p$

Transition Table 3.13

$$\delta_0^2: Q(\Delta-1) \times x_p(t) \rightarrow a_p(t)$$

$\pi_2$ $x_i$	$a_0$	$a_4$	$a_5$	$a_6$
$x_p$	$a_p$	$a_p$	$a_p$	$a_p$

Marked table of probability transitions 3.14

$$\delta_{B_1}^1 : a_p(t) \times e_j(\Delta) \rightarrow \pi_j(a_B(\Delta))$$

$$\lambda_3^{B_1} : a_p(t) \times e_j(\Delta) \rightarrow Y_{III}^{B_1} (y^{\delta_1}(\Delta))$$

	$y^{\delta_1}$	
$e_j$	$y_0(P_{e1}) \vee y_1(P_{e1}) \vee \dots \vee y_k(P_{e1})$	
$e_j$	$a_p$	$\pi_j$
$e_0$	$a_0(P_{e1}) \vee a_1(P_{e1}) \vee \dots \vee a_{k1}(P_{e1})$	$\pi_0$
...	...	...
$e_m$	$a_0(P_{e1}) \vee a_1(P_{e1}) \vee \dots \vee a_{km}(P_{e1})$	$\pi_m$

A probabilistic abstract automaton of the third kind of the second type is given by the table of conservation of states, as in deterministic automata (Table 3.5), tables of single-valued transition to  $a(t)$  state (Table 3.15-3.16), a marked transition table in  $a(\Delta)$  the state of the  $\pi_j$  state block (Table 3.14) with a certain transition probability  $P_e$ .

Table of Transitions 3.15

$$\delta_0^1 : Q(\Delta-1) \times X(t) \rightarrow Q(t)$$

$\pi_1$	$a_0$	$a_1$	$a_2$	$a_3$
$x_i$				
$x_1$	$a_1$	$a_2$	$a_1$	$a_0$
$x_2$	$a_2$	$a_3$	$a_0$	$a_2$

Transition Table 3.16

$$\delta_0^2 : Q(\Delta-1) \times X(t) \rightarrow Q(t)$$

$\pi_2$	$a_0$	$a_4$	$a_5$	$a_6$
$x_i$				
$x_1$	$a_4$	$a_0$	$a_6$	$a_5$
$x_2$	$a_5$	$a_6$	$a$	$a_0$

Probabilistic transitions in the automaton of the third kind of the first type occur in a definite block of  $\pi_j$  states, and in the third-kind automaton of the second kind occur in a certain block of  $\mu_i$  states. Thus, probabilistic automata of the third kind of the first type and probabilistic automata of the third kind of the second type carry out probabilistic transitions to states of completely defined subsets of the corresponding blocks  $\pi_j$  and  $\mu_i$  states with a certain measure of probability.

Marked

Table of Transitions 3.17

$$\delta_0^1: Q(\Delta-1) \times X(t) \rightarrow Q(T)$$

$y_i$	$y_0$	$y_1$	$y_2$	$y_3$
$\pi_1$				
$x_i$	$a_0$	$a_1$	$a_2$	$a_3$
$x_1$	$a_1$	$a_2$	$a_1$	$a_0$
$x_2$	$a_2$	$a_3$	$a_0$	$a_2$

Marked

Transition Table 3.18

$$\delta_0^2: Q(\Delta-1) \times X(t) \rightarrow Q(T)$$

$y_i$	$y_0$	$y_4$	$y_5$	$y_6$
$\pi_2$				
$x_i$	$a_0$	$a_4$	$a_5$	$a_6$
$x_1$	$a_4$	$a_0$	$a_6$	$a_5$
$x_2$	$a_5$	$a_6$	$a_4$	$a_0$

An abstract fuzzy automaton of the third kind is defined by means of tables of conservation of states, as in multifunctional deterministic automata (Table 3.5), tables of single-valued transition to  $a_p(t)$  state, as in probabilistic automata of the third kind of the first type (Table 3.12- 3.13), and the marked transition table in  $a(\Delta)$  the state of the  $\pi_j$  state block (Table 3.18) with a certain probability  $P_n$  of the transition.

Reported fuzzy transition table 3.19

$$\delta_y^H: a_p(t) \times e^H(\Delta) \rightarrow \pi_j(a_n(\Delta))$$

$$\lambda_3^H: a_p(t) \times e^H(\Delta) \rightarrow Y_{III}^H (y^H(\Delta))$$

	$y^H$	
$e_j$	$y_0(P_H) \vee y_1(P_H) \vee \dots \vee y_k(P_H)$	
$e_j$	$a_p$	$\pi_j$
$e_0$	$a_0(P_H) \vee a_1(P_H) \vee \dots \vee a_{k1}(P_H)$	$\pi_0$
...	...	...
$e_m$	$a_0(P_H) \vee a_1(P_H) \vee \dots \vee a_{km}(P_H)$	$\pi_m$

The graphical method for specifying abstract automata of the third kind was considered earlier in the corresponding figures, where they are described:

- for deterministic transitions in Fig. 1.5 (single-valued transition) and Fig. 1.6 (enlarged transition).

- for probabilistic and fuzzy transitions in Fig. 1.13 (probabilistic transitions of the first and second types) and Fig. 1.14 (fuzzy transitions).

Abstract automata with memory are used to realize the functions of single-valued transition to the new state  $a_i(t)$  with the corresponding input signal  $x(t)$ , provided this state is maintained after the end of the signal  $x(t)$ , that is, in the time of the internal cycle  $\Delta$  of the automatic continuous time  $T$ .

The output signals of abstract automata of the first and second kind are respectively represented in the graph either on the arc of the transition from one state to another in automata of the first kind, or near the vertex of the graph in automata of the second kind.

Multifunctional deterministic abstract automata of the third kind are used not only to realize the functions of single-valued transition to the new state of  $a(t)$  with the corresponding input signal that establishes  $x(t)$ , provided this state is maintained after the signal  $x(t)$  ends, as shown in Fig. 1.5; but also with the use of transitions during the internal cycle  $\Delta$  to realize the enlarged transition to the new state  $a(\Delta)$  with the corresponding  $e(\Delta)$  input signal, as shown in Fig. 1.6.

The output signals of abstract automata of the third kind are respectively represented, as in automata of the second kind, near the vertex of the graph.

Probabilistic abstract automata of the first and second types of the third kind are used to realize the probability transition functions in a new state  $a(\Delta)$  for the corresponding input elementary words  $p_{b1}(T)$  or  $p_{b2}(P_e)$  with a certain probability  $P_e$  during the internal cycle  $\Delta$ , as this is shown in Fig. 1.13 (for a probabilistic automaton of the third kind of the first type and of the second type).

Fuzzy abstract automata of the third kind are used to realize the fuzzy transition functions to the new state  $a(\Delta)$  with the corresponding elementary input word  $p_n(T)$  with a certain probability  $P_n$  during the internal clock cycle  $\Delta$ , as shown in Fig. 1.14.

Thus, multifunctional abstract automata increase the types of transitions used in classical sequential abstract automata, extend the concept of the functioning of an

automaton from discrete automaton time  $t$  to an uninterrupted automaton time  $T$ , which makes it possible to study their work more deeply.

### **3.3. The method of specifying a hierarchical abstract machine with a multifunctional system of organizing memory using a polygram**

The automaton can be conveniently described algorithmically, considering its operation in sequence with each transition from one state to another during the time of one external cycle  $T_i$  of the automatic continuous time. A clear way to define classical automata with memory on flip-flops is to assign them in the form of a microprogram [3] or in a more generalized form of an autogram [4].

For the algorithmic description of hierarchical automata with automaton memory with a general state consisting of partial states of  $S_i$  subautomata, the term "polygram" is proposed for the following reasons.

Firstly, the term "microprogram", "autogram" and other ways of specifying automata with memory on flip-flops are oriented quite justifiably to the description of the transition functions in automata only during the time moment  $t_i$ , have a strong restriction that does not allow describing the operation of the automaton during the internal clock cycle  $\Delta_i$  of the automatic continuous time  $T_i$ .

Secondly, the concepts of "microprogram", "autogram" and other ways of setting classical automata are oriented to their functioning in automatic discrete time. In this connection, the terms "microprogram", "autogram" is not able to describe the general state of registers on automatic memory circuits. The term "polygram" is associated with the schematic implementation of control in multi-level registers with a multifunctional memory organization, which is called a multi-level hierarchical automatic implementation [2].

Thirdly, the polygraph is oriented not only to the transformation of input information into output information, but also to the change in the region of memorized states of the automaton, which is determined by the retaining  $e_j(\Delta)$  input signal. This feature makes it possible to use a polygram in describing the structure of an automaton capable of transforming the input information during the clock  $t$  and changing the internal structure of the information transformation during the internal clock cycle  $\Delta$  of the continuous automatic time  $T$  [2].

The polygram describes each state of the hierarchical automaton (IA)  $A$  as a union of the states of the subautomata  $S_i$ .

$$a_k = \bigcup_i a_i \quad (3.1)$$

At each point of the polygram, the operating modes of  $S_i$  subautomata are described for one external cycle  $T$  of automatic continuous time. For one external clock cycle  $T$ , IA  $A$  perceives the input word  $R_k(T)$ , consisting of a set of elementary  $p_i(T)$  input words of the subautomata  $S_i$

$$R_k = \bigcup_i p_i(T). \quad (3.2)$$

Under the influence of the input word  $R_k$ , the IA  $A$  is able to go to the new state and output the output signals  $Y_k^i$  ( $i = 1, 2, 3$ ) consisting of a set of output signals  $Y_k^i$  of certain types of  $S_i$

$$Y_k^1(t) = \bigcup_i Y_i^1(t); \quad (3.3)$$

$$Y_k^2(T) = \bigcup_i Y_i^2(T); \quad (3.4)$$

$$Y_k^3(\Delta) = \bigcup_i Y_i^3(\Delta), \quad (3.5)$$

where the output signals  $Y_i^1(t)$ ,  $Y_i^2(T)$ ,  $Y_i^3(\Delta)$  are determined respectively by the output functions described in equations (3.3) - (3.5).

При неизменяемых сохраняющих  $e_i(\Delta)$  входных сигналах подавтоматы  $S_i$  функционируют в определенных блоках  $\pi_{i_j}$  своих состояний, а, следовательно, и IA  $A$  функционирует в те же периоды времени в определенных блоках  $\pi_k$  своих состояний

$$\pi_k = \bigcup_i \pi_i \quad (3.6)$$

Each  $a_i$  state of the subautomaton  $S_i$  retains its value for the corresponding input signals retaining  $e_j$ . The change in the input signal retained in the  $S_i$  subautomaton takes place at the clock time  $t$ , depending on the output signals coming from the other  $S_i$  automata in accordance with the problem solving algorithm, and during the internal

cycle  $\Delta$  the input-preserving  $e_i(\Delta)$  input signal determines the region of memorized states (states) of the  $S_i$  subautomaton. The conserved  $e_i(\Delta)$  input signals uniquely determine the state blocks in which the  $S_i$  automaton operates, and the set  $E_k$  of the input signals that preserve  $e_i(\Delta)$  uniquely determine the block of  $\pi_k$  states in which the IA  $A$  works

$$E_k = \bigcup_j e_j \quad (3.7)$$

The polygraph point with the deterministic elementary input word  $R_k$  is written in the following form:

$$\begin{aligned} KE. R_1 \rightarrow Y_1^1 - K_1 E_1, \\ R_2 \rightarrow Y_2^1 - K_2 E_2, \\ \text{-----} \\ R_m \rightarrow Y_m^1 - K_m E_m, \end{aligned} \quad (3.8)$$

where  $K$  is the state of IA  $A$ ;

$E$  - preserves the input signal IA  $A$ ;

$R_j$  ( $j = 1, 2, \dots, m$ ) is the elementary input word IA  $A$ ;

$K_j$  ( $j = 1, 2, \dots, m$ ) is the state of IA, to which the transition from the state  $K$  occurs when the line  $j$  is executed;

$E_j$  ( $j = 1, 2, \dots, m$ ) - preserving the input signal, at which the state  $K_j$  is memorized; - output signal of type IA  $A$ ;

$Y_j^i$  ( $j = 1, 2, \dots, m$ ) - output signal of type IA  $A$ .

Each line ( $R_j \rightarrow Y_j^1 - K_j E_j$ ) of the  $KE$  point of the polygram is described as  $S_k$  lines

$$\begin{aligned}
a_1 e_1 p_{j1} &\rightarrow Y_{j1}^i - a_{j1} e_{j1}, \\
a_2 e_2 p_{j2} &\rightarrow Y_{j2}^i - a_{j2} e_{j2}, \\
&\text{-----} \\
a_q e_q p_{jq} &\rightarrow Y_{jq}^i - a_{jq} e_{jq},
\end{aligned} \tag{3.9}$$

where  $q$  is the number of rows of  $S_k$  ( $k = 1, 2, \dots, m$ ) in the description of the line of the polygram  $j$  ( $j = 1, 2, \dots, m$ );

$p_{j_i}$  - elementary input word of the string  $S_k$ ;

$Y_{j_i}^i$  - output vector of the string  $S_k$ ;

$a_{j_i}$  - state of the subautomaton  $S_i$ , to which the transition from the state  $a_i$  to the state  $a_k$  occurs when the line  $S_k$  is executed;

$e_i$  - preserving the input signal of state  $a_i$ .

This description of the behavior of IA  $A$  in the state  $KE$  is the point of the hierarchical algorithm for the functioning of the automaton  $A$ .

The point of the polygraph  $KE$ , as can be seen from the description of the behavior of IA  $A$ , has a hierarchical structure, that is, in the beginning generalized states  $K_0, K_1, \dots, K_m$ , generalized elementary input words  $R_j$  ( $j = 1, 2, \dots, m$ ), generalized input signals  $E_0, E_1, \dots, E_m$ , and then each line of the polygraph point unfolds in  $S_k$  ( $k = 1, 2, \dots, m$ ) of strings that are realized in the  $S_i$  subautomata IA  $A$ , passing from one state  $a_i$  to the state  $a_k$  when executing lines  $S_k$ .

During the functioning of the IA, some, and, possibly, all  $S_i$  subautomata may not change their states at some interval of automaton time. Then, when describing the IA  $A$  polygraph, at the end of the lines there will be notations of states that coincide with the initial notation of the point of the polygram or the point of the line of the polygram.

Each  $j$ -th line of the point of the polygraph  $KE$  consists of the following elements, the functioning of which is described in the automatic continuous time  $T$ : the designation of the  $KE$  point displaying the state  $K$  ( $\Delta-1$ ) stored at the input signal  $E$  ( $\Delta-1$ ); the elementary input word  $R_j(T)$ , consisting of successive respectively infor-

mation  $X_j(t)$  input signal and preserving  $E_j(\Delta)$  of the input signal; output signals described by equations (3.3) - (3.5).

The meaningful meaning of each  $j$ -th line of the polygram consists in establishing the connection between the  $K(\Delta-1)$  IA  $A$  states in the previous external clock cycle  $T_{i-1}$  and the IA  $A$  reaction in the next external clock cycle  $T_i$ -transition to the state  $K_j(\Delta)$ .

Each  $j$ -th line of the polygram can be represented from three parts. The first part specifies the vector  $R_j$ , the second part - the vector of the output signals, and the third part defines the function  $F_2$  of the single-valued conservation of the state vector  $K_j$ , or the function  $F_3$  of the integrated transition to the state  $K_s$  with the vector  $E_j$  of the preserving input signal.

The vector  $E_j$  of the input signal preserving function  $F_2$  of the single-valued conservation of the state vector  $K_j$  defines a certain block  $\pi_j$  of the remembered states IA  $A$  in which it operates at a given time  $\Delta$  of the automatic continuous time  $T$ , and in the implementation of the function  $F_3$ , an enlarged transition to the state  $K_s$  at the time of the internal cycle  $\Delta$  of the automatic continuous time  $T$ . The input signal  $E_j$  provides an inter-level connection of the Si subautomata to the IA  $A$  at the instant of the internal clock cycle  $\Delta$  of the automatic continuous time  $T$ .

The logic of the operation of a system in which the internal interrelation has exactly the same or even greater importance than their external one, often confronts us with the difficulties of formulating interrelated polygrams.

The transformation of topics is more perfect than the more complex and better organized structure of the system it accompanies.

Changing the internal structure of memory states in podavto- $S_i$  and mats in general IA  $A$ , and the conversion information in the incoming external IA  $A$  - is only two interconnected parts of the same process in the conversion information IA  $A$ .

Let's describe some basic types of lines of a polygram.

The line of the polygram having the form

$$KE. R_j \rightarrow Y_j - K_j E_j, \quad (3.10)$$

will be called a common string, emphasizing this term that all three vectors of the above command are represented in this line.

In addition to the common lines in the polygram, there can be lines in which only one of the vectors is explicitly represented, while the other vectors of the command are omitted. The vector skip is represented by the symbol  $\emptyset$ . If the components of the vector are omitted, then the present component is represented instead of the vector.

For example, a string of the form

$$KE. E_i \rightarrow Y_j - K_j E_j \text{ при } E_i \neq E_j \quad (3.11)$$

we shall call a transition row of the third kind, since it describes a transition effected only by the components of the vector  $E_i$  of the stored input signals. In this case,  $X_i = \emptyset$ , and  $E_i \in R_i$ .

A string of the form

$$KE. R_j \rightarrow \emptyset - K_j E_j, \quad (3.12)$$

will simply be called a transition line, since only the transition to the next state  $K_j E_j$  without a vector of output signals is represented in it.

A string of the form

$$KE. R_j \rightarrow Y_j - \emptyset, \quad (3.13)$$

will be called the output string, since it does not represent a transition to the state vector  $K_j$  and does not represent the vector  $E_j$  that preserves the new state.

If the vectors  $E_i$  and  $E_j$  coincide in a row, they can be omitted. A string then (for example, a generic one) takes this form as an output string, since it does not rep-

represent a transition to the state vector  $K_j$  and does not represent a vector  $E_j$  that preserves the new state.

If the vectors  $E_i$  and  $E_j$  coincide in a row, they can be omitted. The string then (for example, the general) takes this form

$$K. R_{j \rightarrow} Y_j - K_j, \quad (3.14)$$

and the vectors  $E_i$  and  $E_j$  ( $E_i = E_j$ ) in this case are implied and define a well-defined block  $\pi_j$  ( $K, K_j \in \pi_j$ ) of remembered states. Such a line (3.16) is called a C-type and is used in the description of the microprogram or autogram [26] of the Mealy, Moore and C automata automata, which use as triggers in registers whose states are preserved for one conserved  $e(\Delta)$  input signal. This description of the firmware underscores once again that the Mealy, Moore and C-automata automata are a subset of the Marakhovsky multifunctional automata, on the basis of which a polygraph describing IA  $A$ , built on automatic memory circuits, is considered.

We note one important feature of the output vectors  $Y_{j_i}^i$  IA  $A$ . The components  $Y_{k_i}^i$  of the vectors  $Y_{j_i}^i$  (3.3) - (3.6) mentioned above can simultaneously initiate various  $T$  operations in controlled objects in the external clock cycle, as well as the functional disconnection of one of the  $S_i$  IA  $A$ , in determining its uselessness or error in the work. This property is important when creating fault-tolerant digital devices [5].

The IA system is considered fault-tolerant or insensitive to non-serviceability if its organization provides for the elimination of faulty  $S_i$  automata from the field of functioning of the IA due to the use of hardware, information and algorithmic redundancy.

Part IA  $A$ , designed to handle general information, is called an EA strategy automaton or abbreviated as IA  $A_M$ . The structure of IA is hierarchical and can be fault-tolerant.

The polygram allows us to describe the IA in general from the general positions of the functioning of hierarchical systems with parallel execution of the branches of the algorithm by  $S_i$  submodules with parallel execution of the algorithm by the automaton of the strategy that processes the general information.

Thus, the polygraph describes not only the tasks of  $S_i$  subautomata, but their inter-layer interaction in IA  $A$  due to the retaining input signals coming from the subautomata of the strategy.

### **3.4. Formulation of a polygram**

Working with the polygraph is simplified if you follow the following rules.

1. The point of a polygraph is conveniently designated by two numbers marked with  $K$  and  $E$ . For example,  $12K, 2E$ .

2. The starting point usually has the number  $11K, 1E$ , and the remaining points of the polygraph are numbered with the numbers up to the maximum number with the symbol  $K$  and up to the maximum number with the symbol  $E$ .

3. The items in the polygram should be arranged in ascending order at the beginning of the number with the symbol  $K$ , numbers near the symbol  $E$ , and after changing the number with the symbol  $E$  the numbering with the symbol  $K$  can be started from the beginning, that is, from the number  $1K$ .

4. The points forming a continuous sequence in a polygram with one number with the symbol  $E$  can be conveniently numbered by a continuous series of numbers.

5. Installation points do not have specific numbers, since they are not taken into account when counting the number of items in a polygraph. The location of the installation point should be determined differently at the beginning of numbering with the symbol  $K$  at a certain number with the symbol  $E$ .

Note. Before starting the polygraph, it is advisable to check for catastrophic failures of the memory circuit of the  $S_i$  IA submodromotors, as was done in the 4th-order automatic machine. When catastrophic failures are detected, such  $S_i$  IA  $A$  subautomata should be disconnected from operation and replaced with new ones (work-

able) or taken into account when the polygram is operating. A polygram is called a correct polygram, which, when realized in an automaton synthesized by formal methods, leads to the functioning of a reliable device that corresponds to its purpose. Assigning an automaton in a tabular form, in the form of a graph, in the form of a polygram or in some other way is the creativity of the designer. On the one hand, the designer must fully understand the operation of the object and the temporal sequences of control actions necessary for controlling the object, on the one hand, and on the other hand, to master the methods of formal synthesis of automata according to a polygram on an accessible element base with automatic memory circuits for superintense integrated circuits (VLSI). Verification of the correct (correct) operation of the polygram or the designed control device can be performed by means of modeling the device on the computer [74].

In the process of formulation of the polygram, an original analysis of the method of solving the problem is carried out into stages, steps and points distributed over time, comprehension of the complex solution of the problem in the implementation of all points, steps and stages. The chosen solutions in the process of program formulation are usually reviewed many times in order to achieve the best organization of object management and obtain the best performance of the implemented system (minimum costs, maximum speed, increased reliability, and so on).

In connection with the hierarchical ability of the internal organization of polygrams, it is necessary to first isolate the whole process of processing private information when formulating a solution to a problem. The generalized part can be provided by the developer and specified in a polygram of a more general level.

In general, the process of formulating a polygraph is of a creative nature and, unfortunately, is not formalized. The author suggests some recommendations when writing a polygram.

1. It is advisable to start describing a polygram with general information of the algorithm, within which the whole process of solving the problem is being processed.

2. The process of splitting the general information of the algorithm into levels is possible with the hierarchical nature of information processing during the solution of the problem.

3. In describing the process of processing hierarchical information, it is advisable to consider, firstly, each process separately in the form of its private polygraph, and, secondly, each point of the private polygram is viewed in time sequence as the point of the entire polygram.

4. When analyzing the points of a polygram, it is necessary to take into account the points of all levels of the polygraph, which collectively describe the IA  $A$  generalizing states and its response to all input signals.

5. The time sequence of polygraph points should bring the controlled object to its original state. This is the expression of a hierarchical (or any other) algorithm for solving the problem.

6. When writing a polygram, it is necessary to describe in sufficient detail the input and output signals of the controlled object and to present them in brief and expressive notations, so as not to create difficulties in the formulation of the polygraph points.

7. The formulation of the polygram must begin with the installation point. If there are several setting points in the polygram, then you can start with an arbitrary one, if there is no priority between them.

8. The structure of the polygraph should reflect the automatic principle of multiple transformation of information due to the return to the starting position. In this connection, the polygram must have a path from the starting point through a sequence of points to the starting point. In this regard, it is expedient to first describe the point-by-point cycle with the initial point (state), and then the intermediate cycles of the polygram.

9. It is necessary to check the timing of the processes occurring in the controlled object and in the machine. If there is a disagreement during the process, it is necessary to correct the polygram. After writing the polygraph and checking it for the

time alignment of the processes occurring in the controlled object and in the control automaton, it is considered that the IA  $A$  task is completed with the polygram.

### **3.5. Automaton of the 4th kind that controls the operability of basic memory circuits**

All abstract automata from the 1st to 3rd genus consider only the domain of their functioning in automatic discrete time or in auto-matem continuous time. In their description they do not consider the possibility of testing memory circuits for operability [2-4; 6-14].

The issues of checking the efficiency of memory circuits are carried out by circuitry in the presence of three identical devices that give out a signal through the comparison circuit that the circuit is operating correctly or not correctly [15-16].

These checks do not reveal the operability of individual basic circuits in the event of catastrophic failures on which the device is built.

In case of catastrophic failures of basic schemes, it is possible to construct a new automaton of the 4th kind, which would give a signal for checking catastrophic failures in the basic memory circuits, rather than making a transition from one state to another.

At present, large integrated circuits (LSIs) are built with the expectation of 100% efficiency of all components of the circuit. Increasing the number of components and reducing their size increases the probability of getting out of the component health and the appearance of breaks in their connections. This leads to a significant marriage in the construction of LSI and a catastrophic failure of them in the process of exploitation.

In order to increase the reliability of the system from unreliable electronic components, multiple redundancy, distribution of network systems, etc., in which the failure of the whole unit or device is damaged is determined by the output signal of the comparison schemes of several blocks [17].

Unfortunately, the definition of catastrophic failures in memory circuits in known works is not considered, much less their detection.

There are two types of catastrophic failures in basic memory circuits. The first type in the basic memory circuits on the OR-NOT (AND-NOT) logic elements is performed when an unchanged active output signal appears on at least one of the output nodes of the logical elements whose value is equal to logical 1 (0). The second type in the basic memory circuits on the OR-NOT (AND-NOT) logic elements is performed when the passive signals are unchanged at the same time on all output nodes of the OR-NOT (AND-NOT) logical elements whose value is equal to the logical 0 (1). The first type of catastrophic failure is an active constant output signal at the output node of the logical element, which, through circuits to the inputs of logical elements of other groups to the basic memory scheme, makes the memory circuit completely inoperative. The memory scheme for the second type of catastrophic failures becomes the same, when the values are passive at all its output nodes of the basic memory scheme. This state of the memory circuit is not stored for any one retaining the  $e(\Delta)$  input signal. With responsible control devices in such areas as: nuclear power plant, any mode of transport, etc. catastrophic uncontrolled failures are unacceptable, since the components of devices and LSIs must operate reliably. Otherwise, it leads to great catastrophes [18-19].

Therefore, the emergence of methods for detecting catastrophic failures in basic memory circuits at the level of the theory of automata seems to the authors to be an actual topic. All known abstract and structural automata from system positions that consider the laws of operation of any device with a memory are classified by the type of the output signal [2; 8].

For all of them, working in deterministic mode, there is one input signal that establishes  $x_p(t)$ , carrying out a probabilistic transition under the action of a preserving  $e(\Delta)$  input signal to a certain block of  $\pi_j$  states. The signal  $x_p(t)$  has for the base circuit at all its incoming nodes the active value equal to the logical 1 (0). This  $x_p(t)$  input signal sets the output nodes of the OR-NOT (AND-NOT) gates to passive values equal to the logical 0 (1). This state  $a_p(t)$  of the output signals is not conserved at

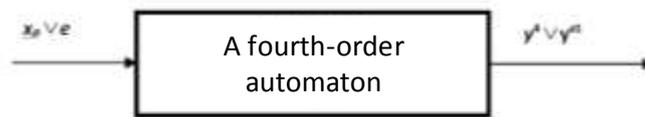
the end of the signal setting  $x_p(t)$  and a probable transition to the state  $a(\Delta)$  is created, which is then stored in a certain set of  $\pi_j$  states of the memory circuit. Therefore, the input signal that sets  $x_p(t)$  is not used in deterministic automata and is forbidden [1-4].

But, this input signal that establishes  $x_p(t)$  can be used in all of the considered 1st, 2nd and 3rd kind automata to determine the output signals of catastrophic failures of the first type, when  $y^4(t) = \lambda_4(x_p(t), a_p(t))$ , and use the output signal of the probabilistic automaton of the third kind  $y^{61}(\Delta) = \lambda_3(a_p(\Delta), e(\Delta))$ , to determine the output signals of catastrophic failures of the second type. The output signal  $y^4(t)$  of the automaton of the 4-th kind differs from the output signal  $y_1(t)$  of the 1-st kind automaton in that it uses the state  $a(\Delta-1)$  in the output function, and the state  $a_p(t)$ , by the input signal  $x_p(t)$ .

Thus, an automaton using the output signal  $y_4(t)$  can be called an automaton of the fourth kind due to the fact that all automata are determined by their output signals. In fact, the output signal  $y^4(t)$  can additionally be added to the output signals of any automata with memory in the basic memory circuits. The output signal  $y^{63}(\Delta)$  of the automaton of the third kind appears after the probabilistic transition of the memory circuit to one of the remembered states of the basic memory circuit under the action of the retaining  $e(\Delta)$  input signal.

These output signals  $y^4(t)$  and  $y^{61}(\Delta)$  can be used to test the performance of memory elements in any devices to detect their catastrophic failures. This is explained by the fact that in case of catastrophic failures of the first type in the automatic memory circuit, when at least one logical element in the NOR memory has a constant output signal equal to the logical 0 (1), then the output signal  $y^4(t)$  changes its value, which determines the failure of the memory capacity to the opposite value of 1 (0). Catastrophic failures of the second type are also formed in the absence of a logic 1 at all output nodes of the circuit, and when input to the input nodes preserves the  $e(\Delta)$  input signal.

The structural diagram of the automaton of the fourth kind is shown in Fig. 3.1.



**Fig. 3.1.** Structural diagram of an automaton of the 4th kind

Thus, we briefly consider the multifunctional abstract automata of the 1st, 2nd, and 3rd kinds and proposed for use in each of these automata an  $x_p(t)$  input signal for determining catastrophic failures in the memory circuits of these automata. Of course, the detection of catastrophic failure in memory circuits does not solve all the problems of checking the efficiency of memory circuits, but it helps to identify the most significant failures - catastrophic, in which the memory circuit is no longer able to work as a storage device.

### **3.6. Conclusion to the third chapter**

In the third chapter, we consider ways to define deterministic multifunctional abstract automata in the form of tables and transition graphs, as well as tabular methods for specifying abstract probability automata.

A method for specifying a hierarchical abstract automaton with a multifunctional memory organization using a polygram is considered, and a method for formulating the polygram is given. The monitoring performance of the basic memory circuits in the automata of the fourth kind is considered.

### **3.7. CONCLUSION TO SECTION 1**

In chapter 1 of the handbook, multifunctional abstract automata capable of processing hierarchical information in automatic continuous time are considered, which is fundamentally different from successive Mealy and Moore automata processing sequential information in automatic discrete time. It is noted that the Mealy and Moore automata are a special case of multifunctional automata of the first and second kinds of Marakhovskiy. In the multifunctional automata of Marakhovskiy, automata of the third kind are considered that process information on two variables  $x(t)$  and

$e(\Delta)$  in automatic continuous time, which in principle distinguishes them from the Mealy, Moore and C automata machines that process information one at a time variable  $x(t)$  in the automaton discrete time. Deterministic, probabilistic and fuzzy multifunctional automata of the third kind are considered and their classification is given. In the second chapter, multilevel abstract automata are considered. On the basis of the proposed principle of hierarchical program management, which allows the simultaneous processing of information in multi-level abstract automata from the principle of program management used in modern computers, which allows to increase the tunability of the work algorithm, increase the processing speed and improve the reliability of information processing. For the first time, the possibilities of monitoring catastrophic failures in memory circuits based on the proposed automaton of the 4th kind are considered. In the third chapter of the first section of the handbook, we consider methods for specifying automata, multilevel abstract automata, which are based on a controlled multifunctional automaton  $A_y$ , controlled by the automaton of the  $A_M$  strategy, which in turn can also be multilevel. The sense of probabilistic and fuzzy transitions in real devices of computer technology, ways of setting multifunctional and multi-level automata are considered. A description of multi-level automata by a polyprogram is proposed, which differs fundamentally from the microprogram and autogram used in the development of control devices with memory on the triggers. An automaton of the fourth kind is described that monitors the operability of basic computer memory circuits.

## SECTION 2

# STRUCTURAL ORGANIZATION OF MULTIFUNCTIONAL AND MULTI-LEVEL MEMORY SCHEMES

## Chapter 4. Multifunctional memory circuits

### 4.1. Basic concepts

In the whole world the search of new circuit technology decisions of elementary memory circuits proceeds with qualitatively new properties. Unfortunately, all examined works on the reconstructed memory circuits were conducted due to the reconstructed functions of excitation and exits, in basis of that there is *RS*- Flip-flop, that have fundamental limitations [1]:

1. all of them work in automat discrete time of  $t_i (i= 1, 2. ., n, .)$ ;
2. the base chart of memory(*RS*-Flip-flop) for known Flip-flop does not allow to reconstruct work of the memorized states;
3. all these devices are described by the automats of Mealy and Moore, that determine successive character of work of devices;
4. passing to the charts of memory takes place on one variable of  $x(t)$ ;
5. the used principle of programmatic management, offered Charlz Bebbidzhem, does not allow simultaneous treatment of general and private information and determines breaking up of information only on two.

In connection with these fundamental limitations of the class of monofunctional memory circuits (flip-flops), multifunctional memory schemes are proposed based on new principles and methods of structural organization.

The principle of the structured organization of multifunctional memory circuits (MFIS) is that  $n$  logical elements OR-NOT (AND-NOT) are used, which are divided into  $m$  ( $m < n$ ) groups. The outputs of the elements of one group are not associated with the inputs of their group of logical elements. They are connected to the inputs of

the elements of other groups of the memory scheme according to one of the defined laws (for example, with the inputs of all other logical elements or the inputs of only elements of two, three, etc. of  $m$  groups of elements). One of the free inputs of each  $i$ -th element is connected to the inputs of the setting input bus, and the second of the free inputs of each  $i$ -th element is connected to the inputs of the memory bus that stores the input bus. The introduction of an additional saving input bus in the memory scheme and the creation of groups of logical elements in a group of more than one was a fundamentally new phenomenon in the development of multifunctional memory circuits.

The principle of memorizing states in an MFIS is that the input signals that establish  $x_i(t)$  that arrive at the nodes of the setting input bus uniquely determine the output active values of at least one logical element of the  $i$ -th group.

The output active values of the installed element through their output structural bonds keep the output values of other elements of the memory circuit inverted, which in turn through the inverse structural connections confirm the established output values of the logic elements when one of the conserved sets of input signals supplied by the conserving input bus. These set values of the state of the memory circuit  $a(\Delta)$  are stored due to the state preservation functions  $\delta_e$  until the next input  $x_i(t)$  input signal arrives [2].

The use of state conservation functions  $\delta_e$  in monofunctional Miley and Moore automata (also in memory circuits (triggers)) was not considered, because all triggers kept their states only with one passive input signal  $e(\Delta)$ , which was called an "empty word of zero length" although this function physically existed. In automatic discrete time, there was no length (interval  $\Delta$ ) for it, and under its influence no single transition from one state to another was realized. But this input signal  $e(\Delta)$  has always been taken into account by the developers of memory circuits.

*Definition 4.1.* The MFIS will be called a single-level multifunctional elementary automaton (MEA) with a complete transition system and a complete output system for each of the  $r_e$  ( $r_e > 1$ ) state conservation functions  $\delta_e$ .

MEAs can be functionally represented as re single-level elementary automata, each of which remembers all its states only with one of the various corresponding sets of input signals that preserve  $e_j(\Delta)$  ( $j = \overline{1, r_e}$ ) [3].

## 4.2. Method of microstructural synthesis of elementary multifunctional memory circuits

Consider a method of microstructural synthesis, which allows us to construct an asynchronous MFIS class  $L$  from the logical elements of a functionally complete system.

We use the combination scheme OR-NOT, which realizes this function:

$$y = \overline{f(a)} \vee \overline{f(x)} \vee \overline{f(e)}, \quad (4.1)$$

where  $f(a)$  – is a function of an arbitrary input signal coming from the output of an element of another group for storing the state in the MFIS ;

$f(x)$  – is a function of an arbitrary set of input signal  $x_i(t)$ ;

$f(e)$  – is a function of an arbitrary saving  $e_j(\Delta)$  set of the input signal.

The combination scheme (4.1.) is called a basic automaton with one state or simpler than a basic automaton (BA). The simplest BA are logical elements of the AND-OR-NOT, OR-NOT or NAND type.

The method of microstructural synthesis of asynchronous MFIS of class  $L$  consists of the following algorithm. We take  $n$  BA and divide them into  $m$  ( $m < n$ ) groups. The BA in each  $i$ -th group ( $i = 1, 2, \dots, m$ ) has no feedback, because their output nodes do not join the input nodes of the BA of this  $i$ -th group. The outputs of the BA of the  $i$ -th group are respectively connected directly or through the OR (AND) separation scheme or directly to the inputs  $f(a)$  of all BAs of the other groups.

One of the free inputs  $z_i$  of each  $i$ -th BA is connected to the inputs of the setting input bus IIX, and the second of the free inputs  $u_i$  of each  $i$ -th BA is connected to the

inputs storing the input bus IIIE of the memory circuit. Input nodes  $z_i$  can receive input  $x_i(t)$  input signals, and input  $e_i(\Delta)$  inputs can be input sequentially one by one for one clock cycle  $T_i$  ( $T_i = t_i + \Delta_i$ ). Stable output signals at the output nodes  $y_j$  of the BA correspond to the shifted states  $a_j(T)$  of the MFIS, where  $a_j(T) = a_j(t) + a_i(\Delta)$ . The shift of the output signal  $y_j(T)$  is equal to the delay of two logic elements, which is necessary to establish a stable output state in the memory circuit.

The  $x(t)$  input signals of the MFIS unambiguously establish a certain state  $a_j(t)$  of the memory circuit. The excitation function  $\delta x$  in an elementary automaton can be described in vector form:

$$a_i(t) = \delta_x[x(t)]. \quad (4.2)$$

The value of the binary set at the input nodes  $z_j$  of the multifunctional memory scheme, under the action of the input signal setting  $x(t)$ , is characterized by the fact that only at the input nodes of the logical elements ( $BA_j$ ) of one  $i$ -th group, the input signal can have a value equal to the passive signal 0 (1) On at least one logical element ( $BA_j$ ) of this  $i$ -th group. At the input nodes of logic elements ( $BA_j$ ) of other groups, the values of the input signal must be equal to the active signal 1 (0). The MFIS stored signal  $e_j(\Delta)$  can memorize one of the states  $a_j(\Delta)$  defined by the state block  $\pi_j$ , predetermined by the setting input signal  $x_i(t)$ . The function  $\delta_e$  of conservation of a state in an elementary automaton can be described in vector form:

$$a(T) = \delta_e[a(T), e(\Delta)], \quad a(t) = a(\Delta). \quad (4.3)$$

The value of the binary set at the input nodes  $u_j$  of the MFIS under the action of the saving  $e(\Delta)$  input signal is characterized by the fact that at least two MFIS groups at the input nodes of the BA have input signals whose value is equal to the passive signal 0 (1). The number  $K$  of storage states for a given  $e(\Delta)$  input signal is equal to

the number of groups in the MFIS, at the inputs of which the value of the input signal at node  $u_j$  is equal to the passive signal 0 (1). Thus, the value of the number  $K$  can vary depending on the input signals  $e(\Delta)$  saving from 2 to  $m$ . The state  $a_i$  of the MFIS is identified with the state of the values of the output signals  $y_j$  BA $_j$  of only one group if at least one of the output signals  $y_j$  of this group is equal to the active signal 1 (0). The unit value of the output signal  $y_j$  in this group is called the MFIS, because this output signal  $y_j$  acts on the BA $_j$  of other groups, setting on them output signals  $y_j$  equal to the logical zero. The characteristic number  $K_i$  of the storage states of the  $i$ -th group is calculated by the formula:

$$K_i = 2^R - 1, \quad (4.4)$$

where  $R$  is the BA number in the  $i$ -th group of the MFIS.

The number  $K_i$  of the storage states of the  $i$ -th group of the MFIS is a block of  $\mu_i$  states. The transition from the state  $a_k$  to the state  $a_s$  in the block of  $\mu_i$  states is possible with a change in the  $e(\Delta)$  input signal. Such a transition is called enlarged. The function  $\delta_y$  of the coarse transition can be represented in vector form:

$$a(\Delta) = \delta_y[a(t), e(\Delta)]; \quad a(t) \neq a(\Delta); \quad a(t), a(\Delta) \in \mu_i. \quad (4.5)$$

Thus, the number of memory states of an MFIS can be represented in a matrix form (Table 1.1), where the rows of the matrix determine the blocks of  $\pi_j$  states that are remembered for the corresponding input signals retaining  $e_j(\Delta)$ , and the matrix columns are the states  $\mu_i$  blocks that are set by the corresponding setting  $x_i(t)$  input signals. A single-valued transition from the state  $a_i$  in the state  $a_k$  of the  $\pi_j$  state block (in the matrix row) is performed under the influence of the input signals establishing the  $x_i(t)$  (Fig. 3.3), and the enlarged transition to

the new state in the  $\mu_i$  state block - under the influence of the preserving  $e(\Delta)$  Input signals (Fig. 3.4) trigger circuits and SMEs have only one block of  $\pi$  states. This characteristic indicates that monofunctional memory circuits (triggers and SMBs) is a particular case of the MFIS. Consider tables of tasks for monofunctional (Table 4.1) and multifunctional (Table 4.2) memory schemes, which supports this conclusion.

Table 4.1

Setting the block n states of monofunctional memory circuits

$e$	$x_1$	$x_1$	$x_2$	...	$x_K$
$e_0$		$A_1$	$A_2$	...	$A_K$

The function  $\varphi$  of the outputs in the MFIS depends on the setting input word  $p = x, e$ , which establishes and stores the states  $a$ . In this case, two cases are possible: when the function  $\delta_e$  of conservation of state (4.3) or the function  $\delta_y$  of the coarse transition (4.5) is realized.

The function  $\varphi_1$  of the outputs, which depends on the state of the automaton  $a(\Delta-1)$  and sets the input signal  $x(t)$ , characterizes the first-order automaton and in the vector form has the form:

$$y(t) = \varphi_1 [a(\Delta-1), x(t)]. \quad (4.6)$$

Table 4.2

Specifying the state matrix of multifunctional memory circuits

$e_j$	$x_1$	$x_1$	$x_2$	...	$x_K$
$e_1$		$A_{11}$	$A_{21}$	...	$A_{K1}$
$e_2$		$A_{12}$	$A_{22}$	...	$A_{K2}$
...		...	...	...	...
$e_m$		$A_{1m}$	$A_{2m}$	...	$A_{Km}$

The function  $\varphi_2$  of the outputs depends on the state  $a(t)$  of the state and on the stored  $a(\Delta)$  state, which characterizes the automaton of the second kind and in the vector form has the form:

$$y(T) = \varphi_2 [a(t), a(\Delta)]; a(t) = a(\Delta), \quad (4.7)$$

or

$$y(T) = \varphi_2 [a(T)], a(T) = a(t) \cup a(\Delta). \quad (4.8)$$

The function  $\varphi_2$  outputs ensures that the set state  $a(T)$  is maintained for a continuous cycle time  $T$ .

The function  $\varphi_3$  of the outputs in the MFIS depends on the set  $a(t)$  state and on the stored input signal  $e(\Delta)$  and which can be represented in vector form as follows:

$$y(\Delta) = \varphi_3 [a(t), e(\Delta)]. \quad (4.9)$$

The function  $\varphi_3$  outputs characterizes automata of the third kind, which determines the direction of the output signal  $y(\Delta)$  as a function of the saving signal  $e(\Delta)$ .

### **4.3. Symbolical description of elementary multifunctional memory circuits**

Symbolic language for the description of elementary memory circuits (triggers, MFIS), through which it is possible to determine in formal ways their basic characteristics: the number of logical elements used to set and store input signals. Knowing the basic characteristics (parameters) of the elementary memory scheme, the designer can intelligently choose the elementary memory scheme necessary for the logical design of a complex device.

The most famous binary elementary memory circuits are triggers based on the *RS*-trigger [4-6], which is a particular case of the MFIS [2]. Such schemes are characterized by three main parameters:  $M$  - the number of stable states  $a$ , each of which

corresponds to a certain output signal of the memory circuit  $y_2(T)$ ;  $r_x$  – is the number of setting input signals  $x(t)$  and  $r_e$  – is the number of conserved input signals  $e(\Delta)$ , which are formally associated with the MFIS structure. The basic MFIS structures created on NAND or NOR logical elements are called basic automata (BA), and their parameters are determined by the proposed formulas.

Memory circuits consist of groups BA (elements), the groups of which are interconnected by feedback circuits, and the characteristic number of memory states  $K_i$  in the  $i$ -th group is determined by the formula (4.4). Thus, the number  $M$  of stable states  $a(\Delta)$  of the MFIS stored under the influence of the input signals retaining  $e(\Delta)$  is determined by the formula:

$$M = \sum_{i=1}^m K_i, \quad (4.10)$$

where  $K_i$  – is the characteristic number of states in the  $i$ th group of the MFIS.

The total number  $r_x$  of different sets of  $x(t)$  input signals set by the MFIS is given by:

$$r_x = M + 1, \quad (4.11)$$

where  $M$  – is the number of stable states of the MFIS, which are conserved;

1 – is an additional set of the input signal that sets  $x_p(t)$ , which uniquely establishes the state  $a_p(t)$ , which is not conserved for any set of the MFIS stored  $e(\Delta)$  input signal. Such a set of  $x_p(t)$  input signal in deterministic devices is forbidden [5].

The number of different  $r_k$  sets preserving  $e(\Delta)$  input signals that can store different groups of  $k$  ( $2 \leq k \leq m$ ) states in the MFIS is determined by the formula:

$$r_k = \sum_{j=1}^k C_m^j \left( \prod_{i=1}^k (2^{R_i} - 1) \right), \quad (4.12)$$

where  $C_m^k$  – is the number of combinations from  $m$  to  $k$ ;

$M$  - is the number of groups of basic automata (BA) in the MFIS;

$R_i$  - is the amount of asthma in the  $i$ -th group of the MFIS.

The total number  $r_e$  of different sets of  $e(\Delta)$  inputs that are saved in the MFIS is determined by the formula:

$$r_e = \sum_{k=2}^m r_k . \quad (4.13)$$

More simply, the total number  $r_e$  of different sets of  $e(\Delta)$  inputs that are stored in the MFIS can be defined by the formula:

$$r_e = \prod_{i=1}^m K_i \quad (4.14)$$

The structure of an MFIS can be described by a symbol number. The symbolic description of the elementary memory device allows representing the structure of the MFIS in the form of a positional number (decimal or hexadecimal). This number should characterize the structure so that, using a number, it would be possible to formally calculate the main parameters of the memory scheme, on the basis of which to choose the optimal structure in the designer's view and build it on logical elements in the form of a functional memory scheme [7].

In the symbolic description of the MFIS, it is advisable to enter a multi-digit decimal number whose number of bits corresponds to the number of logical element groups in the MFIS, and each digit is the number of logical elements in a particular group. The maximum number of decimal places will be 10, which corresponds to the restriction of the number of groups to 10 in the structure of the MFIS. These restrictions are purely conditional, although they correspond to some extent to the restriction of many basic elements of OR-NOT (AND-NOT) integrated circuits [8].

For example, the *RS*-trigger in symbolic form can be described by the number 11, which indicates that the trigger has two groups of logical elements NOR (NAND), one logical element in each group [4]. From the symbolic description of the *RS*-trigger it is possible to determine its basic parameters from the above formulas (4.10) - (4.14) and determine their values:  $M = 2$  (according to the formula 4.10);  $r_x = 3$  (according to the formula 4.11);  $r_e = 1$  (according to the formula 4.14). It should be noted that with the set of input signals  $x(t)$  in deterministic devices, one input signal  $x_p(t)$  is forbidden, since it establishes a state that is not conserved for any conserved input signal  $e(\Delta)$ . In the *RS*-trigger, the  $e(\Delta)$  input signal ( $R = S = 0$ ) that stores the eavesdropper only stores one of the two stable states ( $Q = 1$  or  $Q = 0$ ) [4-5].

A symbolic multi-digit number that defines the structure of an MFIS in decimal notation has the limitations of the number of NAND gates in each group up to 9, which corresponds to the real limitation of integrated circuits [8]. The number of digits that characterize the structure of the MFIS in decimal notation has limitations on the number of possible inputs in the used logical elements (up to 10 in this symbolic description). In the symbolic description, the structure of the MFIS is given by the number of logical elements in each  $i$ -th position of the number and the number of groups (digits) in the number itself.

#### **4.4. Possible variants of multifunctional memory circuits in the symbolic number**

The MFIS is designed with a certain number of logical elements  $n$  ( $n \geq 2$ ), divided into  $m$  ( $m \leq n$ ) groups. The number of variants of the structural solutions of the MFIS from  $n$  logical elements grows rapidly with the corresponding increase in the number of logical elements.

For example:

- If  $n = 2$ , the symbolic description of the MFIS structure is described by only one number 11 (*RS*-type trigger);

- If  $n = 3$ , there are two symbolic descriptions of the structure of the MFIS : 12, 111 (entries 12 and 21 are the same when constructing the MFIS structure);
- If  $n = 4$ , there are four options for describing the MFIS : 22, 13, 112, 1111;
- If  $n = 5$ , there are six options for describing the MFIS : 14, 23, 113, 122, 1112, 11111 and so on.

The MFIS structure is selected from the necessary parameters  $M, r_x, r_e$  by the developer himself at the level of the symbolic description of the structure of the MFIS .

If, in selecting the structure of the MFIS for the criterion to take only one of the main parameters of  $M$ , and the number of  $n$  logical elements used in the MFIS structure, then  $n$  is within the following limits:

$$\log_2 M \leq n \leq M \quad (4.15)$$

The choice of the MFIS structure, when defining the main parameters is quite formalized, and it can be implemented on modern computers [5].

To obtain the results that are obtained by using the symbolic description when selecting the MFIS, you can perform the following steps:

1. In the symbolic description of the MFIS, its main parameters can be determined:  $m$  – is the number of stable states to be stored;  $r_x$  – is the number of input signals that set  $x_i(t)$  and  $r_e$  – is the number of input signals saving  $e_j(\Delta)$ , and also select the necessary criterion (satisfying the received results of the main parameters) for the logical design of the MFIS , which is necessary for creating perspective reconfigurable devices of computers and Networks.
2. Choosing the necessary MFIS , taking into account the basic parameters, it is possible to construct its functional scheme based on the logical elements (AND-NOT, OR-NOT, AND-OR-NOT), and also to find its description in the form of a logical equation system, memory scheme modeling
3. We find a description in the form of a system of logical equations of the memory scheme. With the help of analysis methods, we determine its parame-

ters: sets of setting  $x_i(t)$  input signals and sets of saving  $e_j(\Delta)$  input signals that preserve the established states or change the structure of their memorization. The sets of setting  $x_i(t)$  input signals respectively set the states  $a_i(t)$  in the MFIS and, upon further arrival of the sets of  $e_j(\Delta)$  input signals, store these states  $a_i(\Delta)$  in certain blocks  $\pi_j$  ( $a_i(t) = a_i(\Delta) \in \pi_j$ ) states of the MFIS or performs the enlarged transitions to the specific states of the blocks  $\pi_j$  ( $a_i(t) \neq a_k(\Delta); a_i(t) \in \pi_i; a_k(\Delta) \in \pi_j$ ) at the internal clock cycle  $\Delta$  for one machine cycle  $T$ .

Thus, it becomes clear that two sets of input signals are required for the operation of multifunctional circuits: setting and saving  $e_j(\Delta)$ , which enter one clock cycle of  $T$  ( $T = t + \Delta$ ). A feature of these two sets of input signals is that the setting  $x_i(t)$  absorbs the preserving  $e_j(\Delta)$  if they arrive at the same hour.

$$x_i(t) = x_i(t) \cup e_j(t) \quad (4.16)$$

This is a very important concept, which we will follow in the future when considering the operation of memory circuits.

#### **4.5. Synthesis of multifunctional memory circuits by symbolic description**

The symbolic description of the MFIS in the form of a decimal number contains all the necessary elements for the structural synthesis of an asynchronous MFIS. This description discusses information about the number of elements used, the number of groups to which logical elements are divided, and the number of logical elements in these groups. However, in this description, there is no information on which logical elements the MFIS will implement and in which class of memory -  $L$  or  $L^M$ . The choice of logical elements and the class of memory schemes is the business of the MFIS developer himself.

The choice of the types of logical elements is determined by the number of inputs in the logical element, the speed of the logical elements, their power consump-

tion or the need to use a logical unit (or incoming) in the element as a logical unit or zero.

After selecting the type of logical elements with certain characteristics, you can define the class of the MFIS itself. Multifunctional memory circuits of class  $L$  have more than the MFIS class  $L^M$ , the number of inputs in the logical elements, but have greater speed and less hardware costs due to the absence of logical OR elements that in the  $L^M$  class participate in the connections between groups, their number does not exceed the number groups in the MFIS.

For MFIS class  $L$  microstructural synthesis in the symbolic description is as follows.

We take such a number of logical elements or AND-NOT, or OR-NOT, or AND-OR-NOT, which is equal to the sum of the digits in the numerical symbolic description of the MFIS. We divide these logical elements into so many groups that they are equal to the number of digits in the numerical symbolic description of the MFIS, and in each group we take the number of logical elements to be equal to the value of the corresponding digit in the decimal number of the symbolic description of the MFIS.

Logical elements of one group are connected by their output nodes to the input nodes of all logical elements of other groups. Other inputs of logic elements (at least two) that connect to the corresponding input MFIS buses are used for setting and storing input signals. The output nodes of the logic elements are connected to the output line of the MFIS.

For MFIS of the  $L^M$  class, microstructural synthesis in the symbolic description provides the following.

We take such a number of logical elements or AND-NOT, or OR-NOT, or AND-OR-NOT, which is equal to the sum of the digits in the numerical character set of the MFIS, and the number of AND or OR gates that is equal to the number of bits in the MFIS, where the figure has a value greater than one.

Divide these logical elements (AND-NOT, OR-NOT, or AND-OR-NOT) into as many groups as possible to equal the number of digits in the numeric character set of

the MFIS. In each group, we take the number of logical elements that is equal to the value of the corresponding digit in the decimal number of the symbolic description of the MFIS, as well as the number of AND or OR gates that is equal to the number of digits in the MFIS character set where the number is greater than one.

Divide these logical elements (AND-NOT, OR-NOT, or AND-OR-NOT) into as many groups as possible to equal the number of digits in the numeric character set of the MFIS. In each group, we take the number of logical elements that is equal to the value of the corresponding digit in the decimal number of the symbolic description of the MFIS as well as the number of AND or OR gates that is equal to the number of digits in the MFIS character set where the number is greater than one.

Logical elements of one group are linked by their output nodes (when the digit in the digit is equal to one) or through the logical element AND or OR (when the digit in the digit is greater than one) with input nodes of all logical elements of other groups. Other inputs of logic elements (at least two) that connect to the corresponding input MFIS buses are used for setting and storing input signals. The output nodes of the logical elements (AND-NOT, or OR-NOT, or AND-OR-NOT) are connected to the output MFIS.

Carrying out the microstructural synthesis of the MFIS in the symbolic description of the memory scheme, as in any synthesis of memory circuits, it is necessary to take into account the limitations of the logical elements with the number of admissible input nodes and the number of permissible load capacities of the logical elements used for the design of the MFIS. Let's assume that when constructing an MFIS,  $K$ -input elements AND-NOT, OR-NOT and AND-OR-NOT with the allowed load capacity equal to  $P_1$  are used and  $R$ -input elements AND and OR with the value of the load capacity is  $P_2$ .

For MFIS of class  $L$ , before performing a microstructural synthesis of the functional scheme with its symbolic description, it is necessary to check for the admissibility of such relations:

$$K \geq \sum_{i=2}^m R_i + 2; P_1 \geq \sum_{i=2}^m R_i + 1, \quad (4.17)$$

where  $m$  – is the number of bits in the symbolic number of the MFIS;

$R_i$  - values of the  $i$ -th digit in the symbolic number of the MFIS.

For MFIS class  $L^M$ , before microstructural synthesis, it is necessary to check for the admissibility of such relationships:

$$K \geq m + 1; P_1 \geq 2; R \geq \max(R_i); P_2 \geq \sum_{i=2}^m R_i + 1. \quad (4.18)$$

In the case where the symbol number is 13, to construct an MFIS structure on the AND-NOT, OR-NOT and AND-OR-NOT logic elements with constraints  $K = 5$ ;  $P_1 = 4$ ;  $R = 3$  and  $P_2 = 4$ , the given relationships satisfy the asynchronous memory scheme of the MFIS of class  $L$  and  $L^M$ , which ensures the correctness of the synthesis.

Considering a series of numbers whose sum of digits does not exceed five, and choosing them according to the symbolic description of the MFIS, we can calculate the values of the main parameters from the formulas. Having determined the main parameters of the various MFIS with permissible constraints of the logical elements, it is possible to select the MFIS, which corresponds to the selected parameters. We perform an arbitrary sampling of the structure of the MFIS, the number of states of which is memorized, no less than 6. The closest are the symbol numbers 22 and 13, which satisfy the criteria.

Let us consider an example of the synthesis of the functional scheme of an MFIS on the elements AND-NOT and OR-NOT, which are described by the symbol number 13. First, we use four NAND or NOR elements. Their characteristics have the following meanings:

$$M (\text{number of states that are remembered}) = \sum_{i=1}^m K_i = 1 + 7 = 8;$$

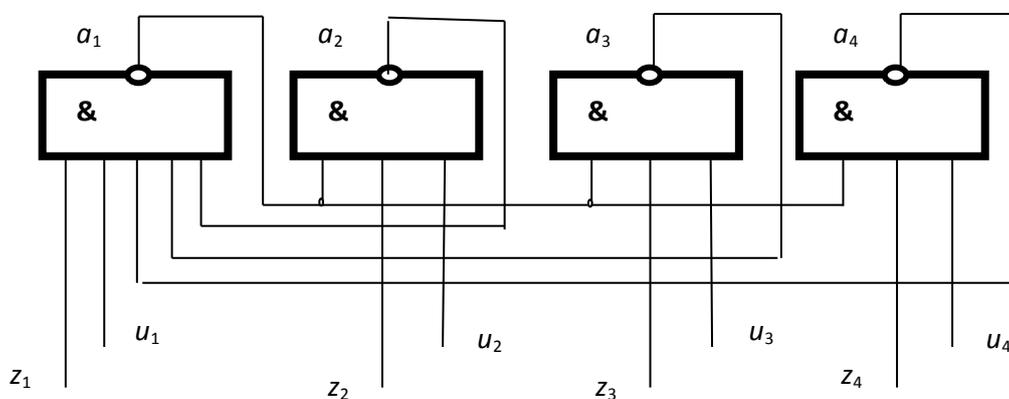
$r_x$  (number of setting input signals) =  $M + 1 = 9$ ;

$r_e$  (the number of input signals retained) =  $\prod_{i=1}^m K_i = 1 \times 7 = 7$ .

First, we divide the elements into two groups: the first will have one element, and in the second group - three. In a group containing more than one element, all output nodes of elements are connected to the input nodes of elements of other groups (in our case, with the input of one element of another group). The output node of the group element consisting of one element is connected to the input nodes of the elements of the group consisting of three elements. The free two input nodes of each MFIS element are connected respectively to the input busbar of the SHX and to the input bus of the SFI of the MFIS storing the states, and the output nodes of all elements with its output bus. MFIS class  $L$ , built on the elements of AND-NOT and OR-NOT, and are shown in Fig. 4.1 and Fig. 4.2.

The MFIS of the  $L^M$  class, built on the NAND and OR-NOT elements (Fig. 4.3 and Fig. 4.4), still has additional elements AND (OR), but has less.

Internal connections, which when used in integrated circuits is very important.



**Fig. 4.1.** MFIS class  $L$  on the elements of the AND-NOT

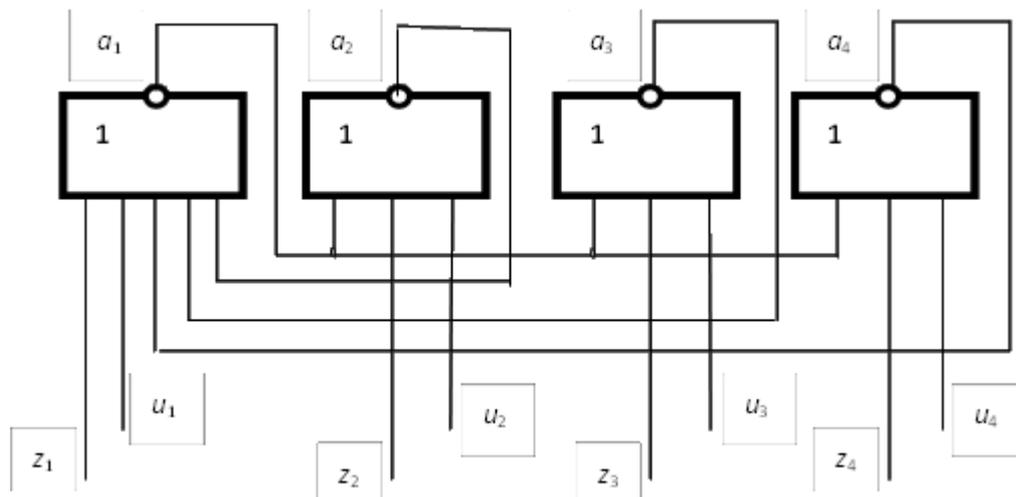


Fig. 4.2. MFIS class  $L$  on the elements of the OR-NOT

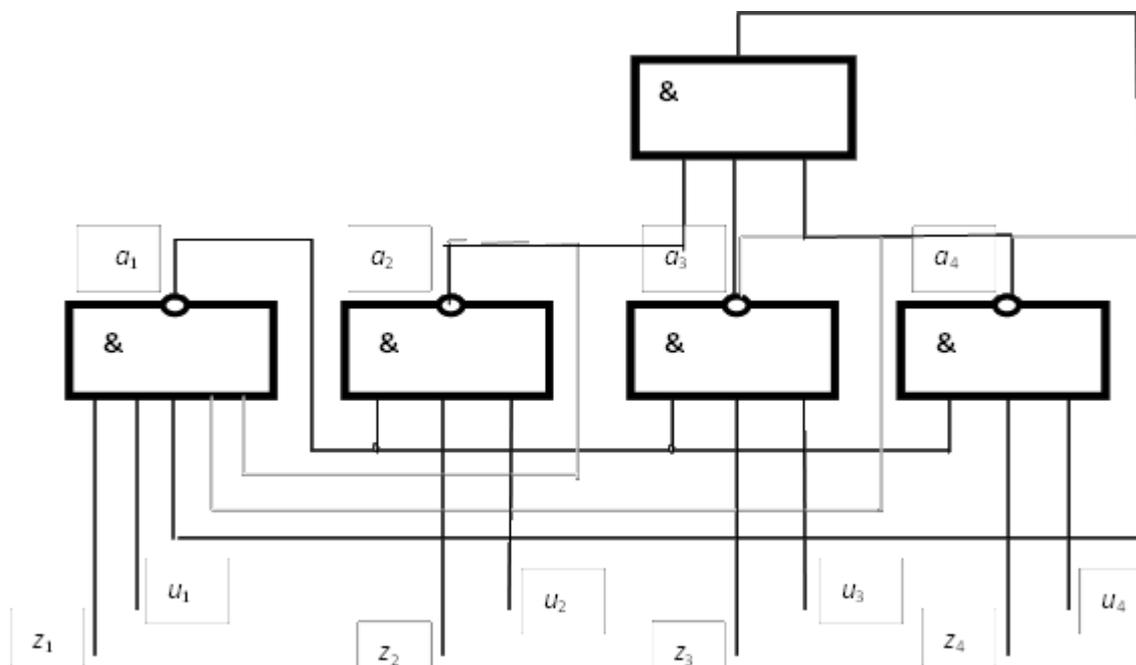
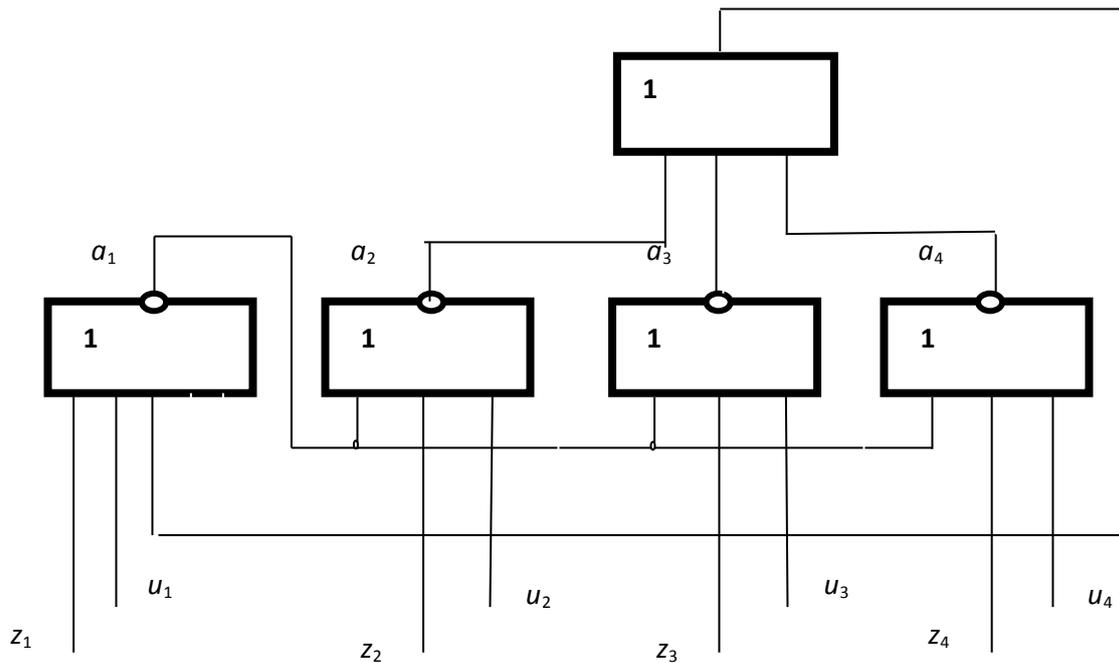


Fig. 4.3. MFIS class  $L^M$  on the elements of the AND-NOT



**Fig. 4.4.** MFIS class  $L^M$  on elements OR-NOT

Thus, having a functional memory circuit with a certain number of elements and using the MFIS analysis technique [57; 64], it is possible to determine the number of setting and saving input signals, the number of states that are stored for certain conserved input signals, and the number of elementary input words: single-valued and enlarged [7].

## **4.6. Definition of input words of circuits of automatic memory**

### **4.6.1. Definition of single-valued elementary input words**

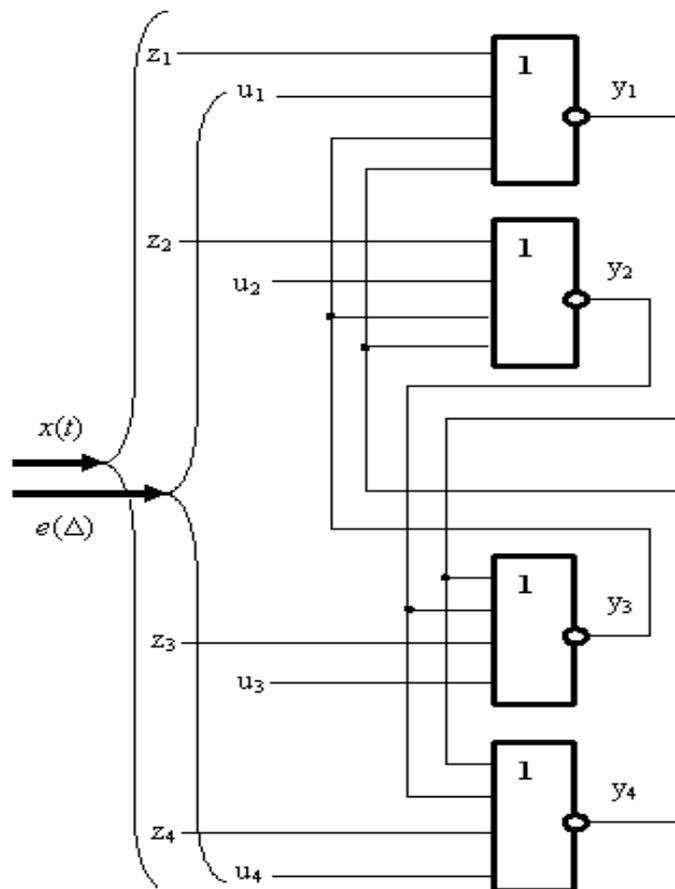
The definition of admissible single-valued elementary input words  $p_0(T)$  of the MFIS consists of the selection of the set of input words  $p_0(T)$  containing pairs of successive sets of input signals that establish  $x_i(t)$ , which uniquely determine the states  $a_i$ , and the sets of  $e_i(\Delta)$  at which the established states are remembered.

First, we determine the set of sets of  $x_i(t)$  input signals that unambiguously set the output signals at the nodes of AND-NOT (OR-NOT) MFIS . A characteristic feature of the sets of setting input  $x_i(t)$  is the presence of unit values of active signals at the input nodes of the AND-NOR gates of all groups except the  $i$ -th.

The number of different sets of  $x_i(t)$  input signals that uniquely determine the states  $a_i(t)$  that are conserved for  $e_j(\Delta)$  sets of input signals in the  $i$ -th group are equal to the characteristic number of the  $K_i$   $i$ -th group, determined by the formula (4.4) [3].

The number of  $r_x$  different sets of setting  $x_i(t)$  input signals that unambiguously set the output signals at the AND-NOT (OR-NOT) MFIS is determined by formula (4.11) [3].

Consider an MFIS of class  $L$  on elements OR-NOT, described by the symbol number 22 (Fig. 4.5).



**Fig. 4.5.** MFIS of Class  $L$

There is a single set of  $x_p(t)$  input signal that has a value of one on all the input nodes of AND-NOT (OR-NOT) in the MFIS and uniquely sets the output signals of all AND-NOT (OR-NOT) in the MFIS in zero during the period of exposure of this  $x_p(t)$  input signal. This  $a_p(t)$  state is not remembered for any set of  $e_j(\Delta)$  input signals. Taking into account these features, we determine the sets of the input signals of the MFIS on the elements OR-NOT (see Fig. 4.5), which are presented in table. 4.3.

By sequentially supplying the sets of input signals that determine the input signals (see Table 4.3) to the input nodes  $z_i$  of the OR-NOT logic elements in the MFIS (Fig. 4.5), we unambiguously determine the states  $a_i$  set on the output nodes of the MFIS (see table 4.4).

Table 4.3

Sets of setting input signals  $x_i(t)$

Input signal $x_i$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$
$z_1$	1	1	1	1	1	0	0
$z_2$	1	1	1	1	0	1	0
$z_3$	1	1	0	0	1	1	1
$z_4$	1	0	1	0	1	1	1

It should be taken into account that at the input nodes  $u_i$  of the OR-NOT logic elements in the MFIS, when the input signals are set to  $x_i(t)$  input nodes  $z_i$ , the values must be equal to logical zero, i.e. A non-active signal for OR-NOR gates. This is determined by the rule that when  $x_i(t)$  and  $e_j(t)$  of the input signals  $e_j(t)$  appear at the same time, (4.16) is absorbed.

Sets saving  $e_j(\Delta)$  of input signals of MFIS are characterized by the fact that at input nodes  $u_i$  of at least two groups there must be at least one passive input signal whose values are equal to logical zero in each of these groups.

Table 4.4

Unambiguous states of the MFIS that are installed  
a set of input signals  $x_i$

Set of input signals $x_i$	$y_1$	$y_2$	$y_3$	$y_4$	States $A_j$
$x_1$	0	0	0	0	$A_0$
$x_2$	0	0	0	1	$A_1$
$x_3$	0	0	1	0	$A_2$
$x_4$	0	0	1	1	$A_3$
$x_5$	0	1	0	0	$A_4$
$x_6$	1	0	0	0	$A_5$
$x_7$	1	1	0	0	$A_6$

The input nodes  $u_i$  of other NOR gates can be supplied with input signals whose value is equal to the active logical unit. In this case, the condition is fulfilled, which means that in the course of the internal cycle  $\Delta$  the input signals at the nodes  $z_i(\Delta)$  are equal to logical zero. The set re of saving sets  $e_j(\Delta)$  of the input signals of the MFIS are determined by the formula (4.14).

We define the set of conserved sets  $e_j(\Delta)$  of input signals of the MFIS with their features taken into account (Table 4.5) [9].

Table 4.5

Sets of saving  $e_j(\Delta)$  input signals

Input signals $u_i$	Sets of saving $e_j(\Delta)$ input signals								
	$e_1$	$e_2$	$e_3$	$e_4$	$e_5$	$e_6$	$e_7$	$e_8$	$e_9$
$u_1$	1	1	1	0	0	0	0	0	0
$u_2$	0	0	0	1	1	1	0	0	0
$u_3$	1	0	0	1	0	0	1	0	0
$u_4$	0	1	0	0	1	0	0	1	0

Exploring Table. 4.5, we come to the conclusion that the set  $A_0$  (see Table 4.4) of the output signals obtained under the influence of the set of the input signal setting  $x_1$  is not remembered for any set of  $e_j(\Delta)$  of the input signals. The set that sets the input signal  $x_1(t)$  upon transition to any set  $e_j(\Delta)$  of the input signals leads to a probabil-

istic transition to a state that is stored in the set of a specific saving set  $e_j(\Delta)$  of input signals.

This is a qualitatively new probabilistic transition that makes the transition to the unknown  $a_i(\Delta)$  state of a certain  $\pi_j$  subset of the MFIS states.

Let us determine the states  $A_j$  that are remembered at the output nodes  $y_i$  of the MFIS with the sets of preserving  $e_j(\Delta)$  input signals of the MFIS (Table 4.6).

Table 4.6

The states of the MFIS, which are remembered for  $e_j(\Delta)$  input signals

The set $e_j(\Delta)$ of the input signals	The states of MFIS that are stored under the influence of sets $e_j(\Delta)$ of input signals
$e_1$	$A_1, A_4$
$e_2$	$A_2, A_4$
$e_3$	$A_3, A_4$
$e_4$	$A_1, A_5$
$e_5$	$A_2, A_5$
$e_6$	$A_3, A_5$
$e_7$	$A_1, A_6$
$e_8$	$A_2, A_6$
$e_9$	$A_3, A_6$

For example, under the influence of the input word  $p(T) = x_l(t)$ ,  $e_9(\Delta)$  of the MFIS fed to the input nodes of the MFIS for one machine clock  $T$ , a definite single-valued subset of the states  $A_3$  or  $A_6$  passes to which state  $A_3$  or  $A_6$ , is unknown. Such a transition is possible only when the MFIS operates in a probabilistic mode. Therefore, the set of  $x_l(t)$  setting input signal of the MFIS, operating in deterministic mode, is forbidden.

This is a qualitatively new theory of constructing automatic memory circuits [3], which allows for transitions in two variables of the input word  $p = x, e$ .

One of the tasks of this section is the development of a methodology for determining the input words of elementary MFIS.

When analyzing the MFIS (Fig. 4.4) from Table. 4.4 and Table. 4.6 select pairs of sets of input signals  $x_i(t)$  and  $e_j(\Delta)$  in such a way that the state  $A_i$ , which is set by the signal  $x_i(t)$  (Table 4.4), is remembered for the signal  $e_j(\Delta)$  (Table 4.6).

We define the number of single-valued elementary input words  $p_0(T)$ . The total number of single-valued elementary input words  $p_0(T)$  can be determined from the formulas:

$$r_{p_k} = k \times r_k, \quad (4.19)$$

$$R_j (j = \overline{1, m}), \quad (4.20)$$

where  $k$  – is the number of  $A_i$  states that are stored for one set of  $e_i(\Delta)$  that stores the input signal;

$r_{p_k}$  – the number of different sets of  $e_i(\Delta)$  preserving input signals that store  $k$  MFIS states;

$m$  – is the number of groups of AND-NOT (OR-NOT) in the MFIS.

The MFIS (Fig. 4.5) has a total number of different sets of  $e_i(\Delta)$  input signals that conserve the  $e_j(\Delta)$  9. For each set of the input-preserving  $e_j(\Delta)$ , one of the two  $A_i$  states can be memorized (Table 4.6). Using formula (4.19), we determine the number of single-valued elementary input words  $p_0(T)$  of the MFIS. Since  $k = 2$  and  $r_k = 9$ , the total number of single-valued elementary input words is 18 ( $p_0(T) = 18$ ). We represent the single-valued elementary input words  $p_0(T)$  of the MFIS in Table. 4.7.

Table 4.7

The single-valued elementary introductory words  $p_0(T)$  MFIS

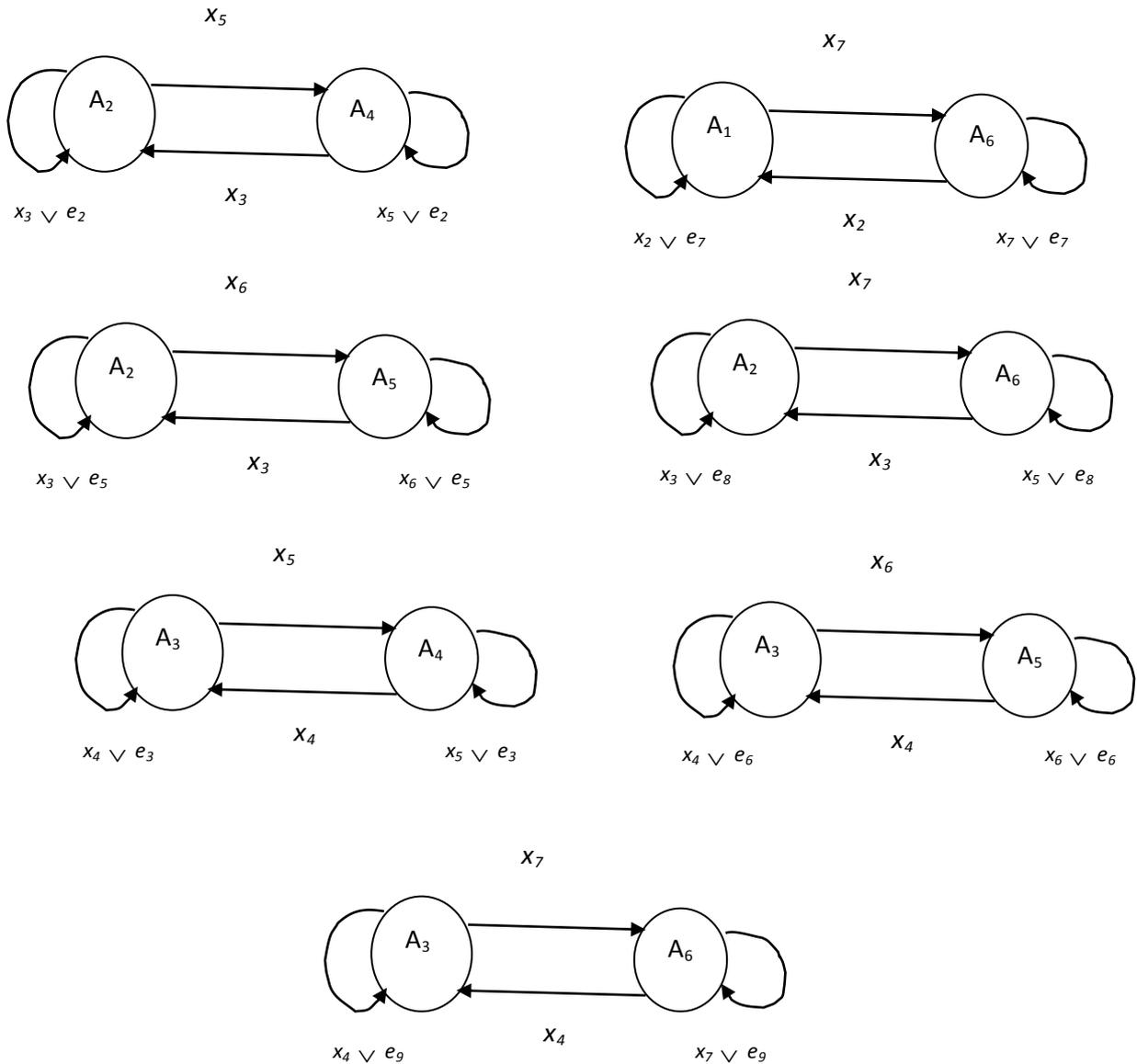
Introductory word $p_0(T)$	Introductory word		Set a condition that remembered
	setting signal	Signal-preserving signal	
$p_1$	$x_2$	$e_1$	$A_1$
$p_2$	$x_5$	$e_1$	$A_4$
$p_3$	$x_5$	$e_2$	$A_4$
$p_4$	$x_3$	$e_2$	$A_2$
$p_5$	$x_5$	$e_3$	$A_4$

$p_6$	$x_4$	$e_3$	$A_3$
$p_7$	$x_6$	$e_4$	$A_5$
$p_8$	$x_2$	$e_4$	$A_1$
$p_9$	$x_6$	$e_5$	$A_5$
$p_{10}$	$x_3$	$e_5$	$A_2$
$p_{11}$	$x_6$	$e_6$	$A_5$
$p_{12}$	$x_4$	$e_6$	$A_3$
$p_{13}$	$x_7$	$e_7$	$A_6$
$p_{14}$	$x_2$	$e_7$	$A_1$
$p_{15}$	$x_7$	$e_8$	$A_6$
$p_{16}$	$x_3$	$e_8$	$A_2$
$p_{17}$	$x_7$	$e_9$	$A_6$
$p_{18}$	$x_4$	$e_9$	$A_3$

The uniquely correct input words  $p_0(T)$  are realized in the course of the clock  $t$  by the excitation function  $\delta_x$ , and in the course of the internal cycle  $\Delta$  is the state conservation function  $\delta_e$ , thereby determining the function  $\varphi_2$  of the MFIS output [3]. The single-valued input words  $p_0(T)$  can translate the MFIS to the state of the block  $\pi_j$  whose states are remembered under the influence of a set of the input  $e_j(\Delta)$  that preserves the input.

The output signals  $y(T)$  of the MFIS have values shifted by a time which is  $2\tau_e$  (the delay of the passage of the input signal through two logic elements), from the beginning of the clock  $t$ , and stable values during the entire external cycle  $T$  of the automatic continuous time [3]. With respect to the switching speed and the characteristics of the output signals  $y(T)$ , the MFIS has the same data as *RS* flip-flops when single-valued elementary  $p_0(T)$  input words arrive on them.

The main difference between MFIS and *RS* flip-flops during their operation is that the MFIS can function in different blocks of  $\pi_j$  of its states, and *RS* flip-flops in only one. For different sets of  $e_j(\Delta)$  input signals stored in the  $\pi_j$  blocks of their states, the MFIS (Table 4.6) can function as 9-x *RS* flip-flops having a different set of their states (Fig. 4.6).



**Fig. 4.6. The MFIS can operate as 9 x RS flip-flops**

### 4.6.2. Definition of enlarged elementary input words

The enlarged  $p_y(T)$  input words differ from the single-valued  $p_0(T)$  input words in that the transitions in them are carried out from one state to another in a certain block  $\mu_i$  of the MFIS states and simultaneously there is a transition from one subset of  $\pi_i$  states to another subset  $\pi_j$  (Table 1.1) [3].

The definition of the permissible elementary enlarged  $p_y(T)$  input words of the MFIS consists of the selection of the set of input words  $p_y(T)$ , which are combined

into pairs of consecutive sets of setting input signals  $x_i(t)$  during the clock  $t$  and sets of  $e_j(\Delta)$   $\Delta$  of the automatic continuous time  $T$  that transfer the MFIS uniquely from the state  $A_i(t)$  to the state  $A_k(\Delta)$  ( $A_i \neq A_k$ ) of the  $\mu_i$  state block ( $A_i, A_k \in \mu_i$ ) during the interval  $\Delta$  between the  $t$ -cycles.

When analyzing the MFIS (Fig. 4.4) Table. 4.4 and Table. 4.6 select the pair of the input signals  $x_i(t)$  and  $e_j(\Delta)$  in such a way that the state  $A_i$ , which is set by the signal  $x_i(t)$  (Table 4.4), having in the first group the active output signals  $y_1$  or  $y_2$ , and the signal  $e_j(\Delta)$  would remember (Table 4.6) another state  $A_k$  ( $A_i \neq A_k$ ) of the  $\mu_i$  state block ( $A_i, A_k \in \mu_i$ ) with the same active output signals  $y_1$  or  $y_2$ .

An example of such an enlarged  $p_y(T)$  of the input word can be a pair of  $x_2, e_2$  input signals for which the signal  $x_2(t)$  uniquely establishes the state  $A_1(t)$  at which the output signals  $y_4=1$  and  $y_3=0$ , and the signal  $e_2(\Delta)$  Transforms the MFIS from the state  $A_1(t)$  to the state  $A_2(\Delta)$  ( $A_1 \neq A_2$ ), at which the output signals  $y_4 = 0$  and  $y_3 = 1$  correspond to the function  $\delta_y$  of the coarse transition [3].

In the set of enlarged  $p_y(T)$  input words, there exists a subset of such words for which the signal  $e_j(\Delta)$  does not change, and the set of the input signal  $x_i(t)$  uniquely determines the active output signal of a certain  $i$ -th group, which then changes during  $\Delta$ . In addition, in a set of enlarged  $p_y(T)$  input words, there exists a subset of such words for which the signal  $e_j(\Delta)$  changes in the absence of a set of input signals that establish  $x_i(t)$ , which leads to a change in the active output signal of a certain  $i$ -th group during  $\Delta$  of the machine clock  $T$ .

The elementary enlarged  $p_y(T)$  input word of the MFIS is unambiguously correct if the set of the input signal  $x_i(t)$  and the ensemble of the input signal that preserves the  $e_j(\Delta)$  during the internal clock cycle  $\Delta$  of the machine clock  $T$  uniquely establishes the state  $A_k(\Delta)$ . The enlarged correct elementary  $p_y(T)$  input word (Table 4.8), which is fed to the input channels (nodes) of the MFIS, is uniquely determined by the function  $\delta_y$  of the junction and determines the function  $\varphi_3$  of the memory scheme output [9].

The number of correct enlarged  $p_y(T)$  input words for the memory scheme shown in Fig. 4.5, can be calculated by the formula:

$$r_{p_y} = (m + 1) \times R_i \times K_i \times (K_i - 1) = 3 \times 2 \times 3 \times 2 = 36 .$$

Enlarged elementary  $p_y(T)$  input words can translate the MFIS into a new state of the  $\mu_i$  state block, which has an active output signal only in a certain  $i$ -th group. The enlarged  $p_y(T)$  input words of the MFIS (Figure 4.5) are listed in Table. 4.8. The number of such words is 36 (20 to 55).

Table 4.8

Enlarged  $p_y(T)$  input words

$p_y(T)$	$x_i(t)$	$e_j(\Delta)$	$A_k(\Delta)$
$p_{20}$	$x_4$	$e_2$	$A_2$
$p_{21}$	$x_4$	$e_5$	$A_2$
$p_{22}$	$x_4$	$e_8$	$A_2$
$p_{23}$	$x_2$	$e_3$	$A_3$
$p_{24}$	$x_2$	$e_6$	$A_3$
$p_{25}$	$x_2$	$e_9$	$A_3$
$p_{26}$	$x_3$	$e_3$	$A_3$
$p_{27}$	$x_3$	$e_6$	$A_3$
$p_{28}$	$x_3$	$e_9$	$A_3$
$p_{29}$	$x_4$	$e_1$	$A_1$
$p_{30}$	$x_4$	$e_4$	$A_1$
$p_{31}$	$x_4$	$e_7$	$A_1$
$p_{32}$	$x_7$	$e_1$	$A_4$
$p_{33}$	$x_7$	$e_2$	$A_4$
$p_{34}$	$x_7$	$e_3$	$A_4$
$p_{35}$	$x_7$	$e_4$	$A_5$
$p_{36}$	$x_7$	$e_5$	$A_5$
$p_{37}$	$x_7$	$e_6$	$A_5$

$p_{38}$	$x_5$	$e_7$	$A_6$
$p_{39}$	$x_5$	$e_8$	$A_6$
$p_{40}$	$x_5$	$e_9$	$A_6$
$p_{41}$	$x_6$	$e_7$	$A_6$
$p_{42}$	$x_6$	$e_8$	$A_6$

$p_{43}$	$x_6$	$e_9$	$A_6$
$p_{44}$	$x_2$	$e_2$	$A_2$
$p_{45}$	$x_2$	$e_5$	$A_2$
$p_{46}$	$x_2$	$e_8$	$A_2$
$p_{47}$	$x_2$	$e_1$	$A_1$
$p_{48}$	$x_2$	$e_4$	$A_1$
$p_{49}$	$x_2$	$e_7$	$A_1$
$p_{50}$	$x_6$	$e_1$	$A_4$
$p_{51}$	$x_6$	$e_2$	$A_4$
$p_{52}$	$x_6$	$e_3$	$A_4$
$p_{53}$	$x_5$	$e_4$	$A_5$
$p_{54}$	$x_5$	$e_5$	$A_5$
$p_{55}$	$x_5$	$e_6$	$A_5$
$p_{43}$	$x_6$	$e_9$	$A_6$
$p_{44}$	$x_2$	$e_2$	$A_2$
$p_{45}$	$x_2$	$e_5$	$A_2$
$p_{46}$	$x_2$	$e_8$	$A_2$
$p_{47}$	$x_2$	$e_1$	$A_1$
$p_{48}$	$x_2$	$e_4$	$A_1$
$p_{49}$	$x_2$	$e_7$	$A_1$
$p_{50}$	$x_6$	$e_1$	$A_4$
$p_{51}$	$x_6$	$e_2$	$A_4$
$p_{52}$	$x_6$	$e_3$	$A_4$

$p_{53}$	$x_5$	$e_4$	$A_5$
$p_{54}$	$x_5$	$e_5$	$A_5$
$p_{55}$	$x_5$	$e_6$	$A_5$

Analyzing the elementary input  $p_0(T)$  and  $p_y(T)$ , we arrive at the following conclusions:

- for the transition of the MFIS from one state  $A_i$  to another  $A_k$ , the single-valued  $p_0(T)$  input word can, during one machine clock  $T$ , make a transition through one variable  $x_i(t)$  of the input signal in one block  $\pi_j (A_i, A_k \in \pi_j)$  of its states, and for triggers;
- for a transition from one state  $A_i$  of the block  $\pi_s (A_i \in \pi_s)$  to another state  $A_k$  of another block  $\pi_j (A_k \in \pi_j)$ , provided that  $A_i, A_k$  belong to the block  $\mu_m$  of their states, a sufficiently correct enlarged elementary  $p_y(T)$  of the input word that is in the state for one machine cycle  $T$  to make the transition with respect to the two variables  $x_i(t)$  and  $e_j(\Delta)$ , which is characteristic only of the MFIS .

The use of such deterministic elementary  $p_0(T)$  and  $p_y(T)$  input words extends the functionality of the triggers. This allows for transitions during one machine clock  $T$  with respect to two variables  $x_i(t)$  and  $e_j(\Delta)$  by input signals, which in principle can not be performed in triggers.

The use of different sets of  $e_j(\Delta)$  preserving input signals allows, in one machine clock  $T$ , to change the structure of the memory states in the MFIS , which is very important in accelerating the rearrangement of information processing algorithms.

#### 4.7. The reliability of multifunctional memory circuits

**Relevance.** A binary memory circuit that has "zero redundancy" and an unchanged structure of functioning is not reliable. This is due to the fact that when a single logic element exits, the basic *RS*-trigger fails. In this regard, it can not effectively serve as an information and arithmetical basis for specialized and reconfigura-

ble computer and measurement systems, as well as for nanoelectronic systems, where problems of reliability, immunity, control, stability, and survivability of systems come to the fore.

It is possible to name the basic requirements for computer devices and control systems used in these technologies, for nuclear power plants, missile technology, aircraft equipment, railway transport, etc., in which "failures" of binary memory lead to great catastrophes. Such requirements may be: reliability and survivability of devices, which are determined when one or more elements fail.

#### **4.7.1. Issues of improving the reliability of memory circuits.**

The increase in the functionality of multifunctional memory circuits is accompanied by an increase in the number of used elements in the  $i$ -th groups, the outputs of the elements of each group of which are connected to the inputs of all elements of the remaining  $(m-1)$  groups (Fig. 4.5). MFIS have two types of input signals: setting  $x(t)$  and preserving  $e(\Delta)$ , arriving at different times of automatic continuous time  $T_i = t_i + \Delta_i$  [3].

MFIS are an open structure, since they require generation of stored  $e(\Delta)$  input signals for their operation.

The number of memorized states in the  $i$ th group is determined depending on the number of  $R_p$ , the used OR-NOT (AND-NOT) elements in the  $i$ -th group. Elements in the  $i$ -th group are connected in the sense of reliability in parallel. The number of  $i$ -groups in the MFIS ranges from 2 to  $m$  ( $2 \leq i \leq m$ ). If we take the minimum number of groups, as shown in Fig. 4.5, then their interaction with each other forms a consistent connection in the sense of reliability.

Failures of the elements of the  $i$ -th group do not affect the functioning of the remaining elements of this  $i$ -th group. However, if the failing element at the output node has the value of an output signal that uniquely establishes inverse values on the

outputs of the elements of other groups, then such a failure is catastrophic for the functioning of the entire MFIS . In the future, we will consider non-catastrophic failures of elements whose output signals of elements do not affect the functioning of elements of other groups.

The minimum number of elements necessary for the functioning of the memory scheme, with one AND-NOT (OR-NOT) element in the group is equal to the number of  $m$  groups. This, so-called, multi-stable memory circuits [4]. The MFIS structure can be considered as a memory scheme, which is reserved in each  $i$ -th group  $(R_i - 1)$  element. In this case, the MFIS is considered as a scheme consisting of  $m$  workers and  $m \cdot (R_i - 1)$  reserve elements. All  $N = m \cdot R_i$  elements can fail.

The number  $M$  of memorized states of the MFIS can fluctuate within

$$m \leq M \leq \sum_{i=1}^m (2^{R_i} - 1), \quad (4.21)$$

If a  $j$ -th failure occurred in the  $i$ -th group of the MFIS by the time  $t$ , then the number  $K_i$  of the remembered states of the  $i$ -th group will change and form the memorized states. In other words, if all  $R_i$  elements fail, the  $i$ -th group falls into the failure state and no changes occur in this  $i$ -th group. To evaluate the performance of the MFIS, which in case of failures  $(R_i - 1)$  of the elements in each group is converted to a multistable memory circuit, which stores all states with one inactive save input signal [4], it must be tested. It is also convenient to use to evaluate the performance of the MFIS with the number of  $r_e$  blocks of the  $\pi_i$  states, which are stored with the corresponding failures of the elements in the  $i$ -th groups. The number  $r_e$  of stored state blocks  $\pi_i$  is determined by the formula (4.14) and can be in the range

$$1 \leq r_e \leq \prod_{i=1}^m (2^{R_i} - 1). \quad (4.22)$$

If a  $j$ -th failure occurs in the  $i$ -th groups by the time  $t$ , then the MFIS stores the number of state blocks  $\pi_j$ ,

$$r_e = \prod_{i=1}^m (2^{R_i - j} - 1). \quad (4.23)$$

In terms of reliability, the MFIS is a parallel-sequential scheme in which the elements of each  $i$ -th group represent a parallel scheme of elements with the same parameters, and the groups of elements are connected to each other in sequence.

Each  $j$ -th element in the general case is characterized by the failure rate  $\lambda_j(t)$  and the probability of failure-free operation

$$P_j(t) = \exp \left[ - \int_0^t \lambda_j(t) dt \right].$$

The probability of failure-free operation of the  $i$ -th group of the MFIS as a whole is determined by the formula:

$$P_{\varphi}^i(t) = 1 - (1 - P_j(t))^{R_i}. \quad (4.24)$$

The probability of failure-free operation of the MFIS on the interval  $[0, 1]$  with different number of elements in the group can be determined by the formula:

$$P(t) = \prod_{i=1}^m [1 - (1 - P_j(t))^{R_i}]. \quad (4.25)$$

With the same number of elements in the group, the probability of failure-free operation of the MFIS is determined by the formula:

$$P(t) = [1 - (1 - P_j(t))^{R_i}]^m, \text{ at } R_i = \text{const}. \quad (4.26)$$

If the fail-safe operation time of an element is subject to an exponential law with a parameter (failure rate), then it is determined by  $\lambda_j$ , the formula:

$$P_j(t) = e^{-\lambda_j t}. \quad (4.27)$$

In this case, for successive connection, the probability of failure-free operation can be expressed in terms of the failure rate as follows:

$$P(t) = \prod_{i=1}^m P_i(t) = P_i^m(t), \text{ at } R_i = 1; \quad (4.28)$$

$$P(t) = e^{-\sum_{i=1}^m \lambda_j t} = e^{-m \lambda_j t}. \quad (4.29)$$

The mean time between failures of the MFIS by the known  $P(t)$  is determined by the formula:

$$T_{cp} = \int_0^{\infty} P(t) dt. \quad (4.30)$$

The mean time to failure of the MFIS at  $R_i = 1$  ( $R_i = const$ ) is determined by the formula:

$$T_{cp} = \frac{1}{m \lambda_j}. \quad (4.31)$$

For the failure rate of the elements  $\lambda_j = 1 \cdot 10^{-7} / \text{ч}$ , the mean time to failure of the MFIS for  $R_i = 1$  in all  $i$ -th groups ( $m = 2$ ) is equal to

$$T_{cp} = \frac{1}{2\lambda_j} = 0,5 \cdot 10^7 \mu.$$

When  $R_i = 2$  and  $m = 2$ , the probability of failure-free operation of the MFIS is determined by the formula:

$$P(t) = [1 - (1 - e^{-\lambda_j t})^2]^2 = 4e^{-2\lambda_j t} - 4e^{-3\lambda_j t} + e^{-4\lambda_j t}.$$

$$\text{Then } T_{cp} = \int_0^{\infty} (4e^{-2\lambda_j t} - 4e^{-3\lambda_j t} + e^{-4\lambda_j t}) dt = \frac{4}{2\lambda_j} - \frac{4}{3\lambda_j} + \frac{1}{4\lambda_j} = \frac{11}{12\lambda_j}.$$

For  $R_i = 3$  and  $m = 2$ , the probability of failure-free operation of the MFIS is determined by the formula:

$$P(t) = [1 - (1 - e^{-\lambda_j t})^3]^2 = e^{-4\lambda_j t} (e^{-8\lambda_j t} - 12e^{-7\lambda_j t} + 66e^{-6\lambda_j t} - 114e^{-5\lambda_j t} + 387e^{-4\lambda_j t} - 468e^{-3\lambda_j t} + 414e^{-2\lambda_j t} - 216e^{-\lambda_j t} + 81).$$

In this case,  $T_{cp}$ , determined by the formula (4.31), is equal to  $20,5 \cdot \frac{1}{\lambda_j}$ .

Thus, with an increase in the number of elements in the groups, the value of the mean time between failures increases, indicating an increase in the reliability of the MFIS as a memory scheme in comparison with single-phase multistable memory circuits.

It should be noted that as the number of groups with the same number of  $R_i$  elements increases, the value of the mean time between failures decreases, indicating that the most preferable in terms of improving reliability are MFIS with two groups with  $R_i > 1$  elements in each of them.

### 4.7.2. Issues of increasing the survivability of memory circuits.

At present, super-large integrated circuits (VLSIs) are built with the expectation of 100% suitability of all components of the circuit. Increasing the number of components and the very area of the VLSI crystal, increasing the length of the tires and reducing the width of their widths, naturally, increase the probability of failure of components and the appearance of breaks in their connections. This leads to a significant VLSI marriage and to a catastrophic failure of them during operation.

In order to improve the reliability of the operation of systems from unreliable elements, multiple backups, distributed network systems, etc., in which the outputs of a whole block or device are faulty, is determined by diagnostic programs and are not reflected catastrophically on the operation of the entire system as a whole. In addition, the unreliability of the basic elementary binary memory scheme, which is used in almost all digital VLSIs [10-13], is noted.

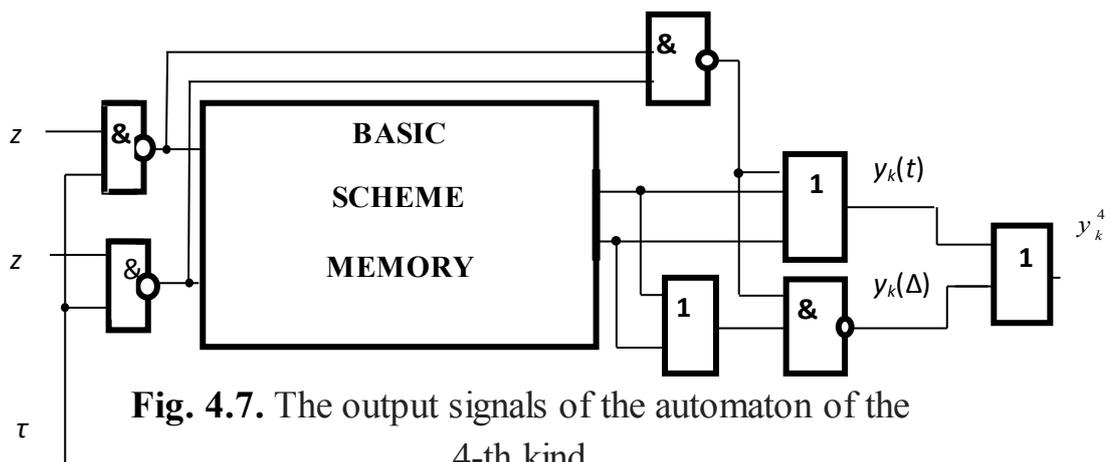
In MFIS, elements are used, the individual output signals of which are active signals for other groups. Suppose that the main malfunction of the elements is the appearance at the output nodes of a constant value equal to logical zero. This assumption is quite plausible, if we take into account that when the input node breaks in the logic elements of integrated circuits, the value of the input signal is perceived to be equal to the logical one. Consequently, the output signal of an element having a broken input, acquires a constant value equal to logical zero. In this case, such an element will simply not participate in the memorization of the states of this group, and the characteristic function of the  $i$ -th group ( $K_i = 2^{R_i} - 1$ ) will decrease its value by one, i.e. Becomes equal  $K_i = 2^{R_i-1} - 1$ , where  $R_i$  is the number of elements in the  $i$ -th group.

In case of malfunctions in the whole group of basic elements, the memory circuit will work as a storage device if the number of operable remaining groups is at least 2 (ie  $m \geq 2$ ) and in each group remains operational, at least for one base element.

### 4.7.3. Issues of monitoring the health of memory circuits

As we saw in the analysis of the basic memory circuits (RS-trigger, SME and MFIS) under the influence of the set of the  $x_p(t)$  input signal, when all input nodes of the logical elements have the same active value equal to 1 (0), then on the output nodes of these Circuits, a passive set of identical inverse values 0 (1) is established with the corresponding logical elements OR-NOT (NAND) in the memory scheme.

Based on the theory of automata of the fourth kind, it is possible to design a circuit that monitors catastrophic failure of the first type in any basic memory scheme. The organization of such a scheme, which controls the catastrophic failure in the basic memory scheme, is shown in Fig. 4.7. In Fig. 4.7 shows the basic memory scheme, which can be either an RS flip-flop, or an SMP, or an MFP. It can use one input signal that, together with the synchronous pulse  $\tau$ , forms a set of the input  $x_p(t)$  input signal at the nodes of the NOR gate (NAND) of the base memory circuit, where each signal from this set is equal to logical 1 (0).



**Fig. 4.7.** The output signals of the automaton of the 4-th kind

This set of setting  $x_p(t)$  input signal in parallel through the AND-NOT (OR-NOT) together with the output signals of the base memory circuit is fed to the OR (AND), circuit (I) whose output generates the signal of the 4th kind automaton  $y^4(t)$ .

The signal of the automaton of the 4th kind  $y^4(t)$ , when the logical node 0 appears on the output node, indicates that the basic memory scheme is working, and when - 1, this indicates that the basic memory scheme has catastrophically failed and is inoperable.

However, after the input signal  $x_p(t)$  ends at the input nodes of the memory circuit, an input signal that stores the  $e(\Delta)$  appears, at which the input nodes of the memory circuit receive passive input signals equal to the logical 0. With such a preserving  $e(\Delta)$  input signal output nodes of at least one logical element of the memory circuit, an active output signal equal to logical 1 should appear, if the circuit works correctly. If in the scheme there is a catastrophic failure of the second type, in which all output signals do not have a value equal to logical 1, then this memory scheme is inefficient, as in the case of catastrophic failure of the first type. The signal  $y^{B3}(\Delta)$  formed in the circuit of Fig. 5.7, at a logic value of 0 indicates the memory circuit is working, and at a value of 1 indicates a memory scheme is inoperative. Thus, the output signal  $y_k^4$  of the memory scheme monitoring circuit and indicates the operability of the basic memory scheme (Fig. 4.7). Thus, it can be stated that for any failures of a single logic element in the basic *RS*-flip-flop circuit, the basic scheme catastrophically leaves the field of operability. In basic memory schemes, such as: SMEs and MFIS, the basic memory scheme, as we see, has two types of failures: catastrophic first and second types and not catastrophic. With non-catastrophic failures, the SME and the MFIS have the ability to narrow the scope of their reliable functioning and exist as a memory scheme. Identification of the operational area of SMEs and MFIS in non-catastrophic failures in memory circuits can be carried out when testing them. Catastrophic failures in the memory circuit, which lead completely to the inoperability of the basic memory scheme, can be controlled, which is very important in responsible control systems used for nuclear power stations, missile technology, aircraft equipment, railway transport, etc.

## 4.8. Characteristics of multifunction circuit parameters memory

Taking into account the specifics of the construction of an MFIS class  $L$  and an  $L^M$  class, which combines the AND-NOT (OR-NOT, etc.), the memory circuits are characterized by such parameters [4]:

- the maximum number of  $M$  memory states when constraining the parameters of the logical elements from which the memory scheme is built;
- limiting working frequency of switching  $F_p$ ;
- Load capacity at outputs  $P_\varrho$ ;
- the number  $S_{\text{внут.с.}}$  of inches. Internal relations;
- the number  $S_{\text{внеш.с.}}$  of ext. External relations;
- the number of  $L$  elements per state;
- the maximum number of alternative mappings  $r_e$ .

**The maximum number of storage states.** For MFIS class  $L$  on the basis of  $K$ -input AND-NOT (OR-NOT) with load capacity  $P_1$ , the number  $R_i \left( R_i = 2, 3, \dots, \frac{K}{2} \right)$  of which is the same in all  $m$  groups, the maximum possible number of  $M_{\max}$  memory states for  $n = K$ ;  $m = 2$ ;  $R_i = \frac{K}{2}$  ( $K \leq P_1$ ) is calculated by the formula:

$$M_{\max} = \sum_{i=1}^m (2^{R_i} - 1) = \sum_{i=1}^2 \left( 2^{\frac{K}{2}} - 1 \right) = 2 \cdot \left( 2^{\frac{K}{2}} - 1 \right). \quad (4.32)$$

For MFIS of the  $L^M$  class on the basis of  $K$ -input elements NAND (NOR) with load capacity  $P_1$ , the number of  $R_i$  ( $R_i = 2, 3, \dots, R$ ) of which is the same in all  $m$  groups, and  $R$  are input elements of AND (OR) with the load capacity  $P_2$ , the maximum possible number of  $M_{\max}$  memory states for  $n < P_2$ ;  $M = K-1$ ; ( $K \leq P_1$ );  $R_i = R$ ;  $n = m R$  is calculated by the formula:

$$M_{\max} = \sum_{i=1}^m (2^{R_i} - 1) = \sum_{i=1}^{K-1} (2^{R_i} - 1). \quad (4.33)$$

For example, let's calculate the  $M_{\max}$  for MFIS of different types based on four-input elements NAND (NOR) with load capacity  $P_1 = 10$  and triad elements AND with load capacity  $P_2 = 10$ .

For an MFIS class  $L$ , according to (4.32), we have:

$$M_{\max} = \sum_{i=1}^m (2^{R_i} - 1) = 2(2^2 - 1) = 2 \cdot 3 = 6.$$

For an MFIS of the class  $L^M$ , according to (4.33):

$$M_{\max} = \sum_{i=1}^m (2^{R_i} - 1) = \sum_{i=1}^3 (2^{R_i} - 1) = 21.$$

**Limiting operating switching frequency ( $F_p$ ).** The maximum switching frequency  $F_{\max}$  for an MFP of Class  $L$  is determined by the minimum allowable time interval between two successive signals of the minimum duration. The maximum switching frequency  $F_{\max}$ , as for triggers, is calculated by the formula:

$$F_{\max} = 1/2\tau_{cp},$$

where  $\tau_{cp}$  - the average delay of one logical element.

But for the MFIS class  $L^M$ , the maximum switching frequency is calculated by the formula:

$$F_{\max} = 1/3\tau_{cp}. \quad (4.34)$$

Thus, the appearance of a new information signal is permissible only after the transient process is over in the MFIS and switched to another stable state.

For reliable transmission of information, the initial information (active) memory circuit signal can be used only after the transient process ends, the duration of which is  $2\tau_s$  for the MFIS class  $L$  and  $3\tau_s$  for the MFIS class  $L^M$ .

Consequently, the limiting operating frequency  $F_p$  of the switching of an asynchronous MFIS class  $L$  with the duration of the incoming information (active) signal equal to  $2\tau_s$  is determined by the formula:

$$F_p \leq 1 / 4\tau_s,$$

whereas for the MFIS of the  $L^M$  class by the formula:

$$F_p \leq 1 / 6\tau_s, \quad (4.35)$$

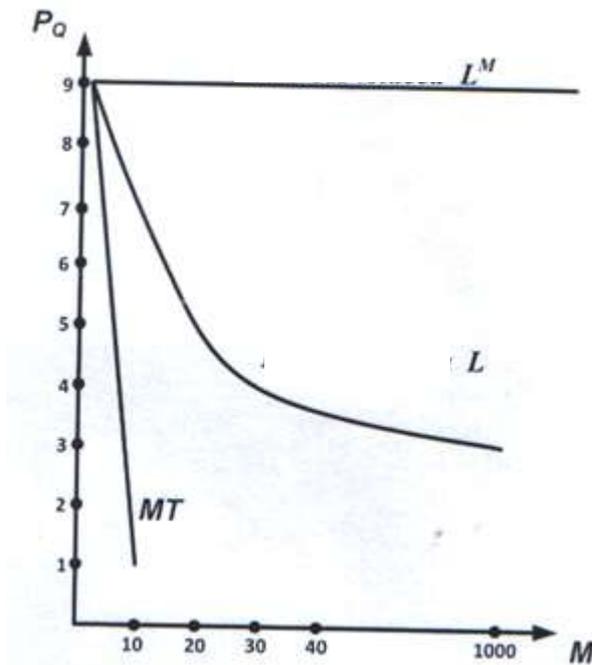
**Load capacity on outputs ( $P_Q$ ).** For MFIS class  $L$ , the load capacity at outputs  $P_Q$  is determined by the formula:

$$P_Q = P_e - R_i = P_e - \frac{K}{2} \left( P_Q \geq 1; P_1 \geq \frac{K}{2}; R_i = \frac{K}{2} \right), \quad (4.36)$$

where  $R_i$  – is the number of elements in the  $i$ -th ( $i = 1, 2$ ) MFIS group.

For MFIS class  $L^M$ , the load capacity for the outputs of  $P_Q$  is determined by the formula and does not depend on the parameters  $P_Q$ , the elements used.

$$P_Q = n_s - 1 \quad (4.37)$$



**Fig. 4.8.** Changes in the load capacity of the  $P_Q$  of multistable triggers (MT) and MFIS with increasing stored states

For example, for MFIS class  $L$ , we have:

$$P_Q = P_e - \frac{K}{2} = 10 - 2 = 8.$$

For MFIS of class  $L^M$

$$P_Q = P_e - 1 = 10 - 1 = 9.$$

Consequently, in the case of a significant (from 4 to 21) increase in the number  $M_{max}$  of storage states, the load capability of the  $P_Q$  MFIS, in contrast to the multistable triggers, does not decrease, but increases, indicating a reduction in the number of internal links between the logical elements in the memory scheme. The graphical de-

pendence of the load capacity  $P_Q$  on the number  $M$  of memory states is shown in Fig. 4.8.

The graphs shown in Fig. 4.8, clearly illustrate the changes in the load capacity of the  $P_Q$  of multistable triggers (MTs) and MFIS of different classes with an increase in the number  $M$  of their storage states.

We come to the conclusion that in terms of such a parameter as the load capacity, the MFIS of the  $L^M$  class are characterized by the constant (maximum) load capacity of  $P_Q$ .

**The number of internal connections ( $S_{\text{внут.с}}$ ).** The parameter  $S_{\text{внут.с}}$  for an MFIS of class  $L$  is calculated by the formula:

$$S_{\text{внут.с}} = mR_i(n - R_i), \quad (4.38)$$

where  $n$  – is the number of NAND (NOR) elements used in the MFIS;

$M$  ( $m < n$ ) – is the number of groups of such elements in the MFIS;

$R_i$  – is the number of elements NAND (NOR) in the  $i$ -th group of the MFIS.

For MFIS class  $L^M$  parameter  $S_{\text{внут.с}}$  is calculated by the formula:

$$S_{\text{внут.с}} = n + mR_i(m - 1), \quad (4.39)$$

where the notation is the same as in (5.38).

For comparison, we calculate the parameters  $S_{\text{внут.с}}$  and  $M_{\text{max}}$  for different types of memory circuits with the same number of  $n$  logical elements.

Consider, for example, schemes that use 10 ( $n = 10$ ) logical elements.

For single-phase multistable triggers (MT), according to the formula, we have:

$$S_{\text{внут.с}} = n(n - 1) = 10 \times 9 = 90, \text{ a } M = n = 10.$$

For MFIS class  $L$  with  $m = 2$ ,  $R_i = 5$ :

$$S_{\text{гнум.с}} = mR_i(n - R_i) = 2 \times 5 \times 5 = 50,$$

$$M_{\text{max}} = \sum_{i=1}^m \left( 2^{R_i} - 1 \right) = 2 \left( 2^5 - 1 \right) = 2 \cdot 31 = 62 .$$

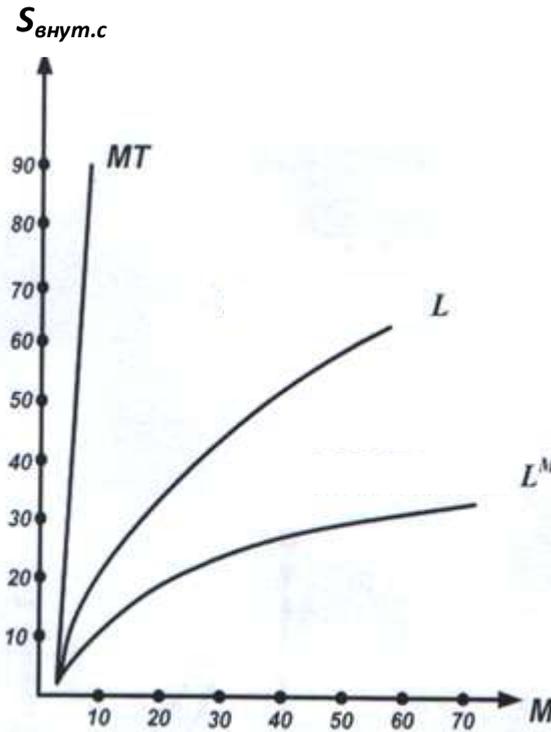
For MFIS of the class  $L^M$  for  $m = 2$ ,  $R_i = 4$ ,  $n = 8$  (elements of NAND) and two elements of AND we have:

$$S_{\text{гнум.с}} = n + m R_i (m - 1) = 8 + 2 \cdot 4 \cdot 1 = 16,$$

$$M_{\text{max}} = \sum_{i=1}^m \left( 2^{R_i} - 1 \right) = \sum_{i=1}^2 \left( 2^4 - 1 \right) = 2 \cdot 15 = 30 .$$

The graphical dependence of the number of internal bonds  $S_{\text{гнум.с}}$  on the number of  $M$  memory states in various memory circuits is shown in Fig. 4.9.

Considering the graphs shown in Fig. 5.9, we see how the values of  $S_{\text{гнум.с}}$  for the multistable triggers (MSPs) and the MFIS of different classes increase with increasing  $M$ . Consequently, by the number of internal connections,  $S_{\text{гнум.с}}$  has the advantage of MFIS of the class  $L^M$ , in which this number is the smallest.



**Fig. 4.9.** Dependence of the number of internal links  $S_{\text{внут.с}}$  on the number of  $M$  memorized states in different memory circuits

**The number of external links ( $S_{\text{внеш.с}}$ ).** For a single-phase, multistable memory scheme,  $S_{\text{внеш.с}} = 2n$ , but for MFIS of different classes for the same value of  $M$  (the number of memory states)  $S_{\text{внеш.с}} < 2n$ .

So, for a single-phase, multistable memory scheme, storing 6 states and built on 6 logical elements, the value of  $S_{\text{внеш.с}}$  is 12, and for MFIS - 8.

**Number of elements per state ( $L$ ).** For an asynchronous single-level  $RS$  trigger and all single-phase multistable triggers, the value of the parameter  $L$  is one, since  $M = n$  [17; 125].

For MFIS of class  $L$ , we have:

$$L = \frac{\sum_{i=1}^m R_i}{\sum_{i=1}^m \left( 2^{R_i} - 1 \right)}, \quad (4.40)$$

and for the MFIS class  $L^M$ :

$$L = \frac{m + \sum_{i=1}^m R_i}{\sum_{i=1}^m (2^{R_i} - 1)}. \quad (4.41)$$

For example, for MFIS of class  $L$ , and for MFIS of class  $L^M$

at  $n = 4; m = 2; R_i = 2$

$n = 4 + 2; m = 2; R_i = 2$

$n = 6; m = 2; R_i = 3$

$n = 6 + 2; m = 2; R_i = 3$

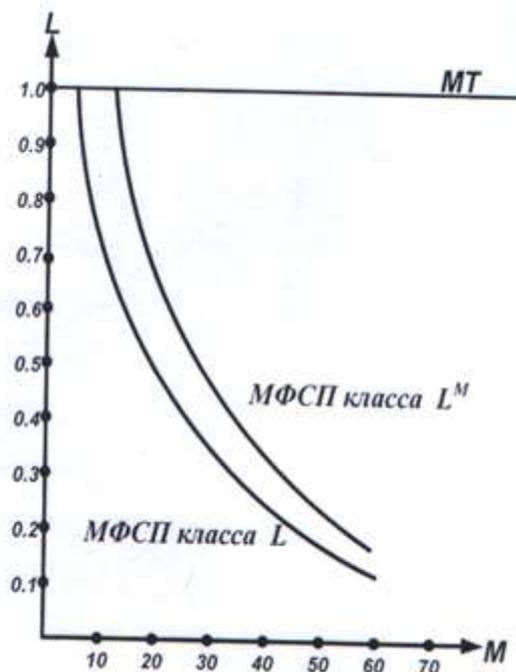
$n = 8; m = 2; R_i = 4$

$n = 8 + 2; m = 2; R_i = 4$

$n = 10; m = 2; R_i = 5$

$n = 10 + 2; m = 2; R_i = 5$

The graphical dependence of the number of  $L$  elements on one state on the number  $M$  of memory states in various memory circuits is illustrated in Fig.



**Fig. 4.10.** Dependence of the number of  $L$  elements on one state on the number  $M$  of memory states

4.10.

**The maximum number of alternative mappings ( $r_\rho$ ).** In the MFIS, one input of each AND (NOR) element of each  $i$ -th group can be connected according to one node  $z_i$  of the setting bus Bx.IIIZ of the whole memory circuit. In this case, you can specify in the table form the MFIS state matrix, as discussed in Table. 4.6.

In contrast to the known binary memory circuits [17; 125] after the  $x_i(t)$  is set by the input signal to the state  $A_k(t)$  of the memory circuit, they can change the states for different sets of  $e_j(\Delta)$  -containing input signals, i.e.  $\{X\} \xrightarrow{e_j} \{A\}$ , where  $e_j$  is the functionality that determines the function  $\delta_e^{(j)}$  of the saving states of the block  $\pi_j$  [7].

The number of memorized states  $M$  can be considered as entropy, which serves as a measure of the freedom of the system: the more entropy, the more states are available to the system, the more degrees of freedom it has [14].

Multistable memory circuits [4] function only in one block of their states, which are remembered under the influence of only one set of  $e_0(\Delta)$  that retains the input signal, which determines the zero degree of freedom.

MFIS function in different blocks  $\pi_j$  of their states, which are stored under the influence of the corresponding sets of  $e_j(\Delta)$  input signals that preserve it. Due to this, the possibilities of monofunctional memory circuits are expanded, which determines the re degrees of freedom. In this case, it is possible to realize alternative maps  $\{X\} \xrightarrow{e_j} \{a\}$  for various conserved input signals  $e_j(\Delta)$  in the memory circuit itself, without corresponding switching of the input and output signals.

Known multifunctional memory circuits achieve multifunctionality due to commutation of input and output signals with corresponding costs of additional logic elements and time for signal passing through switching circuits [7; 15-16]. In contrast to them, MFIS are able, under the influence of only sets of  $e_j(\Delta)$  input signals that remain in the memory circuit itself, to perform enlarged transitions in the internal clock cycle  $\Delta$  of one machine clock  $T$ .

So, we consider the MFIS of different classes with respect to the basic parameters, which have advantages over the class of multistable memory circuits, a special case of which is a single-stage asynchronous *RS*-flip-flop.

These MFIS also have qualitatively new functional properties. On the use of alternative blocks of  $\pi_j$  states at an arbitrary moment in time, which do not have binary triggers [4-5], multistable triggers [4], as well as built multifunctional circuits with memory based on triggers with commutation of their input and output nodes due to additional logical elements [15-16]. A comparative characteristic of the parameters of the basic memory circuits is given in Table 4.9.

Thus, from Table. 4.9 visually see the advantages of MFIS the characteristics of memory circuits.

An increase in the maximum number of memory states in the MFIS is possible up to 30-90, instead of 8 possible in the SME. By decreasing the number of internal connections in the circuit with the state  $M = 18$  from 756 (in MSPs) to 12-18 (in MFIS), which is very important in the construction of integrated memory circuits. By reducing the number of hardware costs of logical elements per state from 1 to SMEs and to 0.2 to 0.3 in the MFIS.

Table 4.9

Comparative parameters of basic memory circuits

Parameter	A multistable memory circuit (SMEs)	MFIS class $L$	MFIS class $L^M$	Class advantage MFIS
$M_{max}$	8	30	90	$L^M$
$F_p$	125 МГЦ	125 МГЦ	100 МГЦ	$L$
$P_Q$	3	6	9	$L^M$
$S_{\text{sym.c.}}(M = 28)$	756	18	12	$L^M$
$L (M = 28)$	1	0,2	0,3	$L$
$r_e$	1	>3	>3	$L$ $L^M$

An important functional advantage of MFIS over SMEs is the possibility of reconstructing the structure of storage states in the process of operation for one machine clock, i.e. Increase in the degree of freedom from 1 to  $r_e$

Multifunctional memory circuits, as was shown earlier, have an advantage over the basic binary memory scheme of the *RS*-flip-flop. MFIS reduce the hardware costs for a single memorized state; Increase the functionality by realigning the structure of state storage and simultaneously processing the levels of hierarchical information represented as sets of input-preserving signals  $e(\Delta)$  as general information and private information represented as sets of input signals  $x(t)$  in one machine clock cycle  $T$ .

In addition, MFIS have increased reliability and survivability, which is very important for their use in building computer systems for such important facilities as nuclear power plants, air and rail transport, space exploration, which is important for the security of any country.

#### **4.9. Conclusion for Chapter 4**

Multifunctional memory circuits, as shown earlier, have an advantage over the base binary memory circuitry of the *RS*-trigger. MFISs reduce hardware costs for one stored state; enhance the functionality by reorganizing the state of the storage structure and simultaneously processing the levels of the hierarchical information presented as sets of preserving inputs  $e(\Delta)$  as general information and the private information represented as sets of input signals  $x(t)$  for one machine cycle  $T$ . In addition, MFISs have increased reliability and vitality, which is very important for their use in the construction of computer systems for such important objects as: nuclear power plants, air and rail transport, astronautics, which is important for the safety of any country.

## Chapter 5. Multilevel memory schemes

### 5.1. Basic concepts

MFIS are an open structure that has the ability to re-structure the state storage structure, and, consequently, to change the direction of the active output information for certain output nodes. In addition, the MFIS has two sets of input signals: setting and storing. These two sets of input signals do not intersect in time. To use the stored input signals, it is necessary to generate them from an additional source. Such an additional source of input signals may be the outputs of another memory circuit. In this case, such a combined memory scheme becomes hierarchical. The entropy of such a hierarchical scheme of memory naturally increases.

In the hierarchical scheme of memory, naturally, there are levels. The maximum number of levels when creating a semi-closed memory circuit is limited to the last (lower) level, which preserves its states only for one storing the input signal  $e(\Delta)$ , as a trigger or a multistable memory scheme. If the lower level of the hierarchical memory scheme has the ability to use several sets of input-preserving input signals ( $r_e > 1$ ), then this memory structure is open.

In this chapter, we will look at the methods for the structural organization of multi-level memory schemes. At the same time, we will pay attention mainly to semi-closed hierarchical memory circuits, which are more familiar to developers (by analogy with the trigger) to use in the design of computer systems. It is necessary to note the analogy of the MFIS with a living cell (neuron), which also has two sets of input signals: excitatory and inhibitory, as well as many output signals that are directed to certain neurons in certain directions.

### 5.2. Symbolic language for describing multi-level memory schemes

A promising and actual trend in the development of memory circuits is the creation on the basis of the MFIS of multilevel memory schemes (MFIS), which are the

basis for solving the actual problem: to store general and particular (hierarchical) information for one machine clock cycle, which cannot be performed on computer devices with memory on triggers [4].

The construction of monofunctional memory circuits (triggers) and methods of constructing devices based on them, whose operation is considered in automatic discrete time, is considered an almost completed topic [4-6]. The situation is different with automatic memory elements, such as non-functional and multilevel memory schemes, and methods for constructing devices based on them, whose operation is considered in automatic continuous time [1-3].

This is explained by the fact that in MFIS, in which  $e_j(\Delta)$ , the input signal preserves certain states of the subset (of the  $\pi_j$  block), new transition functions appear, which expands the functionality of the MUSP using the MFIS in their structure.

Synchronous discrete devices use different types of sync pulses. The most common case is when two series of sync pulses  $\tau_1$  and  $\tau_2$  are used [5; 7]. When sync pulses are applied to memory devices, the memory is called synchronized and is determined by the symbols  $\tau_1$  or  $\tau_2$ . A memory that does not use sync pulses is called asynchronous and is displayed by the  $\emptyset$  symbol.

Single-stage MUSP use several vertically connected MFIS [3], which can be synchronized with a single signal. The number of  $K$  levels of MUSP in its symbolic description is one less than one with the symbol  $y$  for the class  $L_N$  and with the symbol  $b$  for the class  $L_N^B$  [3].

The MUSP can be one- and two-stage. The other stage works in the mode of overwriting the code from the first degree to the other with the synchronous pulse  $\tau_2$ . In the symbolic description, one- and two-stage MUSP are displayed respectively  $1c$  and  $2c$ .

In the symbolic description, the structure of the MUSP is represented in the following order:

- 1)  $(K-1)y$  or  $(K-1)b$  - displays the  $K$ -level structures of the MUSP;

2)  $A_{k-1}, A_{k-2}, \dots, A_0$  - reflects the symbolic description of each MFIS structure, starting from the upper level of the MUSP;

3)  $\tau_i$  ( $i = 0, 1, 2$ ) - reflects the first (or second) stage of the synchronizing MUSP with the signal  $\tau_i$  (or  $\tau_j$ ) or the asynchronous MUSP -symbol  $\emptyset$ ;

4)  $Rc$  ( $R = 1, 2$ ) - reflects one- or two-stage structures of the MUSP.

The symbolic descriptions of the MFIS and MUSP ( $n > 2$ ) are key when describing any class of MUSP. For an MUSP of a class  $L_N$ , each symbolic number  $A_i$  of the MFIS reflects the relationship of the output signals to the MFIS of all lower levels that generate sets of  $e_j(\Delta)$  -containing input signals, with the incoming nodes of the bus-saving upper MFIS. For the MUSP class  $L_N^B$ , the symbolic number  $A_i$  of the MFIS reflects the relationship of the output signals to the MFIS of all lower layers. In this case, sets of  $e_j(\Delta)$  input signals with input nodes for one group of upper MFIS logical elements are generated, in which the number of logical elements has a value greater than one ( $q > 1$ ).

We give examples of the symbolic description of MUSP. Consider a symbolic description of an asynchronous single-stage memory scheme that has an MFIS on three levels and is described by such a record:

$$2y, 22, 112, 111, \emptyset, 1c, \quad (5.1)$$

where  $2y$  - reflects three levels of the MUSP class;

22 - Displays the structure of the MFIS of the third level, which consists of four logical elements, divided into two groups, two elements each;

112 - Displays the structure of the second-level MFIS consisting of four logical elements, divided into three groups, two of which have one element each, and one has two elements;

111 - displays the structure of the first-level MFIS, which consists of three logical elements, divided into three groups, one element each;

$\emptyset$  - indicates that the MUSF is asynchronous;

1c - indicates that the MUSF is one-step.

The symbolic description of an MUSF class  $L_N^B$ , which has the same functionality as the considered class  $L_N$ , MUSF, is represented in this form:

$$1b, 22, 111, 111, \emptyset, 1c, \quad (5.2)$$

where  $1b$  - reflects the two levels of the MUSP class;

22 - displays the structure of the second-level MFIS consisting of four logical elements, divided into two groups, two elements each;

111 - display the structures of the first-level MFIS, which consist of three logical elements, divided into three groups, one element in each and associated with each of the second-level MFIS;

$\emptyset$  - indicates that the MUSP is asynchronous;

1c - indicates that the MUSP is one-step.

### 5.3. Determining the parameters of multi-level memory schemes by symbolic description

From the symbolic description of the upper MFIS, it is possible to determine the number of states of the entire MUSP, which are stored using the following formula:

$$M = m_k \cdot r_e, \quad (5.3)$$

where  $m_k$  – is the number of bits in the symbolic description of the upper MFIS;

$r_e$  – is the number of sets of  $e_j(\Delta)$  input signals retained for the upper MFIS is determined by the formula (4.14).

The number of MUSP states that are stored can also be determined from the number of  $m_k$  bits in the symbolic description of each MFIS using the following formula:

$$M = \prod_{i=1}^K m_i, \quad (5.4)$$

where  $m_i$  ( $i = 1, 2, \dots, K$ ) – is the number of bits in the symbolic description of the  $i$ -th MFIS of the entire MUSP;

$M$  – is the number of MUSP states that are stored.

For example, from the symbolic description 2y, 22, 112, 111,  $\emptyset$ , 1c MUSP of the class's  $L_N$  it becomes clear that the upper MFIS has a numeric description 22, which makes it clear that  $m_3 = 2$ ,  $r_e = 9$ , and therefore the number of MUSP states will be By the formula (5.3) is equal to 18, that is,  $M = 2 \times 9 = 18$ .

Another way to find the number  $M$  of memorized states of an MUASP can be determined by formula (5.4), which consists of the fact that the number of bits in the numbers of the symbolic description of the MUSP is initially determined, and then we find their product. From the symbolic description of the MUSP of class  $L_N$  (5.1) and class  $L_N^B$  (5.2), using formula (5.4), we determine that the number of states of these MUSP that are remembered is the same and equal to 18 ( $M = 2 \times 3 \times 3 = 18$ ).

Due to the fact that the lower MFIS have only one logical element in each digit, these MUSP are semi-closed structures. The MUSP also has an internal multifunction connection, and stores all of its states with one set of  $e_i(\Delta)$  - supervising inputs, when at all input nodes the signal is equal to logical zero, that is, it is inactive.

The search for a symbolic description of MUSP is a creative process. There can be quite a lot of variants of such a description. However, in order to preserve the states of the managed MFIS when creating a symbolic description of the strategy automaton (lower MFIS), the following relation must be observed:

$$r_e \leq M, \quad (5.5)$$

where  $r_e$  – is the number of sets of  $e_j(\Delta)$  input signals of the high-level MFIS of the MUSP;

$M$  – is the number of MFIS states of the lower levels of the MUSP, which are remembered.

The process of searching for symbolic descriptions of the lower MFIS ends when the number of sets of  $e_j(\Delta)$  preserving input signals of the lowest MFIS is one. When a symbolic description of all MFIS in the MUSP is found, then by the formula (5.4) it is possible to determine the number of states of the received MUSP.

#### **5.4. The method of synthesis of multilevel memory circuits by symbolic description**

When synthesizing an MUSP based on  $K$ -input logic elements NAND or NOR with load capacity  $P_1$  and  $R$ -input logic elements AND or OR with load capacity  $P_2$ , it is necessary to take into account the limitations of the used logic elements.

To synthesize a symbolic description of an MUSP, it is first necessary to verify the feasibility of synthesizing an MFIS that is part of the MUSP, taking into account the constraints of the logical elements, and then designing the MFIS with the corresponding symbolic description.

Given the relationship between the levels of the MFIS in the MUSP, it is necessary to ensure that the load capacities of the  $P_1$  of the lower MFIS satisfy this relationship:

$$P_1 \geq \sum_{i=2}^m R_i + K, \quad (5.6)$$

where  $R_i$  – is the value of the digits (except for one minimum digit) in the symbolic description of the lower MFIS;

$K$  – is the number of MUSP levels.

$m$  is the number of bits of the symbol number of the upper MFIS.

The number of admissible input nodes of the logical elements of the upper IFRS  $K_b$  must satisfy the following relation:

$$K_b \geq \sum_{i=2}^m R_i + K, \quad (5.7)$$

where  $R_i$  is the value of the digits (except for one minimum digit) in the symbolic description of the upper MFIS;

$K$  is the number of levels of the MUSP;

$m$  is the number of bits of the symbol number of the upper MFIS.

For an MUSP class  $L_N^B$  with the use of an MFIS class  $L^M$ , the number of input nodes of the logical elements of the upper MFIS  $K_b$  must satisfy the following relationship:

$$K_b = m + K. \quad (5.8)$$

The essence of the synthesis of the MUSP on the MFIS consists in finding hierarchical relationships between the output nodes of the lower MFIS and the input nodes of the upper MFIS.

For an MFIS of the class of connection between the output nodes of the lower MPSEs with the upper ones, taking into account active sets of  $e_j(\Delta)$  preserving input signals. The output nodes of the lower MFIS are connected to the input nodes of the upper MFIS [1-2]. For an MUSP, the class  $L_N^B$  of connection between the output nodes of the lower MFIS with certain groups of upper MFIS is carried out according to the active sets of  $e_j(\Delta)$  preserving input signals. The output nodes of the lower MFIS are connected to the input nodes of separate groups of upper MFIS, in which the number of elements is greater than one ( $q > 1$ ) [3].

Performing connections between levels in the MUSP, we obtain an asynchronous single-stage MUSP, which can be made a synchronous and two-stage memory scheme, as it is performed in two-stage *RS* flip-flops [4-5; 7].

Thus, in the symbolic description of the MUSP, enough data is available to perform the synthesis of the MUSP, to determine the number of states that are remembered, to construct a functional scheme with direct connections, and then use mathematical modeling to determine the law of functioning of the synthesized MUSP.

When synthesizing an MUSP on an MFIS, either the number of  $M$  states that are stored, or the number of  $r_e$  active sets of  $e_j(\Delta)$  storing input signals, or any other parameters of the memory circuit that needs to be designed, are first set. By specifying the main parameter of the MUSP memory scheme that needs to be designed, we define the symbolic description of the future upper MFIS and, based on its description, we calculate the parameter criteria according to the conditions that define the parameters of the MFIS .

The symbolic description of the upper MFIS constructed in this way is considered as a working variant. There are several possible variants of symbolic description of different MFIS, the parameters of which correspond to the specified parameters of MUSP. The designer chooses the MFIS option, given his own experience. The number of  $r_e$  sets storing the  $e_j(\Delta)$  input signals of the upper MFIS is calculated according to the corresponding formulas [3]: determine the required number of MFIS states of the lower levels of the MUSP, symbolic description of which is constructed using the same algorithms as for the upper MFIS. Having determined the symbolic descriptions of the MFIS , which form a symbolic description of the MUSP, we verify their parameters according to equations (5.7) and (5.8). Having determined the symbolic description of the MUSP and all its MFIS, we perform the synthesis of the MFIS first, and then the connections between them in the appropriate order.

Thus, the synthesis of MUSP consists of the following steps:

1. Determine the symbolic description of the upper MFIS for a given number of states of the MUSP, which are stored in accordance with equation (5.4).

2. Taking into account the fulfillment of the relation (5.6), we design the symbolic description of the lower MFIS and on their basis we design the symbolic description of the MUSP.

3. By the symbolic description of the MFIS entering into the description of the MUSP, we design the MFIS on certain logical elements, while observing the limitations of the logical elements themselves in terms of the number of inputs and the load capacities according to relations (5.6) and (5.8).

4. Having determined the active signals of the retaining input signals of the MFIS, we make a connection between the output nodes of the lower MUSP and the sets of  $e_j(\Delta)$  input inputs of the logic elements of the upper MFIS in the asynchronous single-stage MUSP.

5. If, according to the symbolic description, the MUSP is synchronized, then the input nodes of the asynchronous MUSP are connected with the input nodes through the AND circuits that receive the additional clock.

6. If the symbolic description of the MUSP is two-stage, then we design the second stage of MUSP similarly to the first stage.

7. Output nodes of the first degree of MUSP are connected together with the synchronous pulse through the AND circuit with the input nodes of the second stage of the MUSP.

The methods of synthesizing MUSP by their symbolic description are considered, which makes it possible to formalize their design taking into account the limitations of logical elements in terms of the number of incoming nodes and the load capacity.

We would like to draw your attention to the fact that the hierarchy of program management is implemented in the MUSP, which allows processing private information in the upper MFIS simultaneously with general information in the lower MFIS, which in principle distinguishes them from triggers.

## 5.5. The principle of the structural organization of elementary multi-level memory schemes

The study of the operation of triggers, SMEs, MFIS and MUSP reduces to the consideration of methods for switching these memory circuits from one state to a new state.

The disadvantage of the multifunctional memory circuits of the MFIS [3] is the need to feed not only sets of  $x_i(t)$  input signals to the MFIS, but also the generation of sets of  $e_j(\Delta)$  preserving input signals that enable the MFIS to work in new subsets of their states.

The principle of the structural organization of elementary multi-level memory circuits consists in their partitioning into control and controllable multifunctional memory circuits (MFIS) [14], interconnected in this way:

BA of the  $i$ -th group of the controlled MFIS  $A_i$ , the number  $q_i$  of elements of which is greater than one ( $q_i > 1$ ), through the conserving input bus, is connected to the output buses of one or more MFIS  $A_k$  ( $k = 1, 2, \dots, i-1$ ); which establish the input and output buses of the MFIS  $A_i$  ( $i = 1, 2, \dots, N$ ) are connected to the common input and output buses of the multi-level memory scheme.

The essence of the principle of storing states in a multilevel memory scheme with a multifunctional organization system is that the sets of  $x_i(t)$  input signals of the state of the controlled MFIS  $A_i$  are remembered only if they belong to the state blocks  $\pi_i$ , which are stored under the influence of the set preserving  $e_j(\Delta)$  Of the input signal generated by the control MFIS  $A_k$ .

The state  $a_i$  of the controlled MFIS, set under the influence of the input signal set  $x_i(t)$ , may not belong to the state unit  $\pi_i$ , in this case, the state  $a_i$  of the MFIS in the block  $\mu_i$  goes to the new state  $a_k$ , under the influence of the set of  $e_j(\Delta)$  storing the input signal And, accordingly, will be stored in the state block  $\pi_j$ . When memorizing combined states in multilevel memory schemes, a qualitatively new vertically hierarchical relationship arises that determines the state of the  $a_i$  managed MFIS, depending on the storage states of the  $a_k$  control MFIS.

A multi-level memory scheme with a multifunctional organization system defines a structure in which a multi-functional mode of operation of one device is determined by another device, the so-called  $A_M$  strategy automaton. The automatic strategy  $A_M$  in multi-level memory circuits can be a mono-or multi-level memory scheme. The structure of multi-level memory schemes with a multifunctional organization system can be defined as a scheme that preserves the overall (in the  $A_M$  strategy) and private (in the managed MFIS ) information at different levels.

## 5.6. The method of designing a general strategy automaton for the entire multi-level memory scheme

When implementing the  $A_M$  strategy automaton on the structures of multistable memory circuits (SMP ) [4] it is sufficient to know the required number of  $r_e$  sets of  $e_j(\Delta)$  -containing input signals for the controlled MFIS  $A_y$  [3] in order to apply  $r_e$  states to the SMP used as an automaton of the  $A_M$  strategy.

We consider SMP for 9 states [4], as an automaton of the  $A_M$  strategy, for the purpose of organizing and generating nine  $e_j(\Delta)$  input signals for a controlled MFIS  $A_y$ . The sets of the input signals of the nine-stable flip-flop are established in Table. 5.1.

The sets of setting input signals for the nine-stable trigger constructed on the OR-NON elements are characterized by the fact that each logical node  $z_i$  is supplied with a logical unit, except for one node  $z_k$ , which is fed with a logical zero.

In this case, the active logical units set the outputs of their elements to the values of the output signal equal to the logical zero, which along the feedback loop together with the input signal  $z_k$  ( $z_k = 0$ ) set the value of the logical unit at the output of this element.

The set of the  $e(\Delta)$  input signal preserving the SMP is zero at all input nodes  $z_j$  ( $z_1 = z_2 = z_3 = z_4 = z_5 = z_6 = z_7 = z_8 = z_9 = 0$ ), at which all nine states of the strategy automaton are stored.

The sets of setting  $x_i(t)$  input signals unambiguously establish the state of the SMP, nine of which are stored with one input signal  $e(\Delta)$ . Let us consider single-valued memory states (Fig. 5.1), established by  $x_i(t)$  by input signals, which are presented in Table. 5.1.

Table 5.1

Sets of setting input signals ( $x_i(t)$ )

Input signal $z_j$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_0$
$z_1$	0	1	1	1	1	1	1	1	1	1
$z_2$	1	0	1	1	1	1	1	1	1	1
$z_3$	1	1	0	1	1	1	1	1	1	1
$z_4$	1	1	1	0	1	1	1	1	1	1
$z_5$	1	1	1	1	0	1	1	1	1	1
$z_6$	1	1	1	1	1	0	1	1	1	1
$z_7$	1	1	1	1	1	1	0	1	1	1
$z_8$	1	1	1	1	1	1	1	0	1	1
$z_9$	1	1	1	1	1	1	1	1	0	1

The state  $A_0$  (Table 5.2) is not conserved for any  $e(\Delta)$  of the input signal, since in none of the groups the output signal  $b(T)$  of the  $A_M$  strategy automaton under the state  $A_0$  has an active output signal equal to the logical one, which is necessary for arranging feedback To store the state in the memory circuit. The functioning of such SMP (see Figure 5.1) is defined as an elementary automaton of the second kind, which has a complete system of inputs and outputs [6].

Thus, the essence of the method for designing an  $A_M$  strategy automaton for the entire MFIS consists in the definition of SMP with  $M$  states, the number of which corresponds to the number of  $r_e$  ( $M \geq r_e$ ) sets of  $e_i(\Delta)$  preserving input signals of the controlled MFIS  $A_y$ .

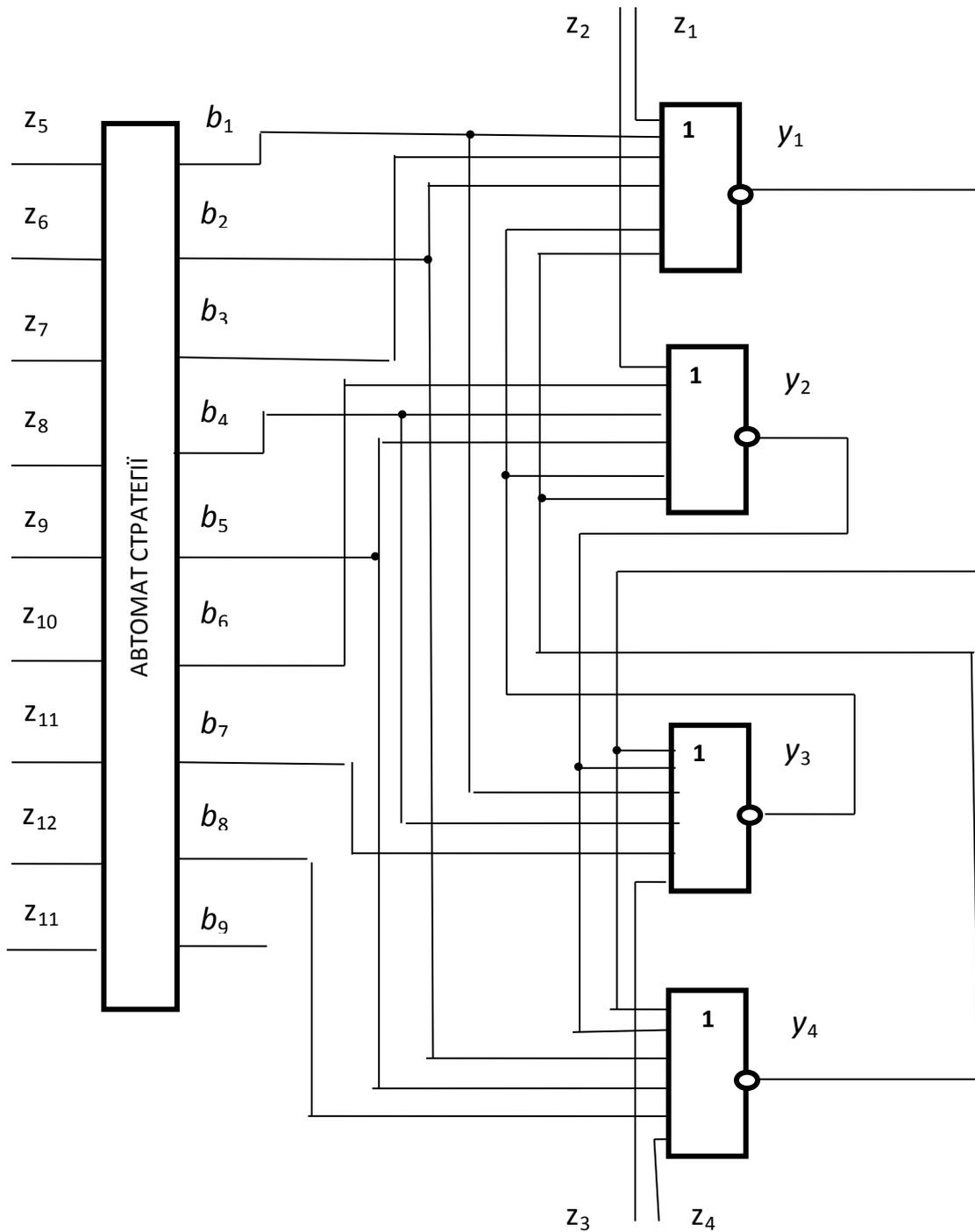
A two-level memory scheme with the  $A_M$  strategy automaton for the whole MFIS is synthesized with a controlled MFIS  $A_y$  in accordance with the structural diagram shown in Fig. 5.2. A two-level MUSP memory scheme consists of two MFIS (or one MFIS and one SMP). The input buses of the two MFIS can be combined into a common setting input bus of the MUSP, and the output nodes of the control MFIS (the strategy machine)  $A_M$  are respectively connected to the input bus of the controlled MFIS  $A_y$ , to which sets of  $e_j(\Delta)$  input signals are stored.

Table 5.2

Unambiguously set memory states

Input signal $x_i(t)$	The output signal $b_i(T)$ of the $A_M$ automaton									Memory status $A_i$
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$	$b_9$	
$x_0$	0	0	0	0	0	0	0	0	0	$A_0$
$x_1$	1	0	0	0	0	0	0	0	0	$A_1$
$x_2$	0	1	0	0	0	0	0	0	0	$A_2$
$x_3$	0	0	1	0	0	0	0	0	0	$A_3$
$x_4$	0	0	0	1	0	0	0	0	0	$A_4$
$x_5$	0	0	0	0	1	0	0	0	0	$A_5$
$x_6$	0	0	0	0	0	1	0	0	0	$A_6$
$x_7$	0	0	0	0	0	0	1	0	0	$A_7$
$x_8$	0	0	0	0	0	0	0	1	0	$A_8$
$x_9$	0	0	0	0	0	0	0	0	1	$A_9$

The interconnection between the output nodes of the  $A_M$  strategy machine and the input nodes of the controlled MFIS  $A_y$  is carried out by the input bus of the controlled MFIS  $A_y$  in accordance with the set of  $e_i(\Delta)$  inputs that are stored, which are determined during the mathematical analysis of the operation of the MFIS  $A_y$ . The setting input nodes of the MFIS  $A_y$  and MFIS  $A_M$  can be combined into a common setting input bus of a two-level memory scheme.



**Fig. 5.2.** Two-level class memory scheme  $L_N$

Thus, a two-level memory scheme consisting of a control SMP (automaton strategy)  $A_M$  for nine states and controlled by an MFIS  $A_y$  having nine conserved input signals is shown in Fig. 5.2.

## 5.7. The method of logical design of a multilevel scheme class $L^M$ memory with one strategy automaton

Let us consider the method of logical design of a two-level memory scheme A, consisting of a controlled MFIS  $A_y$ , in which there are two groups of elements of two BAs in each of them, and which can perceive nine sets of  $e_i(\Delta)$  preserving input signals, and the  $A_M$  strategy automaton (Fig. 5.2), which can generate nine  $b_j(T)$  sets of  $e_j(\Delta)$  input signals for the controlled MFIS  $A_y$ . The basis of the method of logical design is the organization of hierarchical links between the managed MFIS  $A_y$  and the  $A_M$  strategy automaton.

The interconnection of the output nodes  $b_j(T)$  of the strategy automaton  $A_M$  with the input nodes  $BA_j$  (OR-NON elements) MFIS  $A_y$  is performed according to the sets of  $e_i(\Delta)$  input signals determined during the mathematical simulation [3]. The unit values of the sets of eigenfrequency ( $\Delta$ ) storing  $e_i(\Delta)$  input signals of the MFIS  $A_y$  are identified with the corresponding output nodes  $b_j(T)$  of the  $A_M$  strategy automaton, reflecting its states.

The functional diagram of a two-level memory device A is shown in Fig. 5.2.

The number of links between the managed MFIS  $A_y$  and the  $A_M$  strategy machine is determined by the formula:

$$r_c = \prod_{i=2}^m (2^{R_i} - 1) - 1 = r_e - 1, \quad (5.9)$$

where  $i$  – is the  $i$ -th group of BA;

$m$  – is the number of  $BA_j$  groups in the controlled MFIS  $A_y$ ;

$R_i$  – is the number of  $BA_j$  in the  $i$ -th group of the controlled MFIS  $A_y$ ;

$r_e$  – is the number of sets of  $e_j(\Delta)$  inputs retained in the MFIS  $A_y$ .

The operation of a two-level memory device (Figure 5.2) can be described as follows. The memory circuit can receive simultaneously sets of setting  $x(t)$  input signals consisting of two input signal streams:  $x_y(t)$  of the controlled MFIS  $A_y$  and  $x_M(t)$  of the  $A_M$  strategy automaton. The signal  $x_M(t)$  establishes the automaton of the  $A_M$  strategy to the state  $a_i$ , and the signal  $x_y(t)$  is controlled by the MFIS  $A_y$  to the simultaneously (although it is possible and sequentially) during the transition time  $2\tau_e$  (where  $\tau_e$  – is the signal delay at one BA). The output signals appear at the output nodes of the memory circuits after the time  $\tau_e$  after the stable setting signals  $x_y(t)$  and  $x_M(t)$  appear on the input nodes. Thus, even if the input signal with a minimum duration of  $x_M(t)$  is  $2\tau_e$ , then its duration is sufficient for the appearance of the  $A_M$  strategy at the output nodes (through the time  $\tau_e$ ) of the set of the  $e_i(\Delta)$ . The action of the minimum input signal  $x_y(t)$  (the duration of which is  $2\tau_e$ ). With the simultaneous action of the input signals that establish  $x_y(t)$  and the input signals that preserve  $e_i(\Delta)$ , the signal is absorbed by the input signal  $x_y(t)$  retaining the  $e_i(\Delta)$ . The minimization of the number of nodes of the set of  $x_y(t)$  input signals can be explained by the fact that, in fact, only the  $BA_j$  of one group of the MFIS  $A_y$ , when memorizing stable states, should have active values of the output signals that are equal to one, and all output signals  $BA_j$  of other groups of the MFIS  $A_y$  should be inactive. Values that are equal to logical zero. In this case, the number of establishing input nodes can be reduced to the number of  $m$  groups of the MFIS  $A_y$ .

Thus, the number of  $z_i$  ( $i = 1, 2, 3, 4$ ) input signals in the MFIS  $A_y$ . (Fig. 5.2) in Table. 5.2 can be shortened to two, and the number of input signals that set  $x_y(t)$  is up to three:  $x_{y1}$  ( $z_1 = z_2 = 1, z_3 = z_4 = 0$ ),  $x_{y2}$  ( $z_1 = z_2 = 0, z_3 = z_4 = 1$ ),  $x_{y3}$  ( $z_1 = z_2 = z_3 = z_4 = 1$ ). Consider the sets of  $x_i(t)$  input signals for a two-level memory scheme in Table. 5.3.

The setting input  $x_{y3}$  is forbidden in the deterministic mode of operation of the memory circuit  $A_y$ , since it is not stored for any set of  $e_i(\Delta)$  preserving input signal.

The number of input signals can be reduced by combining the input nodes  $z_1$  and  $z_2$  and also  $z_3$  and  $z_4$ . At the same time, the number of input nodes of the two-level memory scheme is reduced to 11.

Table 5.3

Sets of setting input signals ( $x_y(t)$ )

$z_i$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$	$x_{16}$	$x_{17}$	$x_{18}$
$z_1$	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
$z_2$	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
$z_3$	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
$z_4$	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
$z_5$	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
$z_6$	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
$z_7$	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
$z_8$	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1
$z_9$	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1
$z_{10}$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1
$z_{11}$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1
$z_{12}$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1
$z_{13}$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0

The input signals  $x_i(t)$  unambiguously establish the states of the two-level memory scheme. The combined states of  $a_i$  in a two-level memory scheme consist of two states of the MFIS  $A_y$  and MFIS  $A_M$ .

The maximum number  $M_{\max}$  of memory states of memory devices depends on the number of used groups and the number of re sets storing  $e_j(\Delta)$  of the input signals of the MFIS  $A_y$ , equal to the number of states of the  $A_M$  strategy machine, and is calculated by the formula ( $M_{\max} = m_y r_e$ ).

The set of  $e(\Delta)$  input signals that preserve the two-level memory scheme, at which all its states are stored, has the same value equal to logical zero at all its input nodes ( $z_i = 0$ ).

Input signals  $x_i(t)$  and the unified states  $A_i$  of the two-level memory scheme are uniquely determined. 5.4.

The deterministic mode of operation of a two-level memory device for speed is the same as that of an *RS*-flip-flop. The number of  $L$   $BA_i$  (logical elements - BA) per one storage state in comparison with the *RS*-trigger ( $L = 1$ ) decreases and for a two-level memory circuit is  $L \approx 0.7$ , which means a reduction in the hardware costs of logical elements per one memorized state.

In the general case, a two-level memory scheme (Figure 5.2) can be considered as an SME for 18 states. This is due to the fact that the memory remembers all its states with one set of  $e(\Delta)$  preserving input signal. In addition, the  $A_M$  strategy machine remembers the general information, and the controlled MFIS is  $A_y$ -particular, whose state storage structure can be changed.

Table 5.4

Established combined states of a two-level memory scheme

$x_i$	Output signals $BA_i$													States $A_i$
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$	$b_9$	$y_1$	$y_2$	$y_3$	$y_4$	
$x_1$	1	0	0	0	0	0	0	0	0	0	0	0	1	$A_1$
$x_2$	0	1	0	0	0	0	0	0	0	0	0	1	0	$A_2$
$x_3$	0	0	1	0	0	0	0	0	0	0	0	1	1	$A_3$
$x_4$	0	0	0	1	0	0	0	0	0	0	0	0	1	$A_4$
$x_5$	0	0	0	0	1	0	0	0	0	0	0	1	0	$A_5$
$x_6$	0	0	0	0	0	1	0	0	0	0	0	1	1	$A_6$
$x_7$	0	0	0	0	0	0	1	0	0	0	0	0	1	$A_7$
$x_8$	0	0	0	0	0	0	0	1	0	0	0	1	0	$A_8$
$x_9$	0	0	0	0	0	0	0	0	10	0	0	1	1	$A_9$
$x_{10}$	1	0	0	0	0	0	0	0	0	0	1	0	0	$A_{10}$
$x_{11}$	0	1	0	0	0	0	0	0	0	0	1	0	0	$A_{11}$
$x_{12}$	0	0	1	0	0	0	0	0	0	0	1	0	0	$A_{12}$
$x_{13}$	0	0	0	1	0	0	0	0	0	1	0	0	0	$A_{13}$
$x_{14}$	0	0	0	0	1	0	0	0	0	1	0	0	0	$A_{14}$
$x_{15}$	0	0	0	0	0	1	0	0	0	1	0	0	0	$A_{15}$
$x_{16}$	0	0	0	0	0	0	1	0	0	1	1	0	0	$A_{16}$
$x_{17}$	0	0	0	0	0	0	0	1	0	1	1	0	0	$A_{17}$
$x_{18}$	0	0	0	0	0	0	0	0	1	1	1	0	0	$A_{18}$

A two-level memory scheme can perform unambiguous and enlarged transitions to the MFIS  $A_y$  with respect to two variables  $x$  and  $e$ , which triggers can not. In addition, an 18-state SMP has 18 input and 18 output nodes, and also uses 18 logical elements. In the case of a two-level memory scheme, 13 output and 11 input nodes are used, and 13 logical elements are used, which is much smaller than in the SMP. The number of internal connections between  $BA_i$  (logical elements) in SMP for 18 states is  $17 \cdot 18 = 306$ , and in a two-level memory scheme -  $20 + 8 \cdot 9 = 92$ .

Thus, a two-level class memory scheme has significant advantages, both in terms of hardware and functional characteristics, in comparison with SMP .

The  $A_M$  strategy machine can in its turn be multi-level, which allows reducing the restrictions on basic automata by the number of input nodes (NAND and / or NOR elements) [14].

## **5.8. The method of logical design of a multi-level class $L_N^B$ memory scheme with a strategy automaton for each group of multilevel memory schemes**

The principle of the structural organization of elementary multi-level memory schemes of a class  $L_N^B$  with the strategy automaton for each group of the MFIS consists in dividing them into control and manageable multifunctional memory circuits (MFIS) [3], which are interconnected as follows:

- The BA of each  $i$ -th group of the controlled MFIS  $A_i$ , the number  $q_i$  of which is greater than one ( $q_i > 1$ ), are connected through the input bus to which the sets of  $e_j(\Delta)$  saving signals are fed to the output lines of one separate MFIS  $A_k$  or SMP ( $K = 1, 2, \dots, i-1$ );
- the control input and output buses of the MFIS  $A_i$  ( $i = 1, 2, \dots, N$ ) are respectively connected to common input and output buses of multi-level memory circuits.

A multilevel scheme of class  $L_N^B$  memory has the following characteristic properties in comparison with multistable memory schemes:

- it is able to change the memory structure of the states of memory circuits;
- establish stable states with fewer input signals, which are fed to the control part of the input nodes of the memory circuit;

- use fewer output signals;

- use fewer internal connections between the elements of the scheme;

- use at least two levels of multifunctional memory circuits (MFIS), each of which consists of  $n_j$  ( $n_j \geq 2$ ) logical  $K$  - input NAND elements (NOR) with load capacity  $P_l$  divided into  $m_K$  groups, where  $2 \leq m_K \leq n_j$  ( $m_K \leq \frac{M}{j}$ ;  $M$  – is the number

$$\prod_{i=k+1}^M m_i$$

of memory states of the memory scheme),  $q_{j,k}$  of the NAND elements (NOR), where

$$1 \leq q_{j,k} \leq \lceil \log_2 \sqrt{\frac{M}{j} \prod_{i=k}^M m_i} + 1 \rceil;$$

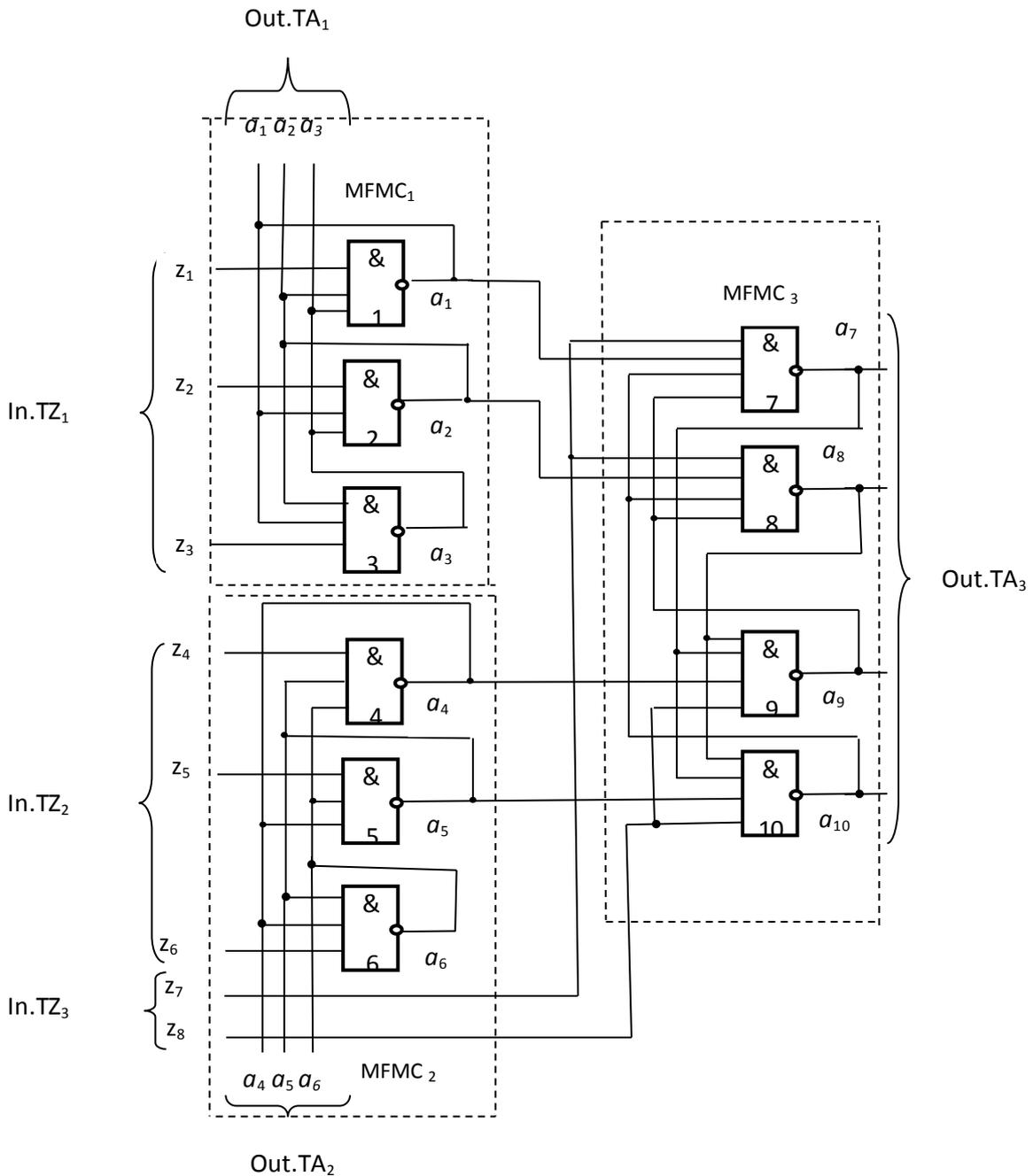
- the outputs of the NAND (NOR) elements, the  $k$ -th group ( $q_k > 1$ ) of the MFIS  $j$ , through the additionally introduced  $T$ -input elements AND (OR) with the load capability  $P_2$  or the  $k$ -th group ( $q_k = 1$ ) are connected directly to one of the inputs of the NAND (NOR) elements of other groups of the given MFIS  $j$ ;

- MFIS  $j$  are located in  $j$  levels, the upper level of the MFIS  $A_j$  contains  $n_j$  elements of the MFIS  $A_j$  divided into at least two groups  $m_j$  ( $j = 2$ ) with respect to  $q_{j,k}$

elements ( $\frac{n_j + 1}{m_j} \geq q_{j,k} \geq \frac{n_j}{m_j}$ );

- one input of each NAND (NOR) element of the  $k$ -th group ( $q_{j,k} > 1$ ) of the junction-level MFIS  $j$  is connected to one information input of  $j$ -th group of the memory circuit  $TZ_j$ , and the outputs of the NAND elements (NOR) MFIS  $A_j$  are connected to the information outputs of the  $j$ -th group of the memory circuit  $TA_j$ , respectively;

- the inputs of the NAND (NOR) elements of each  $k$ -th group ( $q_{j,k} > 1$ ) of the junction level MFIS  $j$  are connected respectively to the outputs of the  $k$ -th group NAND (NOR) elements ( $q_{s,k} = 1$ ) the lower MFIS  $j_s$ , except for the last output, creating links between the levels of the memory circuit.



**Fig. 5.3.** Multilevel scheme of class  $L_N^B$  memory.

The total number of  $M$  memory states of a multilevel memory scheme is calculated by the formula: 
$$M = \prod_{i=1}^J m_i .$$

The total number  $S_{внеш.с}$  of external links is given by:

$$S_{внеш.с} < 2n. \quad (5.10)$$

The total number  $S_{\text{sym.c}}$  of internal connections between elements is determined by the relation:

$$S_{\text{sym.c}} < n \times (n-1). \quad (5.11)$$

In Fig. 5.3 shows the functional multilevel scheme of class  $L_N^B$  memory.

The scheme consists of three MFIS<sub>*j*</sub> (MFIS<sub>1</sub>, MFIS<sub>2</sub>, MFIS<sub>3</sub>), which are located on two levels ( $j = 2$ ). At the upper level is the managed MFIS<sub>3</sub>, which has four NAND elements ( $n = 4$ ), and is divided into two groups ( $m = 2$ ), two elements ( $q = 2$ ) in each group. The control circuits of the MFIS<sub>1</sub> and the MFIS<sub>2</sub>, which are located at the first (lower) level, are designed to control the storage structure of the states in groups ( $q = 2$ ) of the upper MFIS<sub>3</sub> scheme and have three NAND elements ( $n = 3$ ), which are divided into three group ( $m = 3$ ) for one element ( $q = 1$ ) in each. MFIS<sub>1</sub> - MFIS<sub>3</sub> are built on NAND gates.

A distinctive structural feature of the memory scheme is the multi-level memory, where each structural  $j$ -th level consists of a stable MFIS<sub>*j*</sub>, and the inputs of those elements belonging to the  $k$ -th group ( $q_{j,k} > 1$ ) to the MFIS<sub>*s*</sub> ( $s = j-1$ ), which are intended for controlling the structure of storing states in groups ( $q_{j,k} > 1$ ) of the upper.

A distinctive functional feature of the device is the operation of the controlled circuits of the MFIS<sub>*j*</sub> of the upper levels in several different subsets of their states that determine the set of states of the MFIS<sub>*s*</sub> ( $s = j - 1$ ) schemes of the lower levels. This allows you to change the display of incoming and outgoing information in managed MFIS<sub>*j*</sub> circuits, to redirect output information to a specific direction, and to set the states of the memory circuit by a smaller number of input signals arriving only to a part of the input nodes of the device.

In this case, the functional diagram of the multi-stage memory device stores 18 states and has 8 input and 10 output nodes, which in total make up 18 external nodes, is smaller than in the SMP by 18 nodes (twice), as well as 24 internal connections between all the elements AND-NOT, which is less than 12.75 times that available in SMP  $18 \times 17 = 306$ .

The functional mode considers the operation of the memory scheme in two modes: multifunctional (Table 5.5) and enlarged (Table 5.6). Setting sets of  $x_i$  input signals in the functional mode of operation of the multistage device are presented in Table 5.5.

The multifunctional mode considers the operation of the controlled MFIS<sub>*j*</sub> (*j* = 3) of the upper level in various subsets of states, which are remembered for the corresponding MFIS<sub>*s*</sub> states, lower levels.

Table 5.5

Installing sets of input signals

Sets input signals $x_i$	Value input knots $z_i$	Weekend value knots $a_i$	States scheme $A_i$
$x_i$	$z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8$	$a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}$	$A_i$
$x_0$	0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1	$A_0$
$x_1$	1 0 0 1 0 0 1 0	0 1 1 0 1 1 1 0 1 1	$A_1$
$x_2$	1 0 0 1 0 0 0 1	0 1 1 0 1 1 1 1 1 0	$A_2$
$x_3$	1 0 0 0 1 0 1 0	0 1 1 1 0 1 1 0 1 1	$A_3$
$x_4$	1 0 0 0 1 0 0 1	0 1 1 1 0 1 1 1 0 1	$A_4$
$x_5$	1 0 0 0 0 1 1 0	0 1 1 1 1 0 1 0 1 1	$A_5$
$x_6$	1 0 0 0 0 1 0 1	0 1 1 1 1 0 1 1 0 0	$A_6$
$x_7$	0 1 0 1 0 0 1 0	1 0 1 0 1 1 0 1 1 1	$A_7$
$x_8$	0 1 0 1 0 0 0 1	1 0 1 0 1 1 1 1 1 0	$A_8$
$x_9$	0 1 0 0 1 0 1 0	1 0 1 1 0 1 0 1 1 1	$A_9$
$x_{10}$	0 1 0 0 1 0 0 1	1 0 1 1 0 1 1 1 0 1	$A_{10}$
$x_{11}$	0 1 0 0 0 1 1 0	1 0 1 1 1 0 0 1 1 1	$A_{11}$
$x_{12}$	0 1 0 0 0 1 0 1	1 0 1 1 1 0 1 1 0 0	$A_{12}$
$x_{13}$	0 0 1 1 0 0 1 0	1 1 0 0 1 1 0 0 1 1	$A_{13}$
$x_{14}$	0 0 1 1 0 0 0 1	1 1 0 0 1 1 1 1 1 0	$A_{14}$
$x_{15}$	0 0 1 0 1 0 1 0	1 1 0 1 0 1 0 0 1 1	$A_{15}$
$x_{16}$	0 0 1 0 1 0 0 1	1 1 0 1 0 1 1 1 0 1	$A_{16}$

$x_{17}$	0 0 1 0 0 1 1 0	1 1 0 1 1 0 0 0 1 1	$A_{17}$
$x_{18}$	0 0 1 0 0 1 0 1	1 1 0 1 1 0 1 1 0 0	$A_{18}$

In this mode, the controlled MFIS<sub>j</sub> functions in different subsets of its states in accordance with the MFIS<sub>s</sub> states, which are capable of changing the display of information entering the output (Table 5.6). Transitions in the controlled MFIS<sub>j</sub> from one state to another of one subset are performed under the influence of the setting sets  $x_i$  of the input signals.

Enlarged mode considers the change in the states of all MFIS<sub>j</sub> circuits when input sets of input signals are input only on the input nodes of the lower-level controlled MFIS<sub>s</sub>. In this mode, the transitions to the MFIS<sub>s</sub> of the lower levels from one state to another state are carried out under the influence of the set of input signals  $x_i$ , and the enlarged transitions to the MFIS<sub>j</sub> of the upper levels from one state to another are effected by the internal sets of  $e_j$  saving inputs that come from the MFIS<sub>s</sub> lower levels of certain elements of  $i$ -groups that have more than one BA in the MFIS<sub>j</sub> of the upper levels.

Table 5.6. Enlarged transitions in a multilevel memory scheme

Sets input signals $z_i$	Consolidated states of the memory scheme $A_i$
$z_1 z_2 z_3 z_4 z_5 z_6$	$A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11} A_{12} A_{13} A_{14} A_{15} A_{16} A_{17} A_{18}$
1 0 <u>0</u> 1 0 <u>0</u>	$A_1 A_2 A_1 A_2$
1 0 <u>0</u> <u>0</u> 1 0	$A_3 A_4 A_3 A_4$
1 0 <u>0</u> <u>0</u> <u>0</u> 1	$A_5 A_6 A_5 A_6$
0 1 0 1 0 <u>0</u>	$A_7 A_8 A_7 A_8$
0 1 0 <u>0</u> 1 0	$A_9 A_{10} A_9 A_{10}$
0 1 0 <u>0</u> <u>0</u> 1	$A_{11} A_{12} A_{11} A_{12}$
0 <u>0</u> 1 <u>1</u> 0 <u>0</u>	$A_{13} A_{14} A_{13} A_{14}$
0 <u>0</u> 1 0 1 0	$A_{15} A_{16} A_{15} A_{16}$
0 <u>0</u> 1 0 <u>0</u> 1	$A_{17} A_{18} A_{17} A_{18}$

Thus, the proposed scheme is a single multi-level memory scheme that has the ability to change the display of information in the MFIS<sub>*j*</sub> of the upper levels without the influence of setting the input signals due to internal links between the steps. It has fewer internal connections between the elements and can change the state of the entire device with fewer input signals, which in principle is impossible in triggers and SMP.

## 5.9. Classification of basic memory circuits

In connection with the development of a large number of asynchronous basic memory circuits, let us cite their classification (Figure 5.4). The most common case of asynchronous basic memory schemes is the basic MUSP memory scheme, which is created from the MFIS and single-phase SMP. The MFIS is in turn a more general case with respect to single-phase SMP, the private (minimal) case of which is the *RS*-type asynchronous trigger. A generalization of the basic memory schemes is shown in Fig. 5.4.

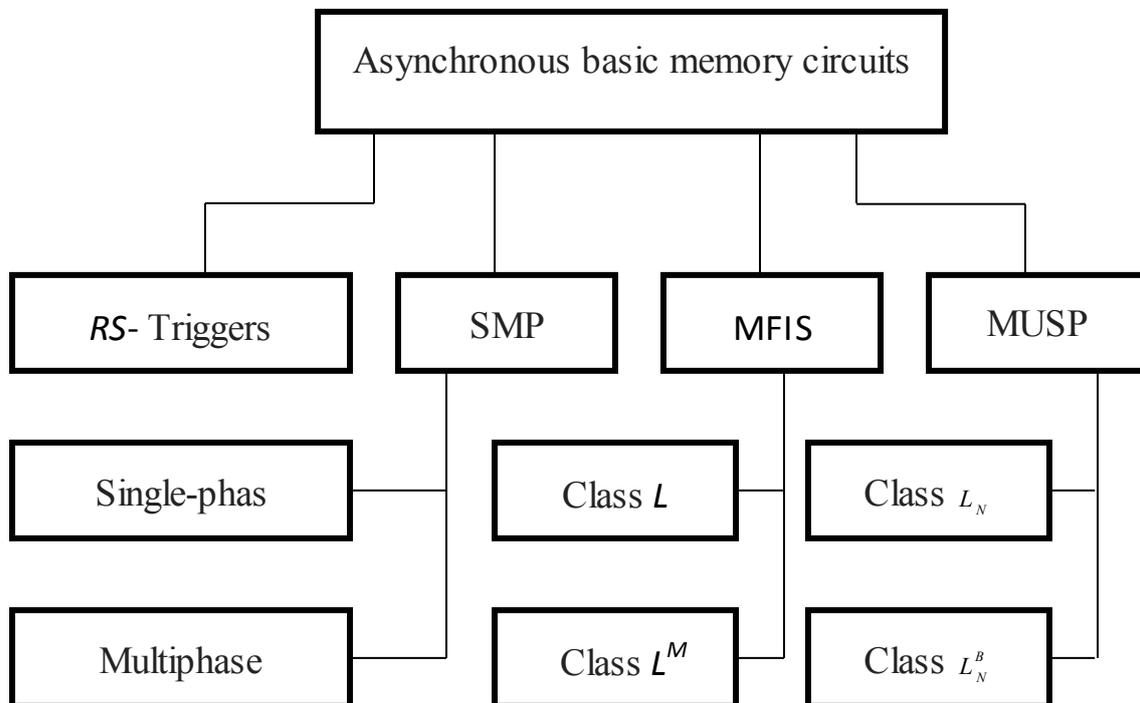
Due to the fact that the MUSP remembers all its states in one set preserving the  $e(\Delta)$  input signal as an SMP, it is advisable to compare and characterize them in one set of parameters:

- the number of logical elements that are necessary for building memory circuits that stores  $M$  states;
- operating switching frequency ( $F_p$ );
- maximum load capacity on outputs ( $n_Q$ );
- the number of internal links ( $S_{\text{внутр.с.}}$ );
- the number of external links ( $S_{\text{внеш.с.}}$ );
- number of elements per state ( $L$ );
- functionality that allows you to  $r_e$  build the memory structure of the states of the memory scheme

## 5.10. Comparison of the parameters of the basic memory circuits

### 5.10.1. Determination of the number of logical elements, necessary for building memory circuits

A single-phase SMP storing  $M$  states should consist of  $M$ -input logic elements NOR (NAND) with the load capacity  $M$  (Figure 5.1).



**Fig. 5.4.** Classification of basic memory circuits

An MUSP class  $L_N^B$ , that stores  $M$  states with  $N = 2$  and  $B = 2$  must consist of  $K$ -input NOR (NAND) gates with a load capacity of  $P_1$  (Figure 5.3).

Compare the SMP and the MUSP, which store 18 states:

- the number of NOR (NAND) logical elements in the SMP is used 18;
- the number of NOR (NAND) logical elements in the MUSP is used 10.

Thus, the MUSP uses a logical number of NOR (NAND) when storing 18 states in 0,56 times less. With the increase in storage states, the difference in the use of logical elements increases, and the relationship between MUSP and SMP decreases and becomes less than 0.36, which reflects a sharp decrease in hardware costs in the MUSP compared to SMP .

### **5.10.2. Determination of the maximum possible load Memory output capabilities**

A single-phase SMP memory that stores  $M$  states, which consists of  $M$ -input logic elements NOR (NAND), has the maximum possible load capability on the outputs of the memory scheme equal to 1, with the possibility of NOR (NAND) Have a load capacity equal to  $M$ . For  $M = 18$  and  $n_Q = 1$  (Figure 5.1).

An MUSP class  $L_N^B$ , that stores  $M$  states with  $N = 2$  and  $B = 2$  must consist of  $K$ -input logic elements NOR (NAND) with a load capability equal to  $M$ . In this case,  $n_Q = M-3 = 15$  (Fig. 5.3).

Let's compare the SMP and the MUSP, which store 18 states with the capacity for the outputs of the memory circuits:

- the number of  $n_Q$  in the SMP with the output capability is 1;
- the number of  $n_Q$  in the MUSP with the output capability is 15.

Thus, in the MUSP with 14 outputs more can connect the logical elements NOR (NAND) than with the SMP .

### **5.10.3. Determining the number of internal connections of memory circuits**

A single-phase SMP storing  $M$  states must consist of  $M$ -input NOR (NAND) gates that have a number of internal links  $S_{\text{внутр. с.}} = M \times (M - 1) = 18 \times 17 = 306$  (Figure 5.1).

An MUSP class that stores  $M$  states with  $N = 2$  and  $B = 2$  must consist of  $K$ -input logic elements NOR (NAND) and has a number of internal links  $S_{\text{внут. c.}} = 24$  (Figure 5.3).

Compare with SMP and MUSP, which store 18 states, the number of internal connections is  $S_{\text{внут. c.}}$  memory circuits:

- the number  $S_{\text{внут. c.}}$  is too in the SMP of internal communications is 306;
- the number  $S_{\text{внут. c.}}$  in the MUSP of internal communications is 24.

Thus, the MUSP has 12.75 times less internal communication  $S_{\text{внут. c.}}$ , Than with SMP, which is very important in the layout of the topology of integrated circuits.

#### **5.10.4. Determination of the number of external links of memory circuits**

A single-phase SMP storing  $M$  states should consist of  $M$ -input logic elements NOR (NAND), which have the number of external links  $S_{\text{внеш. c.}} = M + M = 18 + 18 = 36$  (Figure 5.1).

An MUSP class that stores  $M$  states with  $N = 2$  and  $B = 2$  must consist of  $K$ -input logic elements NOR (NAND), have the number of external links  $S_{\text{внеш. c.}} = 18$  (Figure 5.3).

Compare the SMEs and MUSPs that store 18 states, the number of external links  $S_{\text{внеш. c.}}$  memory:

- The number  $S_{\text{внеш. c.}}$  of external links in MSPs is 36;

The number  $S_{\text{внеш. c.}}$  of in the MUSP links is 18.

Thus, the MUSP has twice the number of external links  $S_{\text{внеш. c.}}$ , than with SMP.

### 5.10.5. Determining the number of elements per state memory circuits

A single-phase SMP storing  $M$  states should consist of  $M$ -input logic elements NOR (NAND), has the number of elements per one state  $L = 1$  (Figure 5.1). An MUSP class that stores  $M$  states for  $N = 2$  and  $B = 2$  must consist of  $K$ -input logic elements NOR (NAND) has a number of elements per state  $L = 0.56$  (Figure 5.3).

Compare the SMP and MUSP that store 18 states, the number of elements per state of  $L$  memory circuits:

- the number  $L$  in the SMP is 1;
- the number of  $L$  in the MUSP is 0.56.

Thus, the MUSP has about 1.78 times less number of elements per state  $L$  than for SMP, which characterizes the hardware costs for one memory state of the memory scheme.

### 5.10.6. Comparison of operating switching frequency and memory options

The operating frequency of the  $F_P$  MUSP is the same in comparison with SMP when using the class  $L$ . MFIS. When using the MFIS class  $L_N$ , the operating switching frequency is reduced by a factor of 1.5.

Functional capabilities of the MFIS in comparison with SMP have the advantage that they are able to reconstruct the storage structure of the MIPE states in the MUSP in a single machine cycle during operation. These additional functionality of the MUSP allow simultaneous processing of common (control information in the strategy automaton) and private (in the MFIS) information, which is basically impossible to do in SMP (flip-flops).

Comparable data of memory circuits will be considered in a tabular form (Table 5.7).

Table 5.7

Parameters of basic memory circuits that store 18 states

Parameters	Single-phase MMC	MUSP
$F_p$	12,5 МГц	12,5 МГц
$n_Q$	1	10
$S_{\text{внутр.с.}}$	306	24
$S_{\text{внеш.с.}}$	36	18
$L$	1	0,56

## 5.11. The questions of building reliable devices on multi-level memory circuits

### 5.11.1. The reliability of multi-level memory circuits

The number of memorized states  $Q$  of a two-level MUSP is determined by the formula:

$$Q = m \prod_{i=1}^m (2^{R_i} - 1), \quad (5.12)$$

where  $m$  – is the number of groups of logical elements OR-NOT (AND-NOT) in the MFIS ;

$R_i$  – is the number of logical elements OR-NOT (AND-NOT) in the  $i$ -th group.

As can be seen from the MUSP (Figure 5.3), MFIS is a multifunctional memory circuit (Figure 5.5), which stores 6 states and has 9 preserving  $e(\Delta)$  input signals capable of reconstructing the state storage structure [3]. The generation of  $e(\Delta)$  input signals for each group of MFIS<sub>3</sub> elements is performed by MFIS<sub>1</sub> and MFIS<sub>2</sub>. The MUSP is capable of storing 18 states.

The number of memorized states in the  $i$ -th group is determined depending on the number of  $R_i$ , which use the elements NOR (NAND) in the  $i$ -th group. Elements

in the  $i$ -th group are connected in the sense of reliability in parallel. The number of  $i$ -groups in the MFIS ranges from 2 to  $m$  ( $2 \leq i \leq m$ ). If we take the minimum number of groups, as shown in Fig. 5.5 and Fig. 5.3, then their interaction with each other forms a consistent connection in the sense of reliability.

The failures of the elements of the  $i$ -th group do not affect the functioning of the remaining elements of this  $i$ -th group. However, if the failing element at the output node has an active output signal value that uniquely establishes inverse values on the outputs of the elements of the other groups, then such a failure is catastrophic for the functioning of the entire MFIS. In the future, we will consider non-catastrophic failures of elements whose output signals of elements do not affect the functioning of elements of other groups.

The minimum number of elements necessary for the functioning of the memory scheme, with one NOR element (NAND) in the group is equal to the number of  $m$  groups. This, so-called, multi-stable memory circuits [4-5]. The MFIS structure can be considered as a memory scheme, which is reserved in each  $i$ -th group  $(R_i - 1)$  element. In this case, the MFIS is considered as a scheme consisting of  $m$  workers and  $m$   $(R_i - 1)$  reserve elements. All  $N = m R_i$  elements can fail.

The number  $M$  of memorized states of the MFIS can fluctuate within

$$m \leq M \leq \sum_{i=1}^m (2^{R_i} - 1), \quad (5.13)$$

If a  $j$ -th failure occurred in the  $i$ -th group of the MFIS by the time  $t$ , then the number  $K_i$  of the remembered states of the  $i$ -th group will change and form the memorized states. In other words, if all  $R_i$  elements fail, the  $i$ -th group falls into the failure state and no changes occur in this  $i$ -th group. To assess the performance of the MFIS, which in case of failures  $(R_i - 1)$  of the elements in each group is converted into a multistable memory circuit, which stores all states with one inactive input signal [4], it must be tested. It is also convenient to use to evaluate the performance of the MFIS

with the number of  $r_e$  blocks of the  $\pi_i$  states, which are stored with the corresponding failures of the elements in the  $i$ -th groups. The number of stored  $r_e$  states is within

$$1 \leq r_e \leq \prod_{i=1}^m (2^{R_i} - 1). \quad (5.14)$$

If a  $j$ -th failure occurs in the  $i$ -th groups by the time  $t$ , then the MFIS stores the number of state blocks  $\pi_i$ ,

$$r_e = \prod_{i=1}^m (2^{R_i - j} - 1). \quad (5.15)$$

In terms of reliability, the MFIS is a parallel-sequential scheme in which the elements of each  $i$ -th group represent a parallel scheme of elements with the same parameters, and the groups of elements are connected to each other in series.

Each  $j$ -th element in the general case is characterized by the failure rate  $\lambda_j(t)$  and the probability of failure-free operation  $P_j(t) = \exp \left[ - \int_0^t \lambda_j(t) dt \right]$ . The probability of failure-free operation of the  $i$ -th group of the MFIS as a whole is determined by the formula

$$P_{\varphi}^i(t) = 1 - (1 - P_j(t))^{R_i}. \quad (5.16)$$

The probability of failure-free operation of the MFIS on the interval  $[0, 1]$  with different number of elements in the group can be determined by the formula:

$$P(t) = \prod_{i=1}^m [1 - (1 - P_j(t))^{R_i}]. \quad (5.17)$$

With the same number of elements in the group, the probability of trouble-free operation of the MFIS is determined by the formula:

$$P(t) = [1 - (1 - P_j(t))^{R_i}]^m, \text{ npu } R_i = \text{const} . \quad (5.18)$$

If the fail-safe operation time of an element is subject to an exponential law with a parameter (failure rate)  $\lambda_j$ , then it  $P_j(t)$  is determined by the formula:

$$P_j(t) = e^{-\lambda_j t} . \quad (5.19)$$

In this case, for successive connection, the probability of failure-free operation can be expressed in terms of the failure rate as follows:

$$P(t) = \prod_{i=1}^m P_i(t) = P_i^m(t), \text{ npu } R_i = 1; \quad (5.20)$$

$$P(t) = e^{-\sum_{i=1}^m \lambda_j t} = e^{-m \lambda_j t} . \quad (5.21)$$

The mean time between failures of the MFIS by the known  $P(t)$  is determined by the formula:

$$T_{cp} = \int_0^{\infty} P(t) dt . \quad (5.22)$$

The mean time to failure of the MFIS at  $R_i = 1$  ( $R_i = \text{const}$ ) is determined by the formula:

$$T_{cp} = \frac{1}{m \lambda_j} . \quad (5.23)$$

For the failure rate of the elements  $\lambda_j = 1 \cdot 10^{-7} / u$ , the mean time to failure of the MFIS for  $R_i = 1$  in all  $i$ -th groups ( $m = 2$ ) is equal to

$$T_{cp} = \frac{1}{2\lambda_j} = 0,5 \cdot 10^7 u.$$

For  $R_i = 2$  and  $m = 2$ , the probability of failure-free operation of the MFIS is determined by the formula:

$$P(t) = [1 - (1 - e^{-\lambda_j t})^2]^2 = 4e^{-2\lambda_j t} - 4e^{-3\lambda_j t} + e^{-4\lambda_j t}.$$

$$\text{Then } T_{cp} = \int_0^{\infty} (4e^{-2\lambda_j t} - 4e^{-3\lambda_j t} + e^{-4\lambda_j t}) dt = \frac{4}{2\lambda_j} - \frac{4}{3\lambda_j} + \frac{1}{4\lambda_j} = \frac{11}{12\lambda_j}.$$

At  $R_i = 3$  and  $m = 2$ , the probability of failure-free operation of the MFIS is determined by the formula:

$$P(t) = [1 - (1 - e^{-\lambda_j t})^3]^2 = e^{-4\lambda_j t} (e^{-8\lambda_j t} - 12e^{-7\lambda_j t} + 66e^{-6\lambda_j t} - 114e^{-5\lambda_j t} + 387e^{-4\lambda_j t} - 468e^{-3\lambda_j t} + 414e^{-2\lambda_j t} - 216e^{-\lambda_j t} + 81).$$

In this case,  $T_{cp}$ , defined by formula (6.23), is equal to  $20,5 \cdot \frac{1}{\lambda_j}$ .

Thus, with an increase in the number of elements in the groups, the value of the mean time between failures increases, indicating an increase in the reliability of the MFIS as a memory scheme in comparison with single-phase multistable memory circuits.

It should be noted that as the number of groups with the same number of  $R_i$  elements increases, the value of the mean time between failures decreases, indicating that the most preferable in terms of improving reliability are MFIS with two groups with  $R_i > 1$  elements in each of them.

### 5.11.2. The issues of the survivability of multilevel memory schemes

At present, super-large integrated circuits (VLSIs) are built with the expectation of 100% suitability of all components of the circuit. Increasing the number of components and the very area of the VLSI crystal, increasing the length of the tires and reducing the width of their widths, naturally, increase the probability of failure of components and the appearance of breaks in their connections. This leads to a significant VLSI marriage and to a catastrophic failure of them during operation.

In order to improve the reliability of the operation of systems from unreliable elements, multiple backups, distributed network systems, etc., in which the outputs of a whole block or device are faulty, is determined by diagnostic programs and are not reflected catastrophically on the operation of the entire system as a whole. In addition, the unreliability of the basic elementary binary memory scheme, which is used in almost all digital VLSIs [4-13], is noted.

The use of two-level memory devices makes it possible to build the assumption that solving the problem of improving the reliability of memory and building a VLSI without 100% of the validity of all components of the memory scheme is possible. Such an assumption is based on the property of a multilevel memory structure with a multifunctional organization system to work in one of the defined subsets  $\pi_j$  of its states with corresponding input signals that preserve  $e_j(\Delta)$ .

For certain faults in the MFIS logic elements or in their links to other elements, it can be assumed that some of the elements fail and, thereby, narrow the scope of the MUSP. However, they do not completely disable it, as an element of memory. In this case, the MUSP functions in bounded subsets of the entire set  $Q$  of its states. The use of partially working memory increases the viability of the memory device, and, consequently, its reliability. The MUSP has the potential to function in limited subsets of its states with partial damage to elements

The MFIS uses elements whose individual output signals are active signals for other groups. Suppose that the main malfunction of the elements is the appearance at the output nodes of a constant value equal to logical zero. This assumption is quite plausible, if we take into account that when the input node breaks in the logic elements of integrated circuits, the value of the input signal is perceived to be equal to the logical one. Consequently, the output signal of an element having a broken input, acquires a constant value equal to logical zero. In this case, such an element will simply not participate in the memorization of the states of this group, and the characteristic function of the  $i$ -th group ( $K_i = 2^{R_i} - 1$ ) will decrease its value by one, i.e. Becomes equal  $K_i = 2^{R_i-1} - 1$ , where  $R_i$  – is the number of logical elements in the  $i$ -th group.

In case of faults in a whole group of logic elements, the memory circuit will function as a storage device if the number of operable remaining groups is at least or equal to 2 (ie  $m \geq 2$ ) and in each MFIS group at least one logical element.

It is clear from the law of the operation of the MUSP [64] that the MFIS is able to function with various input  $e_i(\Delta)$  inputs from the  $A_M$  strategy automaton in certain subsets  $\pi_j$  of its states. In non-catastrophic malfunctions, the MFIS narrows the range of its states in which it is still capable of operating. Malfunctions in the elements can be catastrophic if the output of the logic element is set to an active output signal whose value is equal to the logical one. In this case, the entire MFIS is out of order.

Thus, when synthesizing an MUSP, it is possible to take into account in advance the issues of reliability synthesis, if the known faults of VLSI are known. The use of managed MFIS  $A_y$  together with the control autom-

aton of the  $A_M$  strategy creates the prerequisites for the construction, production and use of partially serviceable devices when processing information.

As an example, consider the two-level MUSP, which includes the managed MFIS<sub>3</sub>  $A_y$  (Figure 4.5) and the  $A_M$  strategy automata for each group of MFIS<sub>1</sub> and MFIS<sub>2</sub> (Figure 5.3), and we estimate the performance of the MUSP as a memory device capable of storing the minimum number of defined states for partial Damage to the circuit.

Suppose that the elements in the event of failure have a constant value of logical zero. In this case, the MFIS  $A_y$  is still capable of functioning as an elementary automaton with memory in the event of the failure of an arbitrary element in any group, which is 25% of the equipment, and one more element in the other group. In this case, the MFIS  $A_y$  is able to function with 50% of faulty elements and 100% failure of the  $A_M$  strategy machine, which consists of two MUSP.

In case of failure of one element in the  $A_M$  strategy machine, the memory device reduces its area of operation to one subset of its states, i.e. by about 16.7%.

In case of catastrophic failure of one element in the  $A_M$  strategy machine for one group of elements of the MFIS  $A_y$ , when its output is always equal to the logical one and the  $A_M$  machine is permanently in only one state, the MFIS  $A_y$  narrows its field of operation.

In case of catastrophic failure of an arbitrary element, in an arbitrary group of the MFIS  $A_y$  a two-level memory device is transformed into a single-level memory device that can function independently according to the laws of the  $A_M$  strategy machine. Only with catastrophic outputs of the

elements in the MFIS  $A_y$  and  $A_M$ , the two-level memory device completely fails as a memory element.

Thus, three cases of the operation of a two-level memory device are possible:

- 1) 100% working capacity of all components and memory devices;
- 2) narrowing the areas of states of the memory device in which it can work with partial malfunctions of its components;
- 3) the failure of the memory device in the event of catastrophic component failure.

Consider the principles of building an MUSP with increased survivability.

The principles of building an MUSP with increased survivability are as follows:

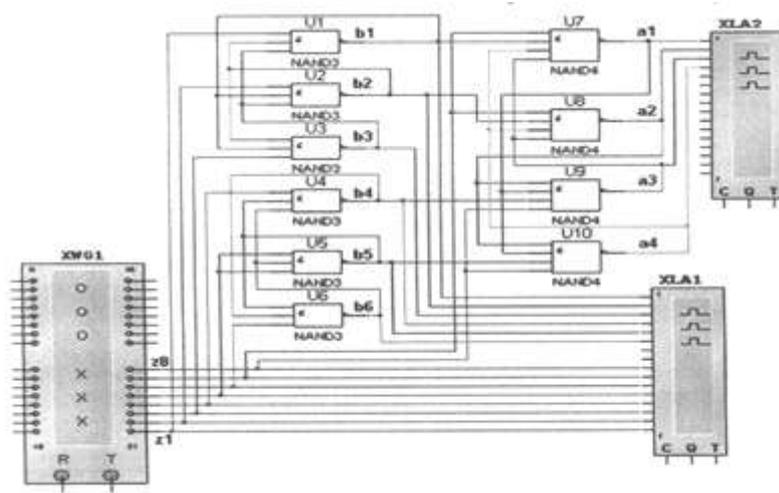
1. Specific malfunctions of circuits that are not catastrophic in the operation of the MUSP are determined.
2. Manufacturing technologies of memory devices are adjusted to eliminate catastrophic failures, in order to reduce non-catastrophic ones.
3. Determine the percentage yield of 100% fit and partially usable devices in relation to the percentage of completely unusable memory devices.

If this percentage of completely unusable devices and 100% of suitable devices satisfy manufacturers, then it becomes possible to use still partially suitable memory devices.

### **5.11.3. Monitoring the performance of the MUSP**

The control of the operability of the MUSP consists in checking the absence of catastrophic failures of the memory circuit when the input signal setting the  $x_p(t)$  is input, in which an active logical unit is supplied to all the input nodes of the memory circuit, which uniquely sets a logical zero on all output nodes of the memory circuit.

For all other  $x_i(t)$  input signals, at least one logical unit is uniquely installed on the output nodes of the memory scheme.



**Fig. 5.5.** Study of the MUSP class  $L_N^B$

The study of a multilevel class memory scheme using Multisim is carried out as follows: first we define a functional memory scheme and connect to it the Word Generator, Logic Analyzer (Figure 6.5).

For greater clarity and persuasiveness of the correct functioning of the two-level memory scheme of the class, we formulate tests of input words  $p(T)$  consisting of elementary sets of input signals that establish  $x_i(t)$  and one set of  $e(\Delta)$  preserving input signal that has a value on all input nodes  $z_i = 1$ .

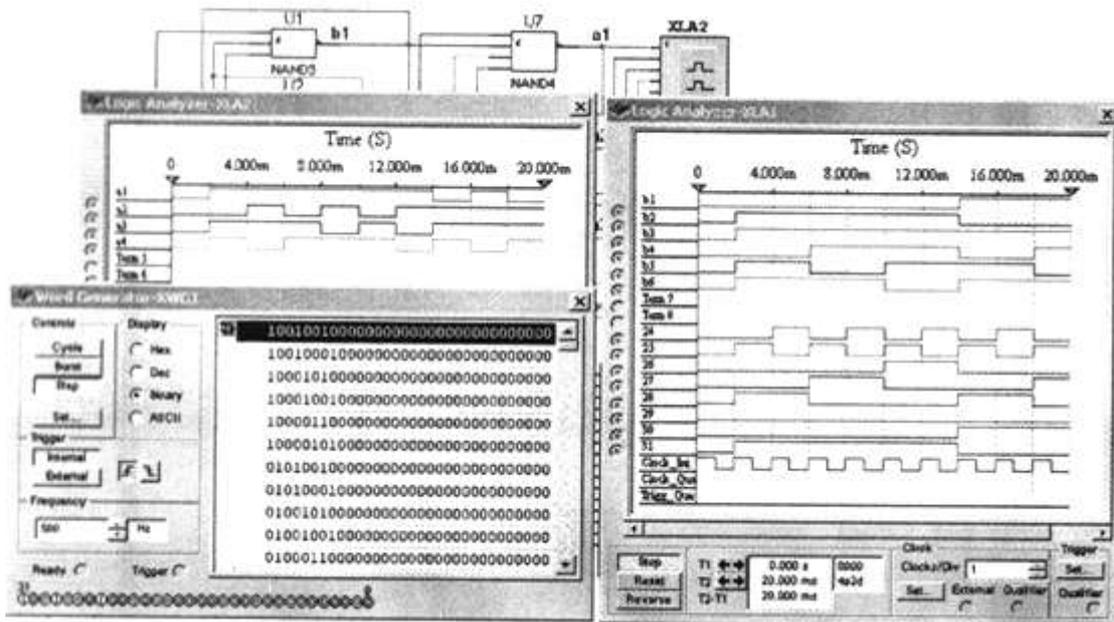
We construct tests of input words  $p = x, e$  to test the operation of the constructed functional scheme in Word Generator XWG1. The analysis of the operation of the memory circuit on the elements of the NAND with the help of simulation "NI Multisim 9" is carried out as follows [9]:

- Run the program "NI Multisim9" on the PC.
- Using the "Place Misc Digital" menu, we call up the necessary logical elements on the work field and build a functional diagram;
- We conduct the study of the scheme by virtual devices Multisim 9 - Word Generator and Logic Analyzer.

Table 5.8

Tests for testing the performance of an MUSP class  $L_N^B$ 

Installing Inputs Signals	Structural Input signals			Structural Output signals			States MUSP
	output numbers			–			
	Word generator						
31 30 29	28 27 26	25 24	$b_1 b_2 b_3$	$b_q b_2 b_3$	$a_1 a_2 a_3 a_4$	$A_i$	
$x_i$	$z_1 z_2 z_3$	$z_4 z_5 z_6$	$z_7 z_8$	$b_1 b_2 b_3$	$b_q b_2 b_3$	$a_1 a_2 a_3 a_4$	$A_i$
$x_1$	1 0 0	1 0 0	1 0	0 1 1	0 1 1	1 0 1 1	$A_1$
$x_2$	1 0 0	1 0 0	0 1	0 1 1	0 1 1	1 1 1 0	$A_2$
$x_3$	1 0 0	0 1 0	1 0	0 1 1	1 1 0	1 0 1 1	$A_3$
$x_4$	1 0 0	0 1 0	0 1	0 1 1	1 1 0	1 1 0 1	$A_4$
$x_5$	1 0 0	0 0 1	1 0	0 1 1	1 1 0	1 0 1 1	$A_5$
$x_6$	1 0 0	0 0 1	0 1	0 1 1	1 1 0	1 1 0 0	$A_6$
$x_7$	0 1 0	1 0 0	1 0	1 0 1	0 1 1	0 1 1 1	$A_7$
$x_8$	0 1 0	1 0 0	0 1	1 0 1	0 1 1	1 1 1 0	$A_8$
$x_9$	0 1 0	0 1 0	1 0	1 0 1	1 1 0	0 1 1 1	$A_9$
$x_{10}$	0 1 0	0 1 0	0 1	1 0 1	1 1 0	1 1 0 1	$A_{10}$
$x_{11}$	0 1 0	0 0 1	1 0	1 0 1	1 1 0	0 1 1 1	$A_{11}$
$x_{12}$	0 1 0	0 0 1	0 1	1 0 1	1 1 0	1 1 0 0	$A_{12}$
$x_{13}$	0 0 1	1 0 0	1 0	1 1 0	0 1 1	0 0 1 1	$A_{13}$
$x_{14}$	0 0 1	1 0 0	0 1	1 1 0	0 1 1	1 1 1 0	$A_{14}$
$x_{15}$	0 0 1	0 1 0	1 0	1 1 0	1 1 0	0 0 1 1	$A_{15}$
$x_{16}l$	0 0 1	0 1 0	0 1	1 1 0	1 1 0	1 1 0 1	$A_{16}$
$x_{17}$	0 0 1	0 0 1	1 0	1 1 0	1 1 0	0 0 1 1	$A_{17}$
$x_{18}$	0 0 1	0 0 1	0 1	1 1 0	1 1 0	1 1 0 0	$A_{18}$



**Fig. 5.6.** Analysis of the operation of the memory circuit

Fig. 5.6 shows a view with a word generator and a logic analyzer after performing a step-by-step sequence of sets of  $x(t)$  input signals and a set of  $e(\Delta)$  saving input signal after 18 tests. Studies have shown the correctness of using elementary  $p(T)$  input words (Table 5.8), which reflected their respective functioning in a deterministic mode.

Thus, the considered methodology for determining the deterministic input words of elementary multi-level memory schemes and testing the operation of these memory circuits using simulation Electronics Workbench (MultiSim 9) [9; 14] convincingly proved their efficiency.

#### **5.11.4. Improving the reliability of devices using the MFIS in the MUSP**

The basic memory circuits of the MFIS have the properties of storing different blocks of  $\pi_j$  memorized states. Combination circuits<sup>6</sup> that realize the excitation functions and output functions of certain  $\pi_j$  states, and the automaton of strategies generat-

ing  $e_j(\Delta)$  -containing input signals to the MUSP are inserted as separate boards into the connectors of the computing device with a diagnostic fault detection system, Possibility of replacing faulty cards (plug-in modules) with serviceable ones.

This is because the input signal corresponds to a logic 1 value when the input wires are hanging, which does not affect the operation of the NAND gate used by the MFIS and MUSP, but narrows the scope of their functioning.

Thus, it is possible to construct a computing device containing one main block of  $\pi_1$  memorized states,  $(r_e-1)$  reserve blocks  $\pi_j$  ( $j = r_e - 1$ ) states and re combinational circuits realizing excitation functions and outputs, one of which is basic and  $(r_e - 1)$  additional.

In terms of reliability, we investigate the operation of the device with a loaded reserve of  $r_e$  blocks, unrestricted and "fast" recovery of the failed blocks, that is,

$$\max_i T_0^{(i)} / \min_i T_1^{(i)} \ll 1, \quad (5.24)$$

where  $T_1^{(i)}$  – is the mean time between failures of the i-th unit of the device;

$T_0^{(i)}$  - average recovery time of the i-th unit of the device.

In this case, the mean time between failures of the system can be determined by the exact formula

$$T_1 = \frac{\prod_{i=1}^{r_e} (T_1^{(j)} + T_0^{(j)}) - \prod_{j=1}^{r_e} T_0^{(j)}}{\prod_{i=1}^{r_e} T_0^{(i)} \left( \sum_{j=1}^{r_e} \frac{1}{T_0^{(j)}} \right)}. \quad (5.25)$$

The approximate formula for "fast" recovery is:

$$T_1 = \left[ \left( \prod_{j=1}^{r_e} \frac{T_0^{(j)}}{T_1^{(j)}} \right) \cdot \left( \sum_{i=1}^{r_e} \frac{1}{T_0^{(i)}} \right) \right]^{-1} \quad (5.26)$$

The average recovery time of the system is determined by the following formula:

$$T_0 = \left[ \sum_{j=1}^{r_e} \frac{1}{T_0^{(j)}} \right]^{-1}. \quad (5.27)$$

The probability of failure-free operation of the system with "rapid" recovery can be approximated by the exponential law:

$$P(t) = e^{-\frac{t}{T_1}}. \quad (5.28)$$

Assuming that the  $i$ -th block consists of successively connected elements having the same magnitude of failure rate ( $\lambda_o = 1 \cdot 10^{-7} \text{ 1/ч}$ ), the value of the mean time between failures of the  $i$ -th block will be equal to

$$T_1^{(i)} = \frac{1}{100 \lambda_j} = 1 \cdot 10^5 \text{ ч}. \quad (5.29)$$

Having determined the average recovery time of the  $i$ -th block of the system equal to one hour ( $T_0^{(i)} = 1 \text{ hour}$ ), it is possible to determine  $T_0$  and  $T_1$  of the system according to the corresponding formulas (5.16-5.18).

We define  $T_0$  and  $T_1$  of the system for  $r_e = 2$ .

$$T_1 = \frac{(T_1^{(i)} + T_0^{(i)})^2 - (T_0^{(j)})^2}{(T_0^{(i)})^2 \cdot \frac{2}{T_0}} = \frac{(10^5 + 1)^2}{1^2 \cdot \frac{2}{1^4}} = 0,5 \cdot 10^{10} \text{ h}$$

$$T_0 = \left[ \frac{2}{T_0^{(j)}} \right]^{-1} = \frac{T_0^{(j)}}{2} = 0,5 \text{ h}$$

These calculations show that the average recovery time falls, and the average time between failures of the system increases when the fault recovers, which indicates an increase in system reliability.

In the literature on systems for processing and calculating the reliability of radio electronics and automation equipment [15], recommendations are given for building reliable systems by building nodes, devices and computing systems with a changing architecture in the event of failures of individual products.

From the analysis of the MFIS and MUSP from the point of view of reliability, it can be concluded with certainty that they significantly increase the reliability of multistable memory schemes by introducing redundancy of the elements in the MFIS groups. On the other hand, it is possible to design multi-functional nodes, devices and computer systems with a changing architecture on their basis [3].

The proposed MFIS and MUSP expand the capabilities of the element base of computer systems due to the possibility of their functioning in different subsets of  $\pi$  states. They are also able to shorten the time to rebuild the computer from one algorithm to another, reduce the hardware costs in memory circuits to a single memorized state, and significantly speed up the solution to the problem of creating workable devices with partial malfunctions of their components.

## **5.12. Conclusion to chapter 5**

Multilevel memory schemes, which are considered, are semi-closed structures. They, as well as multifunctional memory circuits, have a big advantage in comparison with the triggers for the hardware costs for a single memorized state, for the functional possibilities of processing general and private information for one machine clock  $T$ , and also for increased reliability and survivability.

In the author's view, MUSP can be used to build reconfigurable and more reliable computer devices and systems than computer devices with memory on the triggers.

## **CONCLUSION TO SECTION 2**

In Section 2 of Chapter 4, the basic concepts of multifunctional memory circuits, the method of microstructural synthesis of elementary multifunctional memory circuits, the symbolic language for describing elementary multifunctional memory circuits, possible versions of multifunctional memory circuits in the symbolic number, the synthesis of multifunctional memory circuits by symbolic description, the definition of single-valued input and Enlarged elementary words of automatic memory circuits. At the end of the chapter, the issues of improving reliability and survivability are considered. As well as monitoring the efficiency of memory circuits. The characteristics of the parameters of multifunctional memory circuits and their comparison with SMP and triggers are given.

Chapter 5 of the basic concept of multilevel memory circuits, character description language multilevel memory circuits, determining memory multilevel circuit parameters a character description memory multilevel circuit synthesis method for symbolic description, the principle of the structural organization of the elementary multilevel memory circuits, a general strategy machine design method for the entire multilevel memory circuits, logic design methods with one machine gun strategy and a strategy for each group multilevel x memory circuits, classification basic elementary memory circuits.

The comparison of the parameters of the basic memory circuits with monofunctional memory circuits by the basic parameters is given: the number of logical elements necessary for constructing memory circuits, the maximum possible load, the ability to output memory circuits, the number of internal and external links of memory circuits, the number of elements per state Memory schemes, Comparison of the operating switching frequency and memory scheme functionality, Issues of con-

structing reliable devices in multi-level memory circuits and. Increasing the reliability of devices using MFIS in the MUSP

## **SECTION 3**

### **TYPICAL COMPUTER DEVICES FOR AUTOMATIC MEMORY SCHEMES.**

#### **Chapter 6**

#### **METHODS FOR BUILDING RECONFIGURABLE REGISTERS**

##### **6.1. Basic concepts**

The creation of high-performance computing systems is included in the top ten vital programs of the world's leading countries.

The best solution to this problem is to force the development of original developments in the field of high-performance computing systems. Very promising is the concept of building reconfigured multiprocessor computers.

The essence of this concept is that the architecture of the computer system must be able to adapt to the structure of the problem being solved. The development of principles and methods for designing reconfigured systems of different firms with elementary memory on flip-flops is historically justified by the fact that when developing the technology of designing large-scale integrated circuits, the functions of the element base during their creation did not change, and the systems based on them needed to be adapted to solve problems.

For digital automata, the element base was determined even during the creation of the first computers in the 40s of the 20th century. Theoretically, the composition of the element base was justified by Academician V.M. Glushkov in 1962 in the theorem on the structural completeness of elementary automata, which stated that for the construction of any arbitrary automaton an elementary Moore automaton with non-trivial memory (i.e. RS-flip-flop) and a functionally complete system of logical elements is sufficient [1].

The creation of large-scale integrated circuits (LSI) divided developers into their new occupations, which required great knowledge. Developers of computers on

LSIs, whose composition currently exceeds a billion components, were divided into whole scientific areas:

- developers of computer systems based on LSI [2-3];
- programmers and system programmers using the commands of already created processors within the BIS [4];
- Developers of LSI on increasingly thin nanotechnologies [5-7].

This historical trend of the division of labor of developers into "narrow" specialists made it difficult to take an integrated approach to the creation of effective computing systems and hindered the development of the elemental base of BIS to date. This was further facilitated by the fact that American firms became monopolists in creating new promising developments of modern computers on BIS. At the heart of them they raised the question of obtaining superprofits. While Moore's law was working, which said that every year there will be an increase in components in the BIS twice, they did not have special problems. But by 2014 this law ceased to function, as was shown in [8].

To adapt the systems to the solution of problems (instead of the concept of creating a new element base for LSI), a multifunctional array of LSIs was used. This array was rebuilt and adapted with the control device of another LSI. Such a two-level system allowed on the "automatic" level to carry out sequential processing of the general information that controlled the reorganization of algorithms and the processing of a separate algorithm of a multifunctional device or computer system.

In fact, this means that the user should be given the opportunity to program problem-oriented multi-processor computing systems, the structure of which is adequate to the task assigned to them. At the same time, unlike the existing architecture of John von Neumann, high real performance of computing systems is achieved on a wide class of problems, as well as an almost linear increase in performance with an increase in the number of processors.

The development of methods for constructing elementary multifunctional memory circuits [22] and methods for constructing elementary multi-level memory circuits [20-21] allowed them to consider the possibility of building reconfigured de-

vices of computer systems using the "elemental" level of memory schemes [18-19; 23-24]. The use of MFIS and MUSP in the construction of reconfigured devices of computer systems made it possible to make a comprehensive approach that partially simplified the method of constructing reconfigured devices without requiring additional special devices to rearrange the functioning algorithms and, thereby, increased the processing speed of hierarchical information during the reorganization of its processing.

Increasing the speed of processing of hierarchical information is explained by the fact that the reconfiguration of reconfigured computer systems built at the "automatic" level requires an additional machine clock, which is used to reconstruct the general information in the strategy automaton [9-17]. If you use the same as the elementary memory of an MUSP, this can be avoided [18-38].

## **6.2. Methods for constructing reconfigured registers on multi-level memory circuits**

A register is a node providing reception, storage and delivery of information, as well as performing a number of logical operations on information that is stored in the register. The main functional purpose distinguishes between memory registers and shift registers. In addition to these basic functions in registers, you can perform the operations of converting direct code to inverse code and vice versa, and also bitwise operations of conjunction and addition by mod 2. In the literature, memory registers occur under the name of static registers on triggers [39].

The main purpose: parallel reception of multi-digit codes (words) and their storage for a long time. In modern computers (for example, Pentium) 32-bit registers are widely used. Registers are found for general purpose (RON), floating-point registers (FPDs), super-operative memory (SOS), result and data registers, input registers, buffer registers, information registers, number registers, byte and tetad transfer regis-

ters of arithmetic logic units (ALU) processors, main memory address (OOP) registers, etc. [39].

The information signals  $x$  for the MUSP can be clocked by the signal  $\tau$ . In the absence of a signal  $\tau$  at the input nodes of the MUSP, only one  $e(\Delta)$  input signal is fed to the input nodes.

The information signals  $x$  for the MUSP can be clocked by the signal  $\tau$ . In the absence of a signal  $\tau$  at the input nodes of the MUSP, only one  $e(\Delta)$  input signal is fed to the input nodes.

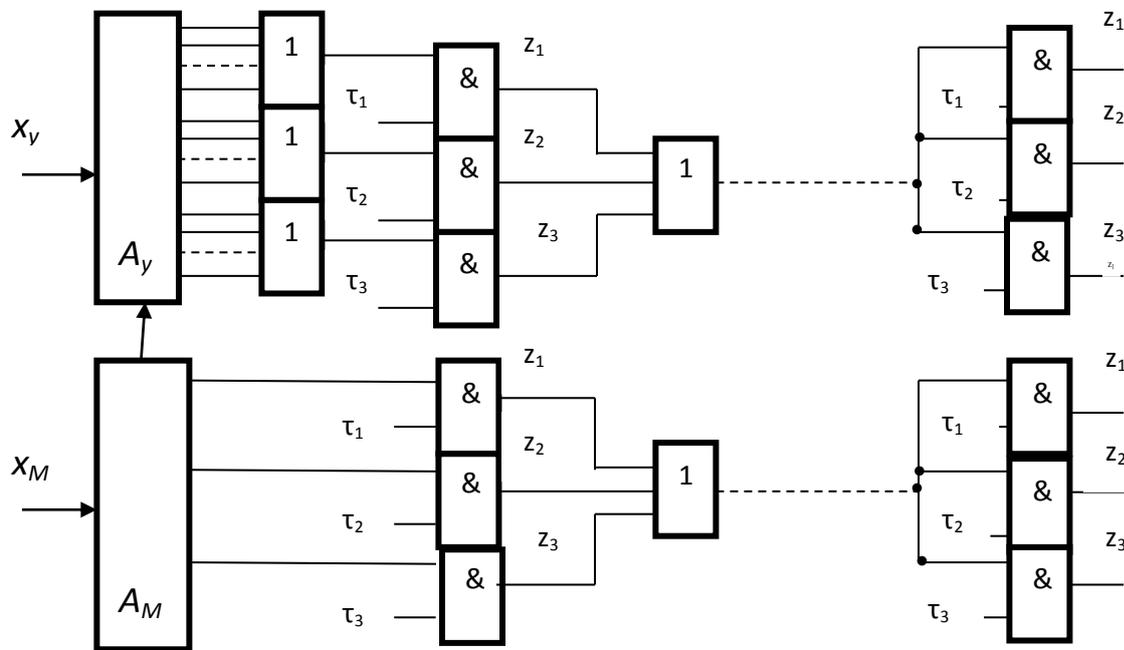
The general principles of constructing registers for receiving and transferring words of information without first setting the MUSP to a known initial state have a record of the desired word at each level of the MUSP memory. The input signals of the input word MUSP exist in two types: the input signals  $x(z_y)$  of the controlled MFIS  $A_y$  and the input signals  $x(z_M)$  of the MFIS of the  $A_M$  strategy machine. In this case, the input word MUSP can have a direct or inverse value of each level of the  $i$ th bit of the register. The input nodes of the MFIS can be fed together with the information signal  $x$  and the clock signal  $\tau$ .

Thus, the MUSP can function as an automaton of the second kind, having a transition in the cycle  $t$  from one state to another, giving an output signal  $y(T)$ , or as an automaton of the third kind, having a transition in the cycle  $\Delta$  from one state to another and producing an output signal  $y(\Delta)$  [23].

When performing transitions to an MUSP (as an elementary automaton of the second kind), two modes can be used: the simultaneous transition of all MFIS of a multilevel memory under the influence of sets of  $x_i(t)$  input signals from one state to another or a transition to the MFIS  $A_y$  (with the machine state unchanged  $A_M$  strategy) only from one state to another in a certain block of  $\pi_j$  states under the influence of sets of  $x_y(t)$  input signals.

When the MUSP functions (as an elementary automaton of the third kind) during the internal cycle  $\Delta$ , one can use enlarged transitions in a certain block of  $\mu_i$  states under the influence of only sets of  $x_M(t)$  input signals of the  $A_M$  strategy automaton.

The output signals of the MUSP (or other memory circuits) can be received by other memory circuits only when the next clock signal  $\tau_{i+1}$  appears. For this, the output signals of the MUSP must have stable values after the machine clock  $T$ , which reflects the period between two clock signals  $\tau_i$  and  $\tau_{i+1}$ , to reliably remove the information. The output signals  $y(T)$  of automata of the second kind as well as the output signals  $y(\Delta)$  of automata of the third kind can be used when the next sync pulse  $\tau_{i+1}$  appears. Receiving information from the source nodes of memory circuits can be performed in the usual ways: asynchronously or synchronously. The removal of information from the output nodes of memory circuits can be performed in parallel from all nodes or sequentially, as shown in the diagram in Fig. 6.1.

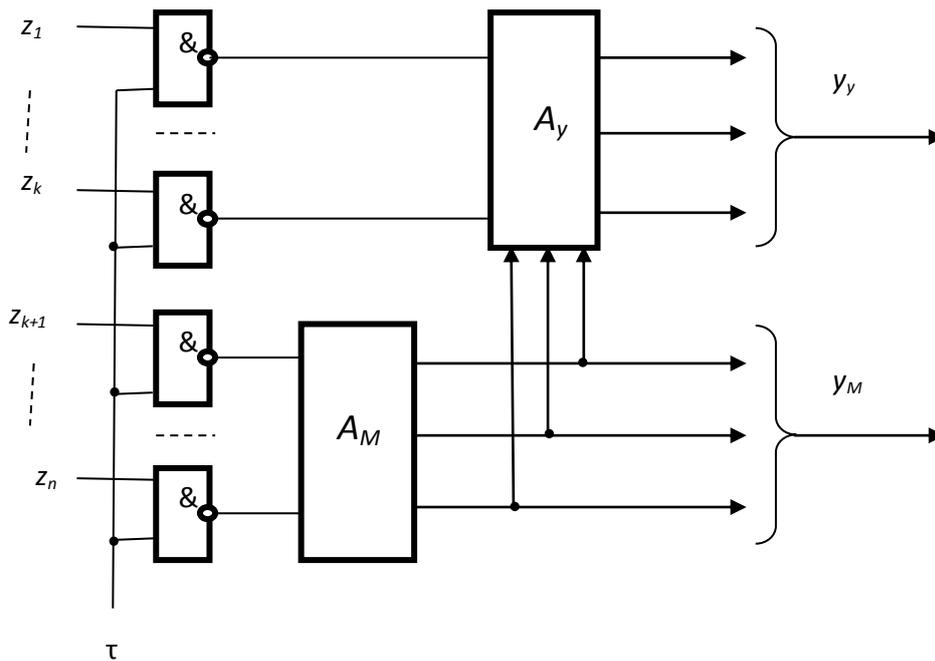


**Fig. 6.1.** Consistent organization of information transfer

With such consecutive data removal from the original nodes of the memory scheme, the number of links is reduced to the number of memory levels. The clock signals  $\tau_i$  ( $i = 1, 2, \dots, K$ ) have a signal duration that is sufficient to record information in other devices of computer system components. Consistent organization of information retrieval is used when it is necessary to reduce communications between devices or their units. In this case, the speed of information transfer from one block to another is reduced.

All incoming nodes are synchronized with the  $\tau$  signal. The output signals are taken in parallel from all the original nodes of the MUSP. The construction of functionally reliable devices is performed with arbitrary coding of the state of the machine due to the use of clock signals (Figure 6.2) and two-level registers (for example, registers on triggers) [39].

Let us give an example of a single-stage MUSP (Figure 6.2). Two-stage registers allow recording of information in the first-stage MUSP under the influence of sets of  $x_i(t)$  input signals, which include stable arguments of sets  $y_j(\Delta)$  of output signals. The second stage MUSP uses the sync signals  $\tau_2$  (Figure 6.3).



**Fig. 6.2.** Single-stage synchronous MUSP

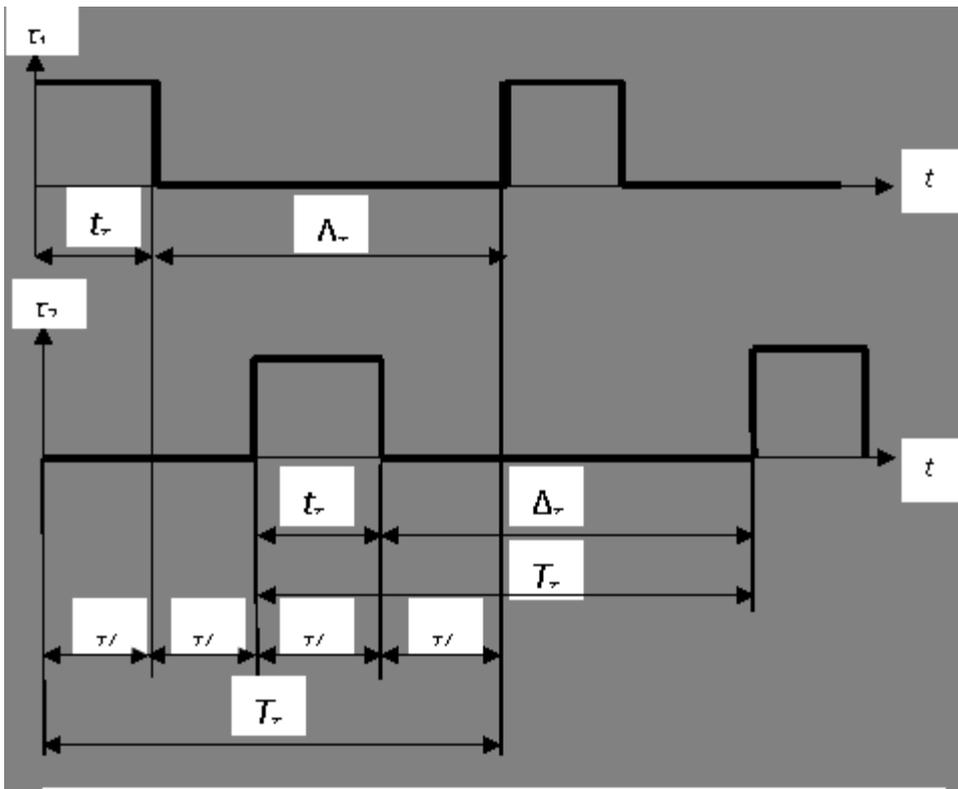


Fig. 6.3. Synchronous signals  $\tau_1$  and  $\tau_2$

The sets of  $x_k(t)$  input signals applied to the buses  $z_i$  ( $i = 1, \dots, n$ ) of single-stage

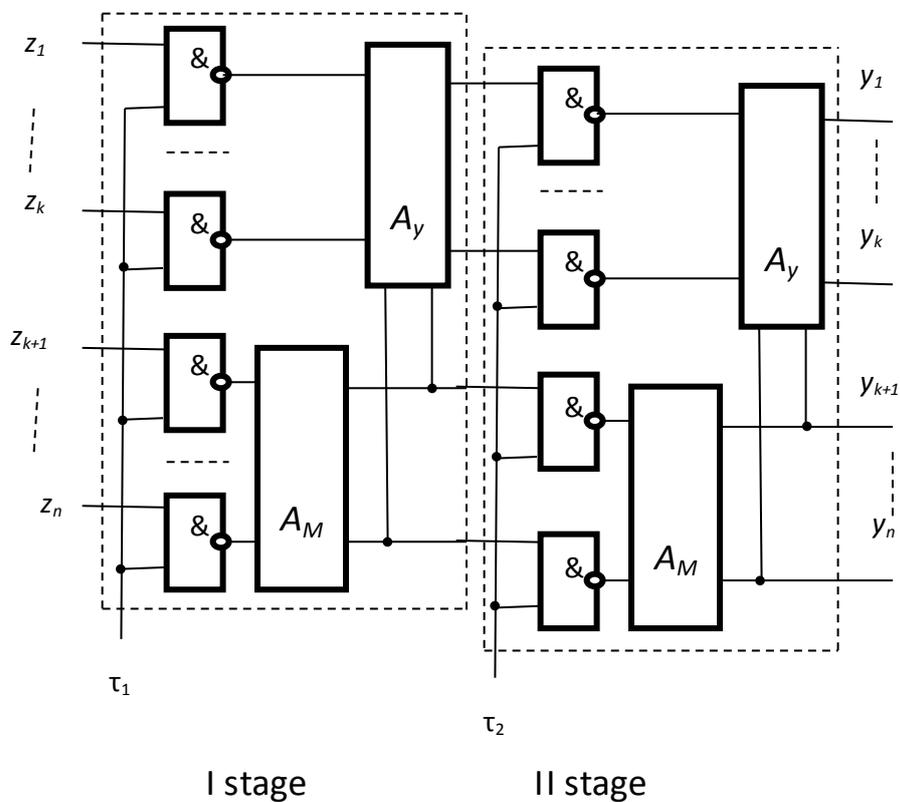


Fig. 6.4. Two-stage MUSP

synchronous MUSP (Figure 6.2) have a logical zero value for all MUSP groups for all  $BA_i$  groups except one. The value of the sets  $y_j(\Delta)$  of the output signals of the first-stage MUSP are equal to the values of the sets of the  $x_k(t)$  input signals of the second-power MUSP. On the basis of this property, single-stage MUSP are connected to each other and find application as the  $i$ -th bit of the parallel register (Figure 6.4).

Each stage of the MUSP (Figure 6.4) can work by executing the transition in cycles  $t_{vj}$  and  $\Delta_{vj}$  ( $j = 1, 2$ ). Transitions to the first stage MUSP end before the second stage clock signal  $\tau_2$  appears, and the transitions in the second stage end before the first stage clock signal  $\tau_1$  appears, which is important for reliable operation of the two-stage synchronous MUSP.

The synchronous MUSP is characterized by the fact that each of its transitions occurs when a clock signal (synchronous) signal  $\tau$  is applied to the input or when a synchronous signal  $\tau$  ends with a minimum delay of one logic element, allowing on the output nodes of the MUSP to the appearance of the clock pulse  $\tau_{i+1}$  to have stable sets  $y_j(\Delta)$  of output signals.

Single-stage asynchronous and synchronous MUSP and two-stage MUSP can be implemented in the construction of various registers in the components of computer systems.

Repeating the structure of the  $i$ th digit of the two-stage register on the MUSP  $n$  times, it is possible to compile a general scheme of the  $n$ -bit parallel register (Figure 6.5).

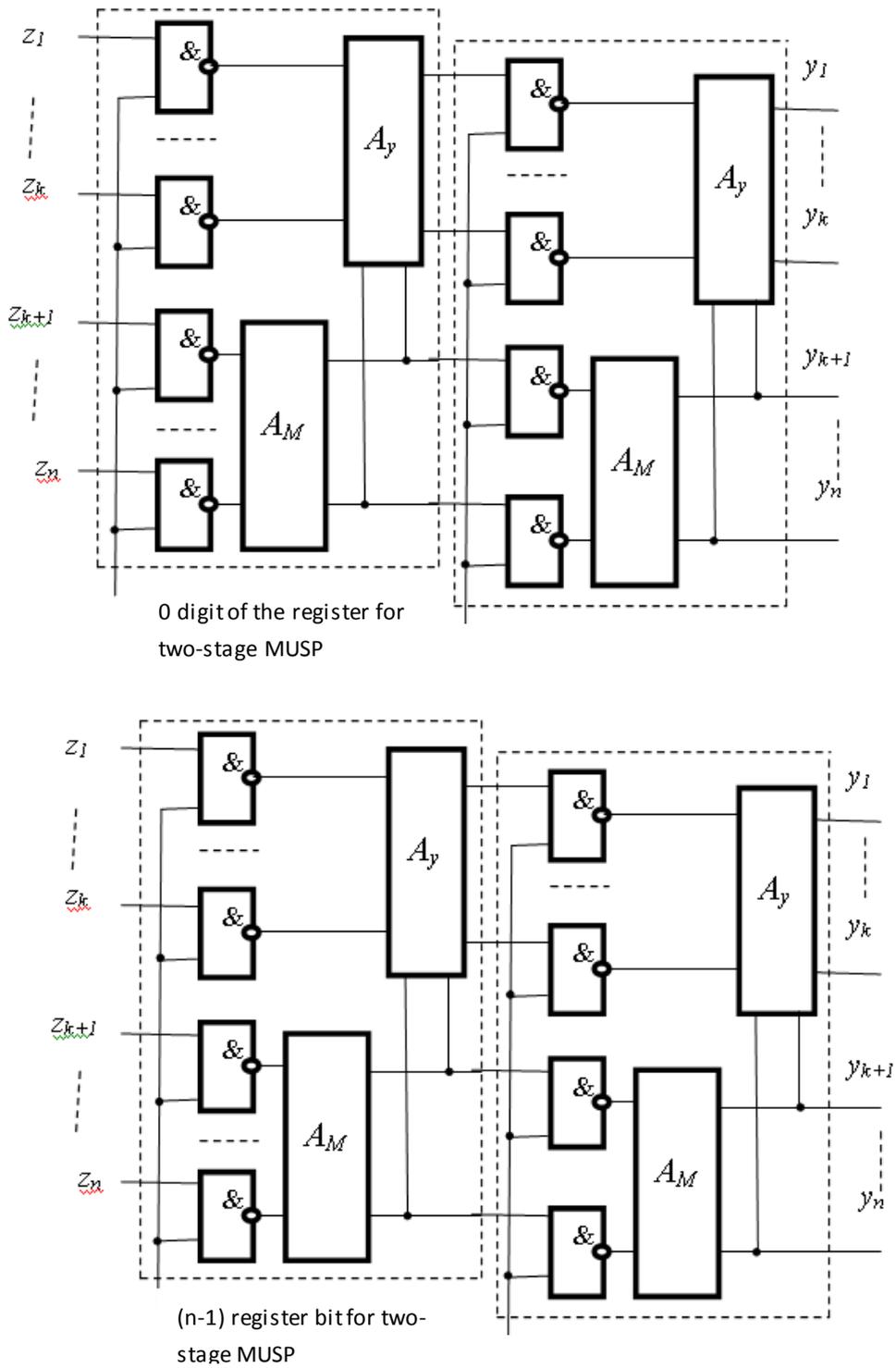
The number of memory states  $M_p$  of this register can be determined by the formula:

$$M_p = M_N^n, \quad (6.1)$$

where  $n$  is the number of register bits;

$M_N$  is the number of memory states of the MUSP.

The number of  $M_N$  memory states of the MUSP is determined by the formula:



**Fig. 6.5.** Parallel register on n-bits

$$M_N = \prod_{j=1}^N m_j, \quad (6.2)$$

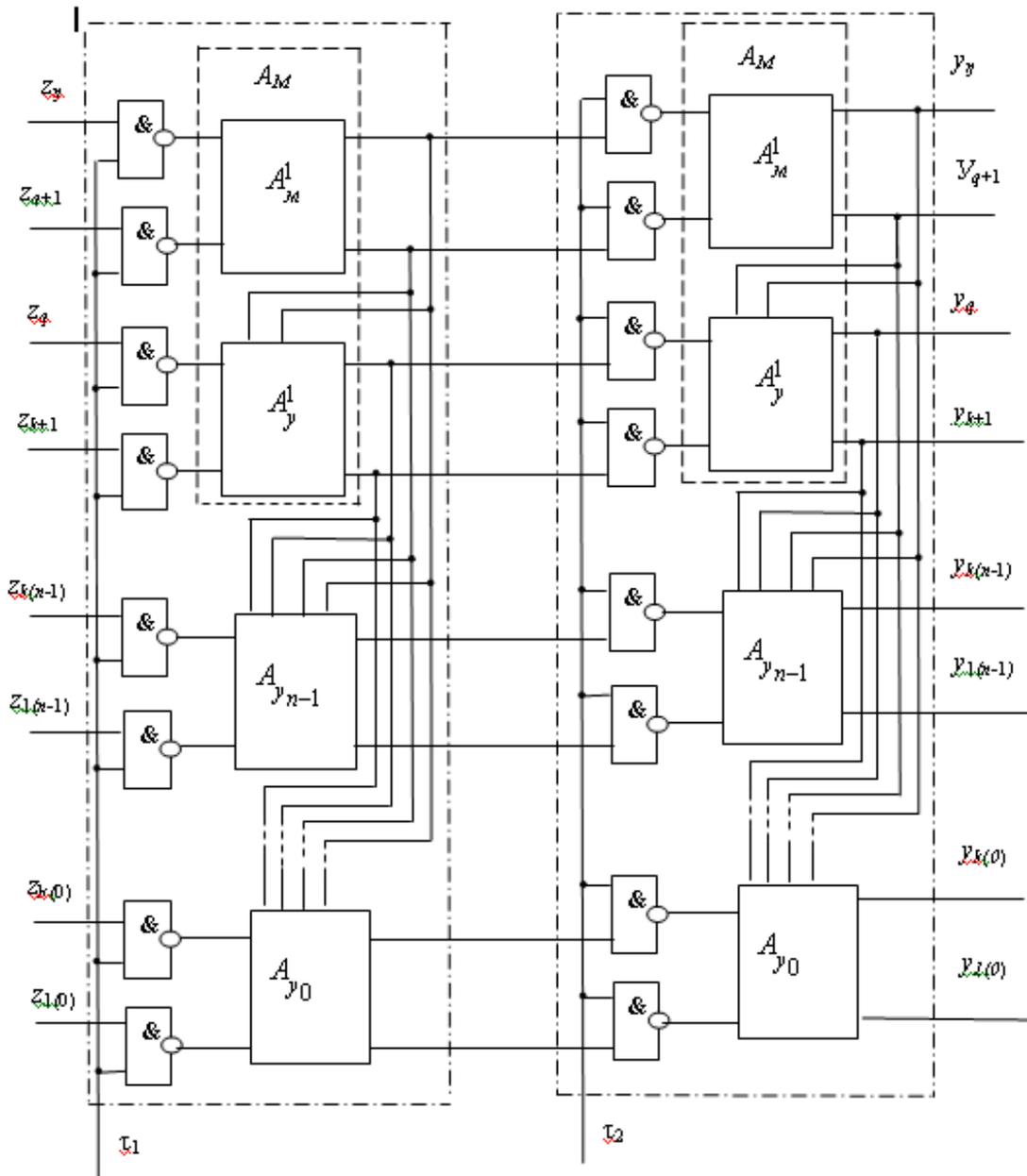
where  $N$  is the number of levels in the MUSP;

$m_j$  is the number of BA groups in the  $j$ -th level MUSP.

The range of integers that are identified with the number of register states is as follows:

$$0 \leq \lfloor A \rfloor < M_p. \quad (6.3)$$

In a multilevel parallel register, you can use the  $A_M$  strategy automaton not in each MUSP  $i$ -th bit, but one for all MFIS of the entire register (Figure 6.6).



**Fig. 6.6.** One registry strategy machine for all MFISs

The range of integers that are identified with the number of register states with one  $A_M$  strategy automaton looks like this:

$$0 \leq [A] < M_M \times M_y^n, \quad (6.4)$$

where  $M_m$  is the number of storage states of the  $A_M$  strategy machine;

$M_y$  is the number of memory states of the MFIS  $A_y$ ;

$n$  is the number of digits in the register.

The ranges of register numbers are divided into numerical segments determined by the states of the  $A_M$  strategy automaton. Parallel registers on the MUSP can function in different memory circuits of the strategy automaton, as well as in the matrix state blocks of the MFIS  $\pi_j$  and  $\mu_i$  [23]. If we represent the block of  $\pi_j$  states in the form of a numerical segment of integers, each number of which is identified with the state of all MFIS in the MUSP register, then the enlarged transitions in blocks of  $\mu_i$  states allow the register to go from one numerical segment to another.

Enlarged deterministic transitions are performed in the inner cycle  $\Delta$  of the machine clock and characterize the register on the MUSP as an automaton of the third kind [23].

Output signals of a single-stage parallel register (the structure of the  $i$ th bit of the parallel register on the MUSP, which is depicted in Figure 6.3) is determined for each  $i$ -th bit simultaneously with the outputs of the  $A_M$  strategy machine and with the output signals of the MFIS  $A_y$ . A characteristic feature of the set of  $y_i$  output signals of a single-stage register on the MUSP under deterministic work is their equality to a set of input signals that establish  $x_i$ .

This feature allows for the relatively simple organization of two-stage registers.

Two-stage parallel registers can be implemented on two-step MUSP, as shown in Fig. 6.4. The number of  $i$ -th bits of a two-stage register on the MUSP determines the bit capacity of the register itself. The input information is fed to the input nodes of the first stage of the register and, when the clock signal  $\tau_1$  appears, is written to the first stage of the register.

After carrying out transients in the first stage of the register and turning off the clock signal  $\tau_1$ , the information from the first power can be written to the second stage of the register. The recording of information into the second stage of the register can be performed when the sync signal  $\tau_2$  appears (Figure 6.6). This two-stage organization of the parallel register in conjunction with the clock signals allows free

encoding in the implementation of finite automata of digital devices of computer systems.

Qualitatively new properties of MUSP allow two-stage synchronous memory devices to rebuild the algorithm of their functioning without losing performance, while simultaneously memorizing general information in the  $A_M$  strategy machine and local information in the MFIS  $A_y$ .

### **6.3. Analysis of parameters of reconfigured parallel registers in multi-level memory circuits**

The limiting operating frequency of the  $F_p$  switching of asynchronous single-stage parallel registers to the MUSP corresponds to the limiting operating frequency of the switching of the basic memory circuits (flip-flops, SMEs, MFISs and MUSPs). Unlike asynchronous memory circuits, synchronous registers at the inputs of each i-group of the MFIS in the MUSP have additional AND (OR) circuits, the first input nodes of which are combined with the input of the synchronous pulse  $\tau$ , and other input nodes are used to receive the information sets  $x(t)$  of the input signals. The information that is fed to the information inputs can be received by the base MUSP only when the synchronization pulse  $\tau$  appears.

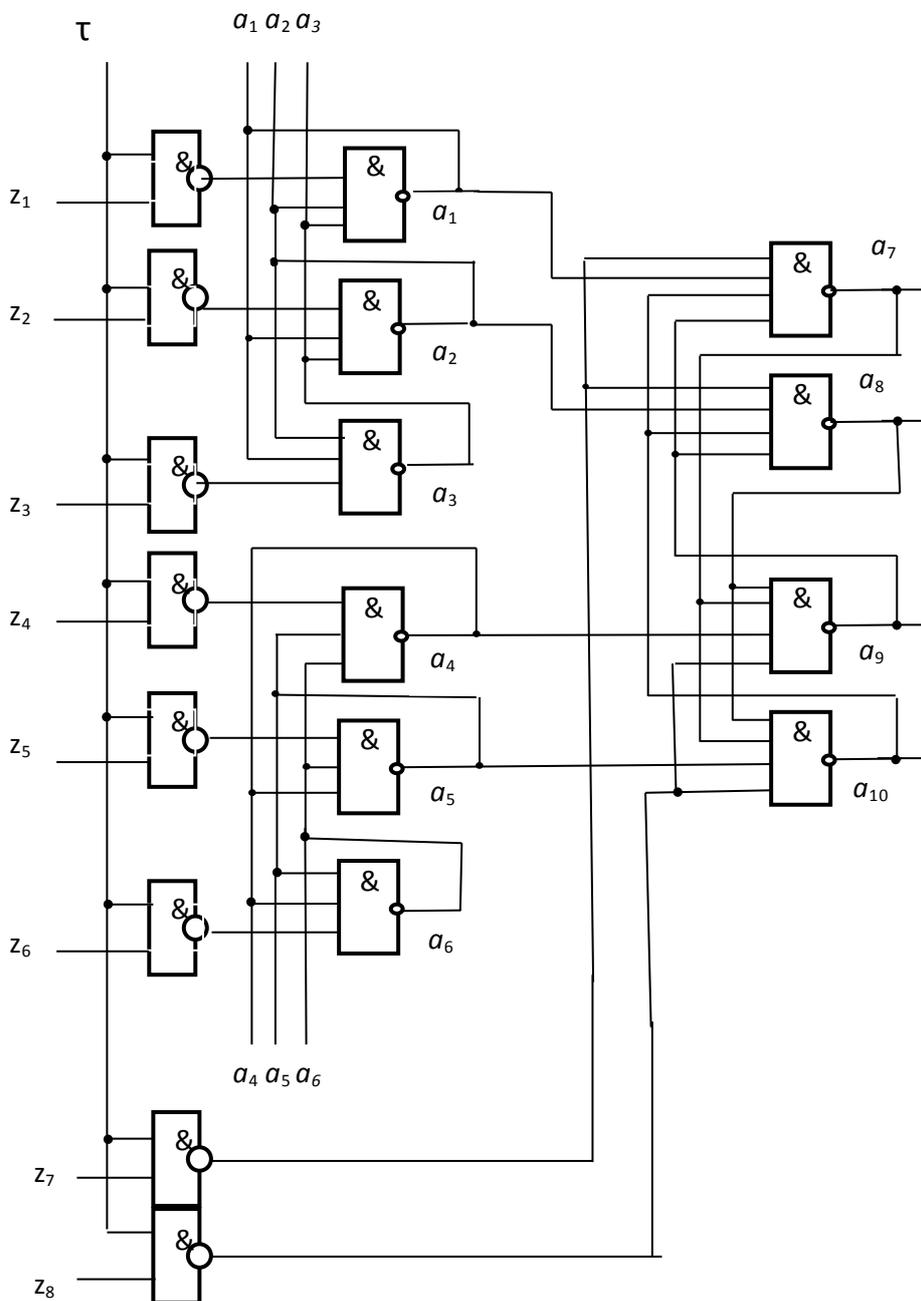
To build a synchronous MUSP on an MFIS of class  $L$ , we use a memory device with parameters corresponding to the speed of synchronous triggers. In connection with the fact that MFIS class  $L$  on the speed of work corresponds to the speed of the triggers, the maximum operating frequency of the register, which is built on the MUSP, is equal to:

$$F_p \leq 1/(4\tau_e), \quad (6.5)$$

where  $t_e$  is the propagation delay of the signal to one logical element.

A register built on an SMP can store an analogous number of  $M$  states, as well as a register that is built on an MUSP class  $L_N^B$ . However, the register for SMEs for hardware costs is less economical and has less functionality. Let's compare the data of parallel registers that store the same number of  $M$  states.

The number of NAND elements used in the  $i$ th bit of the parallel register on the MUSP is 18, and has 8 input nodes  $z$ , to which sets of setting  $x(t)$  input signals are fed, and one clock pulse  $\tau$ , the number of elements in the synchronous MUSP at one state equals 1 ( $L = 1$ ) (Figure 6.7).



**Fig. 6.7.** The register rank on the MUSP class  $L_N^B$

The number of NAND elements used in the  $i$ th bit of the parallel register on the triggers is 4 [39]. The number of elements in the synchronous trigger per one state is 2 ( $L = 2$ ). This is twice as much as in the synchronous MUSP (Figure 6.7). The number of internal links in a synchronous trigger per one state equals 2 ( $S_{\text{int.s}} = 2$ ),

which is more than in a synchronous MUSP, in which the number of internal connections is  $34/18 < 1.9$  ( $S_{\text{int.s}} < 1.9$ ). The number of external links in a synchronous trigger per one state is 2.5 ( $S_{\text{vnt.r.s}} = 2.5$ ), which is more than in a synchronous MUSP, in which the number of external links is  $19/18 < 1.06$  ( $S_{\text{vnt.r.s}} < 1.06$ ).

Thus, the synchronous register on the flip-flops uses more hardware costs than the synchronous register on the MUSP for one memorized state.

The register on the MUSP can still change the functioning of the MFIS included in its composition, realizing the various reflections  $\{X\}$  in  $\{Y\}$  in the corresponding blocks  $\pi_j$  of the memory states and carrying out the enlarged transitions in blocks  $\mu_j$  of the memory states the MFIS storage units, which is essentially unavailable to the registers on the flip-flops. Let's compare the analysis of registers, which demonstrates the advantages of parallel synchronous registers for MUSP for hardware costs, the number of internal and external links per one state and the advantage in terms of functionality, in Table. 6.1.

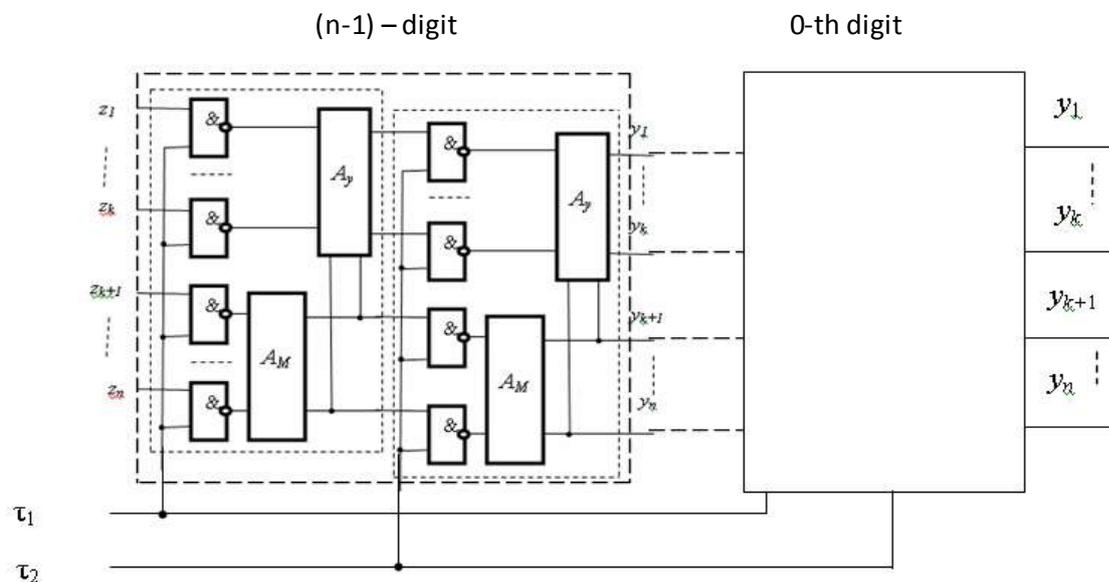
Table 6.1

Comparison of registers

Options	Register on Triggers	Register on MUSP
Number of logic elements per state	$L=2$	$L=1$
Number of internal links per state	$S_{\text{int.s}} = 2$	$S_{\text{int.s}} < 1,9$
Number of external links per state	$S_{\text{vnt.r.s}} = 2,5$	$S_{\text{vnt.r.s}} < 1,06$
Functions for implementing different mappings	NO	How in MFIS
Implementation functions of large-scale transitions	NO	How in MFIS
Border Operating Frequency	the same	the same

## 6.4. Methods for constructing the reconfigured shift registers on multi-level memory circuits

Shift registers are typical computer nodes. The registers are allocated with a shift to the right, left or reverse shifts of the shift register on a two-stage memory scheme [39]. The scheme of the shift register for two-step MUSP with a shift to the right is shown in Fig. 6.8.



**Fig. 6.8.** Reconfigured shift registers on the MUSP

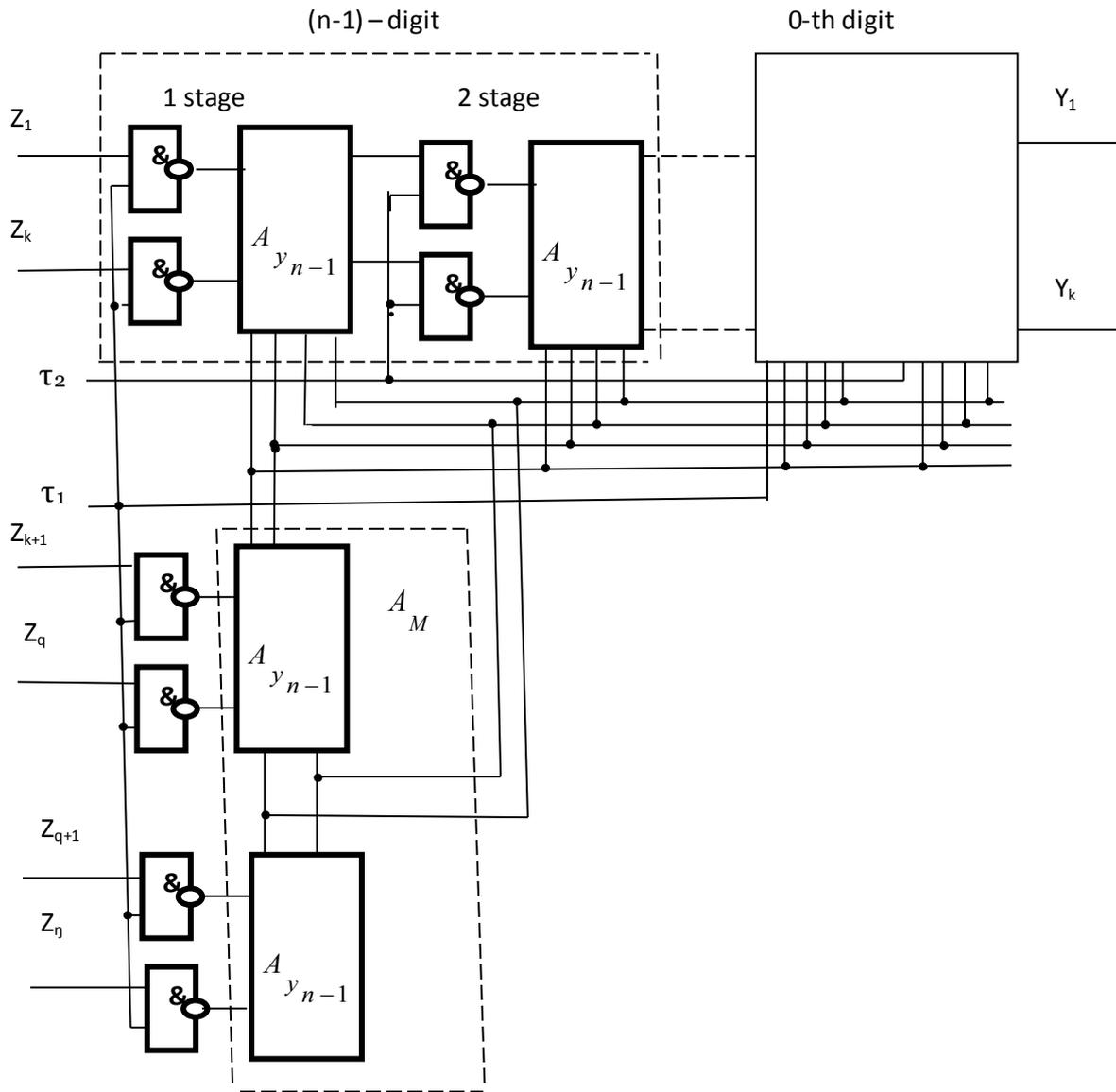
Such shift registers on the MUSP have an increased power of storage states in each  $i$ -th bit than with binary registers (shift registers on flip-flops), which allows shifting at once modulo  $M_N$  ( $M_N$  is the number of storage states in one register bit).

The shift registers can also be implemented at the IFAP with a single AM strategy automaton. The scheme of the shift register on the MFRS with a single AM strategy automaton is shown in Fig. 6.9.

The work of the shift register on the MFIS is carried out in certain blocks  $\pi_j$  of its states, which are stored for the corresponding states of the  $A_M$  strategy machine. When only the states of the  $A_M$  strategy machine are changed in the shift register, a coarse transition occurs in the block  $\mu_i$  of the register states.

By the same principle, it is possible to implement shift registers to the left and reverse registers.

Applying parallel registers and shift registers, we see that they are able to remember simultaneously the general and particular information, to make transitions in two variable input signals  $x$  and  $e$ , to reconstruct the algorithm of their work, which in principle can not be done on registers with memory on the triggers.



**Fig. 6.9.** Reconfigured shift registers on the MFIS with a single machine strategy  $A_M$

## **6.5. Conclusion to chapter 6**

In Chapter 6 registers are considered parallel, shifting and reversing on the MUSP, which have fewer hardware costs of logical elements for one memorized state, a matrix structure for storing states. These shift registers increase the shear rate by at least 4 times compared to the registers on the triggers.

## Chapter 7

### METHODS FOR CONSTRUCTING RECONFIGURED COUNTERS ON MULTI-LEVEL MEMORY CIRCUITS

#### 7.1. Basic concepts

The main task of the structural theory of automata is the study of the composition of automata, i.e. methods for constructing complex automata on simple automata. The theory of structural synthesis of automata makes it possible, based on general methods, to construct the structural diagrams of automata on the basis of the composition of a given finite number of standard automata. The goal of structural synthesis is the construction of a functional scheme that implements an automaton of logical elements of a certain type [1].

In structural synthesis, automata are not divided into asynchronous and synchronous, since in practice all automata - asynchronous and the stability of their states are provided by the introduction of synchronization. For simplicity, in the future we will enter synchronizing signals during the  $t$  cycle of automatic discrete or continuous time. In this case, the automata are synchronized by some independent synchronizing source (synchronizing signal generator).

A well-known canonical method of structural synthesis, according to which elementary automata of two types are used: monofunctional memory circuits of the second kind (triggers) and automata without memory (combinational circuits). Theoretical substantiation of the canonical method of synthesis of automata of the first and second kind, functioning in automatic discrete time, is the theorem on structural completeness [1].

Let us give our definition of the theorem on structural completeness, which V.M. Glushkov suggested back in 1962, from the standpoint of automatic continuous time [1].

Each system of elementary automata, containing a **single-stage monofunctional** automaton (Moore automaton with non-trivial memory), which has complete systems of transitions, outputs and **only one system of state conservation functions**, and any functionally complete system of logical elements is a structurally complete system. (In bold, the additions to the existing theorem are noted) .

In this theorem it is established for a more precise definition that an elementary automaton is one-step and monofunctional, as it corresponds to a Moore automaton with a nontrivial memory, and also that this elementary automaton has only one system of conservation functions. This refinement is necessary when compared with MFIS, which have several conservation functions (at least two), as well as with MUSP, which are multistage.

There is only one method (the canonical method), which makes it possible to reduce the problem of the structural synthesis of arbitrary automata of the first and second kind to the problem of synthesis of combinational schemes [1].

The restriction of this theorem does not allow us to construct Marakhovsky automata having large transitions at the inner cycle  $\Delta$  and functioning in the automaton continuous time  $T$ . To remove this restriction, we proposed an extended theorem on the structural completeness of elementary automata, proposed by L.F. Marakhovsky in 1996 [2].

We consider the extended theorem on structural completeness.

*Each system of elementary automata, containing an elementary multifunction machine (MFIS), which has complete systems of transitions, outputs, and a system of conservation functions (where the number of functions is not less than two), and any functionally complete system of logical elements is a structurally complete system.*

There is only one method (canonical method) that allows to reduce the problem of structural synthesis of arbitrary reconfigured automata of the 1st, 2nd and arbitrary automata of the third kind to the problem of synthesis of combinational schemes [2].

The theoretical basis for the canonical method for the synthesis of automata of the 1st, 2nd, and 3rd kind, functioning in automaton continuous time, is an extended theorem on structural completeness. This theorem makes it possible to construct

multifunctional automata of the first and second kind that have transitions during the time  $t$ , and automata of the third kind that have large transitions with the internal cycle  $\Delta$  and operate in the automaton continuous time  $T$  [2].

## 7.2. Methods for building reconfigured counters on multi-level memory circuits

A counter is an automaton that, by certain rules, calculates the input signals (pulses), which forms and stores the result of the count in some code [3].

An important characteristic of the meter is the  $K$  conversion factor (module, period) of the counter - the maximum number of input signals that the counter can calculate.

Consider the counter on multi-level memory devices (Figure 7.1). It can carry out a coarse transition in the internal cycle  $\Delta$  of the automatic continuous time  $T$ . On the basis of the extended theorem on the structural completeness, it is possible to construct counters, as well as any discrete devices that, in addition to the transition in cycle  $t$ , also have transitions during the internal cycle  $\Delta$  of automatic continuous time  $T$  [2].

Consider the method of structural synthesis of the counter unit 18 with the multifunctional system of organizing memory that has the ability to function as machines Marakhovsky 2nd and 3rd kind.

At each instant of time  $T$ , equal to the machine clock, the automaton can take the elementary input word  $p(T)$ , consisting of the input signals  $x(t)$  and  $e(\Delta)$ , making transitions from the state  $a_i(\Delta - 1)$  to the state  $a_k(\Delta)$  and output signals  $y_L \in Y$ .

The law of functioning of the deterministic Marakhovsky abstract automaton of the second kind is described by equations (1.12).

In the deterministic abstract automaton of the Marakhovskii of the second kind, the function  $\delta_o(a(\Delta-1), x(t))$  uniquely transforms the MFIS from the previous state  $a(\Delta-1)$  under the influence of the input signal  $x(t)$ , and the function  $\delta_e(a(t), e(\Delta))$  remembers the set state  $a(t)$  in the interval  $\Delta$  of the automatic continuous time  $T$  under

the influence of the  $e(\Delta)$  input signal. The function  $\varphi_2(a(t), a(\Delta))$  produces a shifted output signal  $y_L^2(T)$ , as in a single-stage RS trigger [3].

The law of functioning of a determinate abstract Marakhovsky automaton of the third kind is described by equations (1.13).

In the deterministic abstract automaton of the Marakhovskoy of the third kind, the function  $\delta_0(a(\Delta-1), x(t))$ , like in the second-kind automaton, uniquely transforms the MFIS from the previous state  $a(\Delta-1)$  under the influence of the setting  $x(t)$  of the input signal to a definite state  $a(t)$ , and the function  $\delta_y(a(t), e(\Delta))$  under the influence of the input signal preserving  $e(\Delta)$ , makes a large transition from the state  $a(t)$  to a definite state  $a(\Delta)$ . The function  $\varphi_3(a(\Delta), e(\Delta))$  outputs the shifted output signal  $y_L^3(\Delta)$ .

A characteristic property of deterministic Marakhovsky abstract automata of the second kind is the domain of admissible  $x(t) \in X$  input signals capable of putting the automaton into the state  $a(t) \in \pi_j$ , which is stored under the influence of the input signals  $e_j(\Delta) \in E$ , and, thus, to establish a single-valued state  $a_s(T)$  of the automaton. The input signals  $x_k(t)$ , that transfer the automaton to the state  $a_k(t)$ , which is not conserved by the subsequent action of the input signals  $e_j(\Delta) \in E$ , create the region of forbidden input words  $p_k(T)$  in determinate automata of Marakhovsky of the second kind.

In deterministic automata of Marakhovskoy of the third kind, the domain of admissible input words expands by using enlarged transitions that establish a new single-valued state  $a_s(\Delta)$  of the automaton. For deterministic automata, there is one forbidden input signal  $x_p(t)$ , which establishes a single-valued state  $a_p(t)$  of the automaton, which is not remembered for any input signal  $e_j(\Delta) \in E$ .

The definition of the law of functioning ends with the definition of an abstract automaton.

As the memory of the counter, you can select a multilevel scheme of memory class  $L_N^B$ , shown in Fig. 5.7. This scheme contains an MFIS and two flip-flops, each of which has three states (strategy automata for each MFIS group). Analysis of a

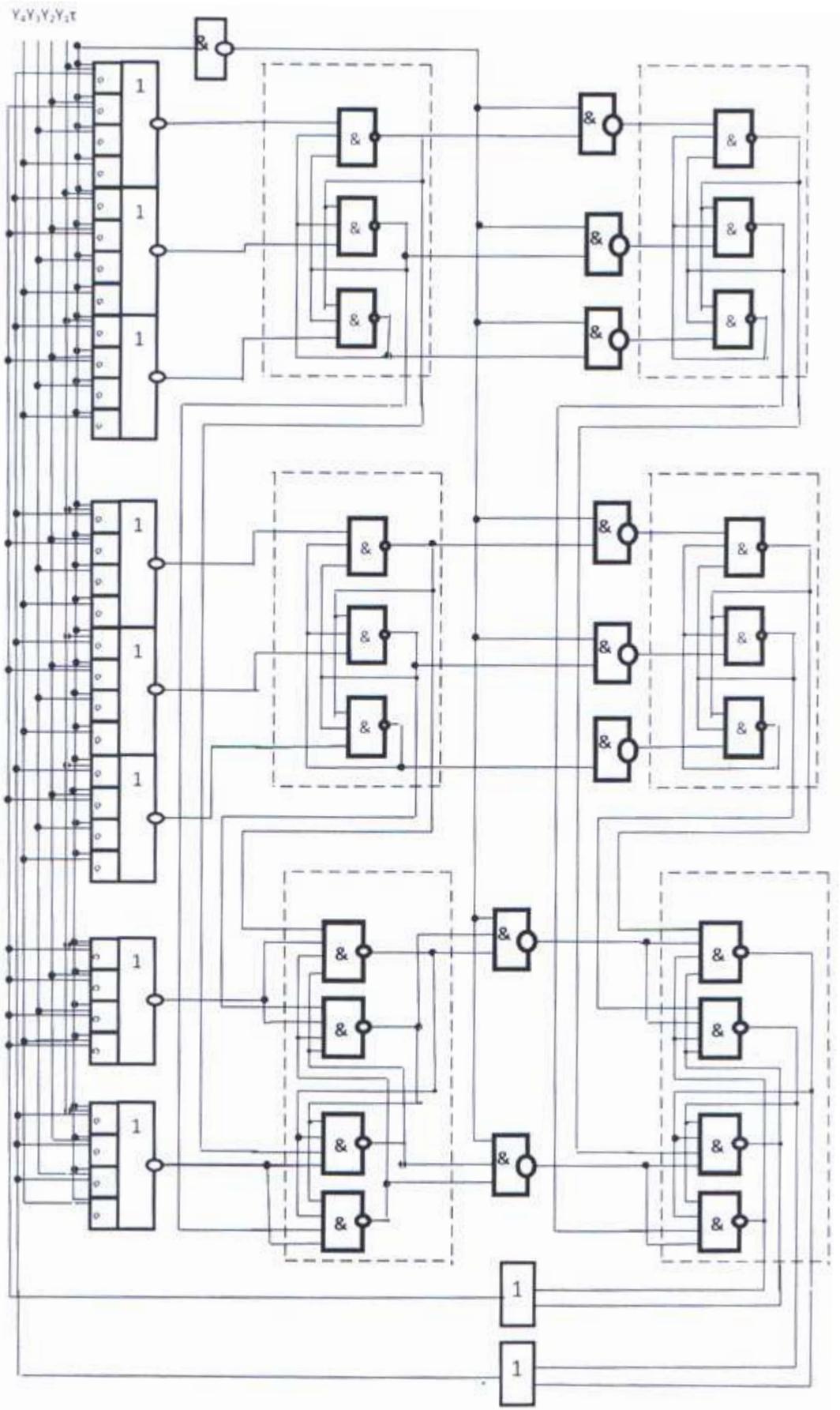
multi-level memory scheme (Figure 4.5), presented in Table. 4.6, showed that out of the 18 sets of  $x_i(t)$  input signals, which carry unambiguous transitions of the memory circuit to the corresponding states  $a_i(i = 1, 2, \dots, 18)$ , and Table. 4.8 - enlarged transitions that characterize the Marakhovskiy automaton of the third kind and are carried out under the influence of preserving  $e(\Delta)$  input signals.

We are constructing a two-stage scheme on an MFIS class  $L_N^B$ , in which the second stage is used as a delay of the output signal necessary for reliable operation of the counter, as it is done in two-stage triggers [3]. First, let us consider a system of synchronous signals (Figure 6.3), which is used in circuit diagrams of automata.

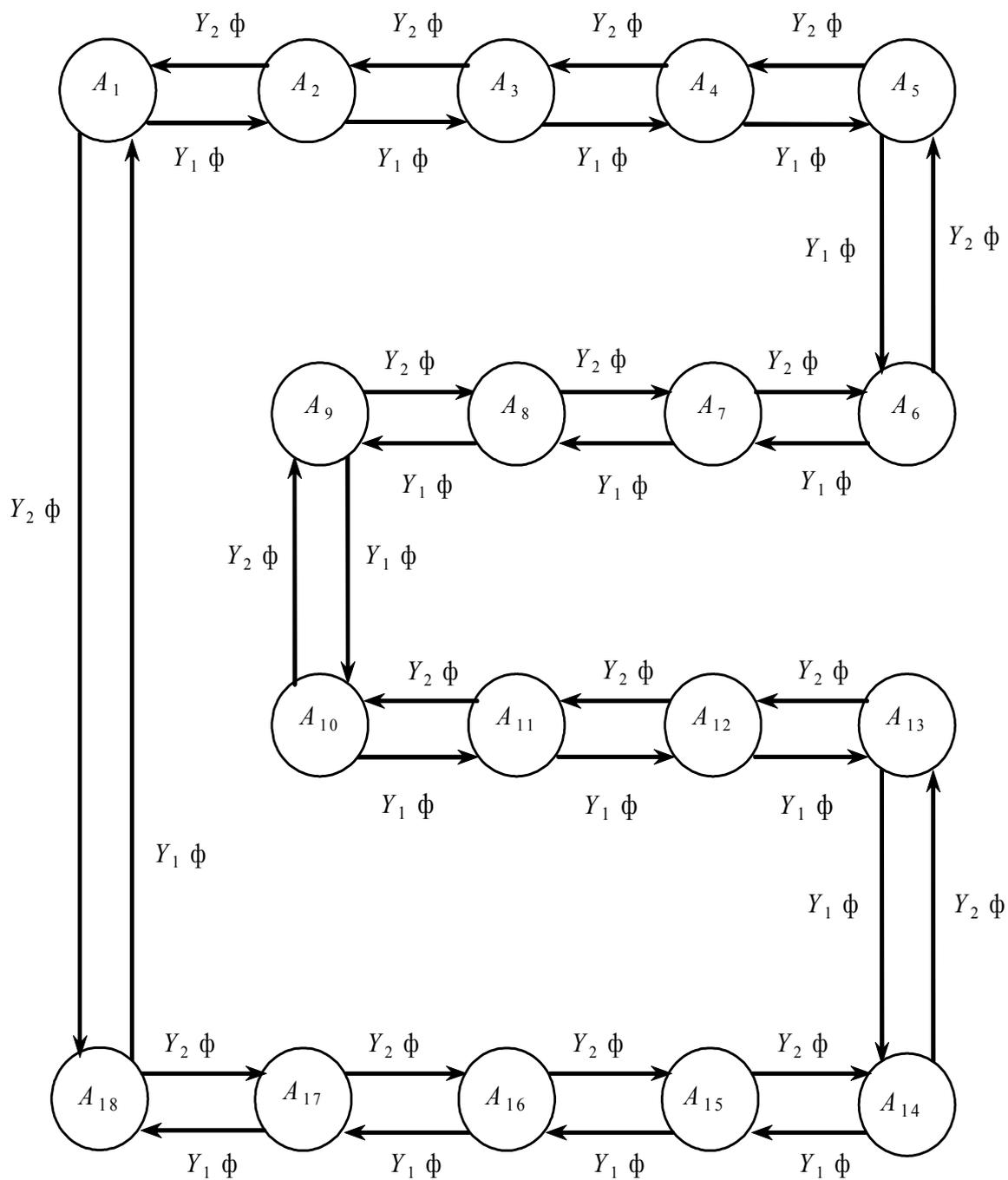
The machine clock  $T$ , which reflects the time interval from the appearance of one clock cycle  $\tau_1$  to the appearance of the next one, consists of two time intervals  $t$  and  $\Delta$ . The time interval between the two sync signals  $\tau_1$  and  $\tau_2$  is denoted by the symbol  $\Delta_0$ , and the time interval from the beginning of the clock signal  $\tau_1$  to the beginning of the clock signal  $\tau_2$  is indicated by the symbol  $T_0$ . It is very important to understand for the reliable functioning of the two-stage MFIS that during the time  $T_0$  the transients in the first stage of the MUSP must end in order that its output signals be stable before the appearance of the  $\tau_2$  clock.

A two-stage scheme of the counter memory on the MUSP class is shown in Fig. 7.1. Each stage of the MUSP is synchronized by the signal  $\tau_i (i = 1, 2)$ .

The algorithm for the operation of the counter (Figure 7.2), which is regarded as a Marakhovsky automaton of the second kind, consists in that the transitions to the MFIS (upper level) of the MUSP are carried out in one block of  $\pi_j$  states with constant stable states of the strategy automata (triggers for three states lower levels), as in the low-order counter.



**Fig. 7.1.** The scheme of the reverse counter on the MUSP class  $L_N^B$



**Fig. 7.2.** The algorithm of the counter as an automaton of the 2nd kind

To change the structure of memorization of states in the MFIS, we need to make a new transition in the triggers of the strategy. The MFP can change its algorithm of operation in this MFIS scheme nine times, working as an RS-type trigger. One of the triggers of the strategy can be taken as the second digit of the counter. The transition

in it depends on one state of the MFIS (for example, when the original node to the counter  $y_8$  the signal value is active, i.e., equal to 0). Another of the strategy triggers is then taken as the third digit of the counter. The transition depends on the same state of the MFIS ( $y_8 = 0$ ) and on the state in the counter discharge (for example, when the output node to the counter  $y_6$  the signal value is active, i.e., equal to 0).

Thus, the MFIS works as one of nine two-stage  $T$ -flip-flops, changing its state each time to the opposite under the influence of the cycles  $\tau_1$  and  $\tau_2$  in the period of the machine clock  $T$  (Fig. 7.3) under the  $Y_1$  and  $Y_2$  regimes of the reversing counter according to the algorithm presented in Fig. 7.2. To do this, the output nodes of the MFIS are connected as follows:  $y_8$  connects to the input node  $z_7$ , and  $y_7$  - to the input node  $z_8$ .

The second digit of the counter (trigger for three states), under the influence of transfer from the low-order counter and the right shift mode to  $Y_1$ , makes a transition from  $a_i$  to the next state  $a_{i+1}$ , and in the left shift mode,  $Y_2$  makes the transition from  $a_i$  to the previous state  $a_{i-1}$ .

To perform such actions, the AND-OR-NOT element is used at the input nodes of the second digit of the counter.

For one circuit AND, in addition to the signal  $\tau_1$ , there are also input signals of the  $Y_1$  or  $Y_2$  mode and the output signal from the node  $y_8$ . In  $Y_1$  mode, the output node  $y_4$  is connected to the input node  $z_5$ ,  $y_5$  - to the input node  $z_6$ ,  $y_6$  - to the input node  $z_4$ . In  $Y_2$  mode, the output node  $y_4$  is connected to the input node  $z_6$ ,  $y_5$  - to the input node  $z_4$ ,  $y_6$  - to the input node  $z_5$ . To construct such actions, the AND-OR-NOR element is also used at the input nodes of the third digit of the counter. For one circuit AND we supply, in addition to the signal  $\tau_1$ , input signals of the mode  $Y_1$  or  $Y_2$  and the output signal from the node  $y_8$  and  $y_6$ . In  $Y_1$  mode, the output node  $y_1$  is connected to the input node  $z_2$ ,  $y_2$  - to the input node  $z_3$ ,  $y_3$  - to the input node  $z_1$ . In  $Y_2$  mode, the output node  $y_1$  is connected to the input node  $z_3$ ,  $y_2$  - to the input node  $z_1$ ,  $y_3$  - to the input node  $z_2$ . The third digit of the counter under the influence of the transfer from the low and the second digits of the counter and the mode of shifting to the right  $Y_1$  makes a transition (similar to the second digit of the counter) with  $a_i$  to

the next state  $a_{i+1}$ , and in the left shift mode,  $Y_2$  makes the transition from  $a_i$  to the previous state  $a_{i-1}$ .

The algorithm for the operation of the counter, which is regarded as a Markovskiy automaton of the third kind, consists in the fact that the transitions in the low-level strategy automata due to the internal multifunctional memory organization of a multi- yayout enlarged MFIS transitions in the upper levels.

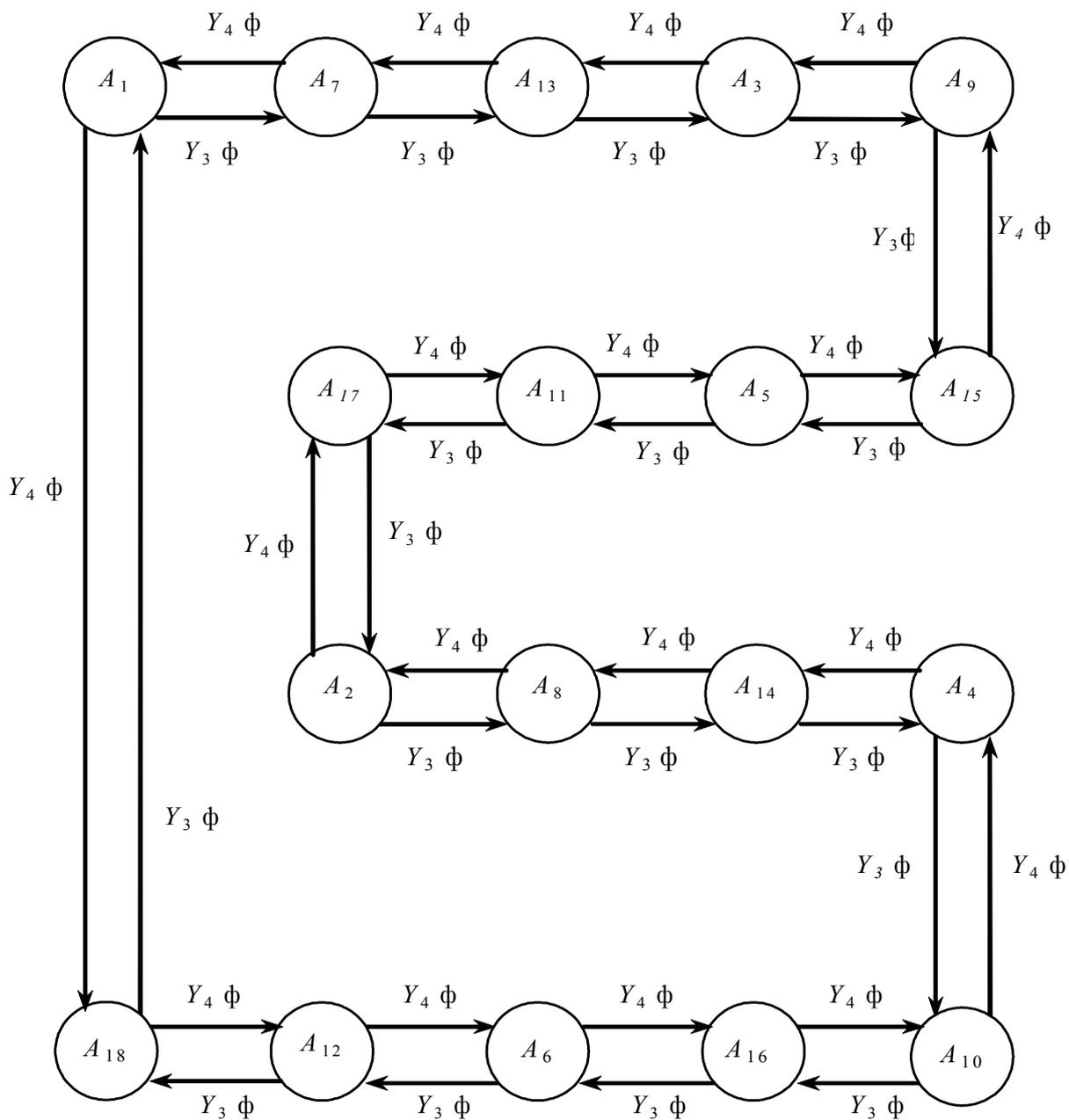
For this purpose, the trigger of the strategy automaton is selected as the low-order counter of the counter, the other trigger of the strategy automat is chosen as the second digit of the counter, and the MFIS is selected as the third digit of the counter.

You can use two  $Y_3$  modes, which is used to shift to the right and  $Y_4$ , which is used to shift to the left.

As we see from the algorithm of the counter operation (Fig. 7.3), as an automaton of the third kind, transitions in this case are carried out to other states of the counter that they are realized as enlarged ones. The construction of connections in flip-flops and MFIS is carried out both in the counter of the second kind, but in the low-order category they depend only on the operating modes of  $Y_3$  and  $Y_4$  and the sync pulses  $\tau_1$  and  $\tau_2$ . In the second digit of the counter (another trigger), the transitions from one state to another depend on the operating modes  $Y_3$  and  $Y_4$ , the sync pulses  $\tau_1$  and  $\tau_2$ , and the  $y_3$  low-order output of the counter.

Transitions to the MFIS from one state to another depend on the operation modes  $Y_3$  and  $Y_4$ , the sync pulses  $\tau_1$  and  $\tau_2$ , the low-order counter output  $y_3$ , and the second-digit counter output  $y_6$ . In the low-order counter of the counter in  $Y_3$  mode, the output node  $y_1$  is connected to the input node  $z_2$ ,  $y_2$  - to the input node  $z_3$ ,  $y_3$  - to the input node  $z_1$ .

In  $Y_4$  mode, the output node  $y_1$  is connected to the input node  $z_3$ ,  $y_2$  - to the input node  $z_1$ ,  $y_3$  - to the input node  $z_2$ . The transition of this bit depends only on the clock pulses. The second digit of the counter (trigger for three states), under the influence of the transfer from the low-order counter and the right-shift mode to the right,  $Y_3$  makes the transition from  $a_i$  to the next state  $a_{i+1}$ , and in the left shift mode,  $Y_4$  makes the transition from  $a_i$  to the previous state  $a_{i-1}$  (Figure 7.1).



**Fig. 7.3.** The algorithm of the counter as an automaton of the third kind

To perform such actions, the AND-OR-NOR element is used at the input nodes of the second digit of the counter. For one circuit AND we supply, in addition to the synchronous pulse  $\tau_1$ , the input signals of the  $Y_3$  or  $Y_4$  mode and the output signal from the  $y_3$  node of the low-order counter.

In  $Y_3$  mode, the output node  $y_4$  is connected to the input node  $z_5$ ,  $y_5$  - to the input node  $z_6$ ,  $y_6$  - to the input node  $z_4$ . In  $Y_4$  mode, the output node  $y_4$  is connected to the input node  $z_6$ ,  $y_5$  - to the input node  $z_4$ ,  $y_6$  - to the input node  $z_5$ . To perform such

actions, the AND-OR-NOR element is also used at the input nodes of the third digit of the counter. For one circuit AND we supply, in addition to the synchronous pulse  $\tau_1$ , the input signals of the  $Y_3$  or  $Y_4$  mode and the output signal from the node  $y_3$  and  $y_6$ .

The combined states of a two-level memory device have active output signals in only three output nodes. Let's imagine these active output signals in Table 7.1.

The higher-level MFIS states are characterized by the fact that only in one group the value of the active structure output signal  $y_i$  is zero. The state of the counter memory is characterized by a set of states of the upper-level MFIS and the strategy automata (Table 7.1).

The counter scheme realized on a two-stage memory device that can perform various modes of  $Y_i$  ( $Y_1 - Y_4$ ) operation in module 18 is shown in Fig. 7.1.

The counter scheme is built traditionally. The two-level upper-level MFIS is constructed as a  $T$ -type trigger, which is considered modulo 2, and the strategy automata as counters modulo 3.

When the synchronization signal  $\tau_1$  appears, the value of the first stage of the counter changes according to the operation mode  $Y_i$  ( $Y_1 - Y_4$ ) and the corresponding operation algorithm (Figures 7.2 to 7.3). When stable output signals appear at the output nodes of the first stage of the counter and the synchronous signal  $\tau_2$ , the values of the first stage of the counter are rewritten in the second stage of the counter. These new functionality of large-scale transitions can be used in control devices on multi-level memory circuits.

Thus, it can be clearly seen that the reversing counter on the MUSP can have four operating modes, exceeding the functionality of the reversing counters on the triggers, which have only two modes.

These new functionality of the enlarged transitions can also be used in control devices on multi-level memory circuits.

Table 7.1

Output signals of the second stage of the counter states

$x_i$	$z_1$	$z_2$	$z_3$	$z_4$	$z_5$	$z_6$	$z_7$	$z_8$	Output signals $y_i$	States $A_i$
$x_1$	0	1	1	0	1	1	0	1	$y_1 y_4 y_7$	$A_1$
$x_2$	0	1	1	0	1	1	1	0	$y_1 y_4 y_8$	$A_2$
$x_3$	0	1	1	1	0	1	0	1	$y_1 y_5 y_7$	$A_3$
$x_4$	0	1	1	1	0	1	1	0	$y_1 y_5 y_8$	$A_4$
$x_5$	0	1	1	1	1	0	0	1	$y_1 y_6 y_7$	$A_5$
$x_6$	0	1	1	1	1	0	1	0	$y_1 y_6 y_8$	$A_6$
$x_7$	1	0	1	0	1	1	0	1	$y_2 y_4 y_7$	$A_7$
$x_8$	1	0	1	0	1	1	1	0	$y_2 y_4 y_8$	$A_8$
$x_9$	1	0	1	1	0	1	0	1	$y_2 y_5 y_7$	$A_9$
$x_{10}$	1	0	1	1	0	1	1	0	$y_2 y_5 y_8$	$A_{10}$
$x_{11}$	1	0	1	1	1	0	0	1	$y_2 y_6 y_7$	$A_{11}$
$x_{12}$	1	0	1	1	1	0	1	0	$y_2 y_6 y_8$	$A_{12}$
$x_{13}$	1	1	0	0	1	1	0	1	$y_3 y_4 y_7$	$A_{13}$
$x_{14}$	1	1	0	0	1	1	1	0	$y_3 y_4 y_8$	$A_{14}$
$x_{15}$	1	1	0	1	0	1	0	1	$y_3 y_5 y_7$	$A_{15}$
$x_{16}$	1	1	0	1	0	1	1	0	$y_3 y_5 y_8$	$A_{16}$
$x_{17}$	1	1	0	1	1	0	0	1	$y_3 y_6 y_7$	$A_{17}$
$x_{18}$	1	1	0	1	1	0	1	0	$y_3 y_6 y_8$	$A_{18}$

### 7.3. Conclusion to Chapter 7

The reverse counter on the MUSP is considered, which has 4 operating modes, which it is not possible to implement on the triggers.

## **Chapter 8.**

# **CONTROL DEVICE ON MULTI-LEVEL MEMORY CIRCUITS**

### **8.1. Basic concepts**

Microprogramming is not a new concept in the technology of computer control devices [1-2]. Its appearance is usually associated with the name of Wilkes, who proposed it in 1951. The microprogram control is used as an orderly and flexible means of technical implementation of computer management. Microprograms are written in the special matrix processor memory of the processor - usually constant, because the set of computer operations by the computer programmer-user used is unchanged and is set when the microprocessor is made.

For users of modern computers, microprogramming appears as an engineering technique for the internal realization of a computer that does not directly affect its external properties. Only a computer developer and manufacturer is able to assess the significant advantages of microprogramming in terms of managing the internal structure of the computer and making it easier to make the necessary changes to it. The microprogram includes an extensive set of micro-operations necessary for the direct use of commands of high-level algorithmic languages or for software compatibility of small machines with high-performance ones.

By creating the necessary set of firmware, the developer is able to give the computer any structural properties: you can make the computer software compatible with another computer, which can borrow programs; and, finally, you can improve the performance of the computer to solve a certain class of problems by selecting the most suitable for this class of composition of machine operations [1].

One of the limitations of microprogram control is an unchanged set of microprograms for a particular class of instructions for each microprocessor. It is known that universal computers often give way to the efficiency of solving a certain class of tasks for specialized computers. In this regard, this chapter proposes to consider a

control device that is able to change the command system necessary to effectively solve a particular class of tasks.

## **8.2. Methods for constructing a reconfigured control device on multi-level memory circuits**

The control device (CU) in modern computers is a part of the central information processing unit (processor) intended for automatic control of the computing process, which ensures the coordination of operation of all devices of the computer, using synchronizing and control signals that are generated during program execution [1].

The control devices of modern computers are distinguished by the use of new computers and the improvement of the previously known principles of organizing computers. The most widely known for performing CU in the integrated components of the processors is a CU with a Wilkes-Stringer matrix structure and using registers on flip-flops, a microprogram control scheme with two matrices. This scheme was used in model 45 of the Spectra 70 system of RCA, 360 system and in many others [1].

The disadvantage of these structural schemes of a matrix-structured matrix is that due to the implementation of memory on flip-flops, only sequential processing of local and general information is implemented in register structures.

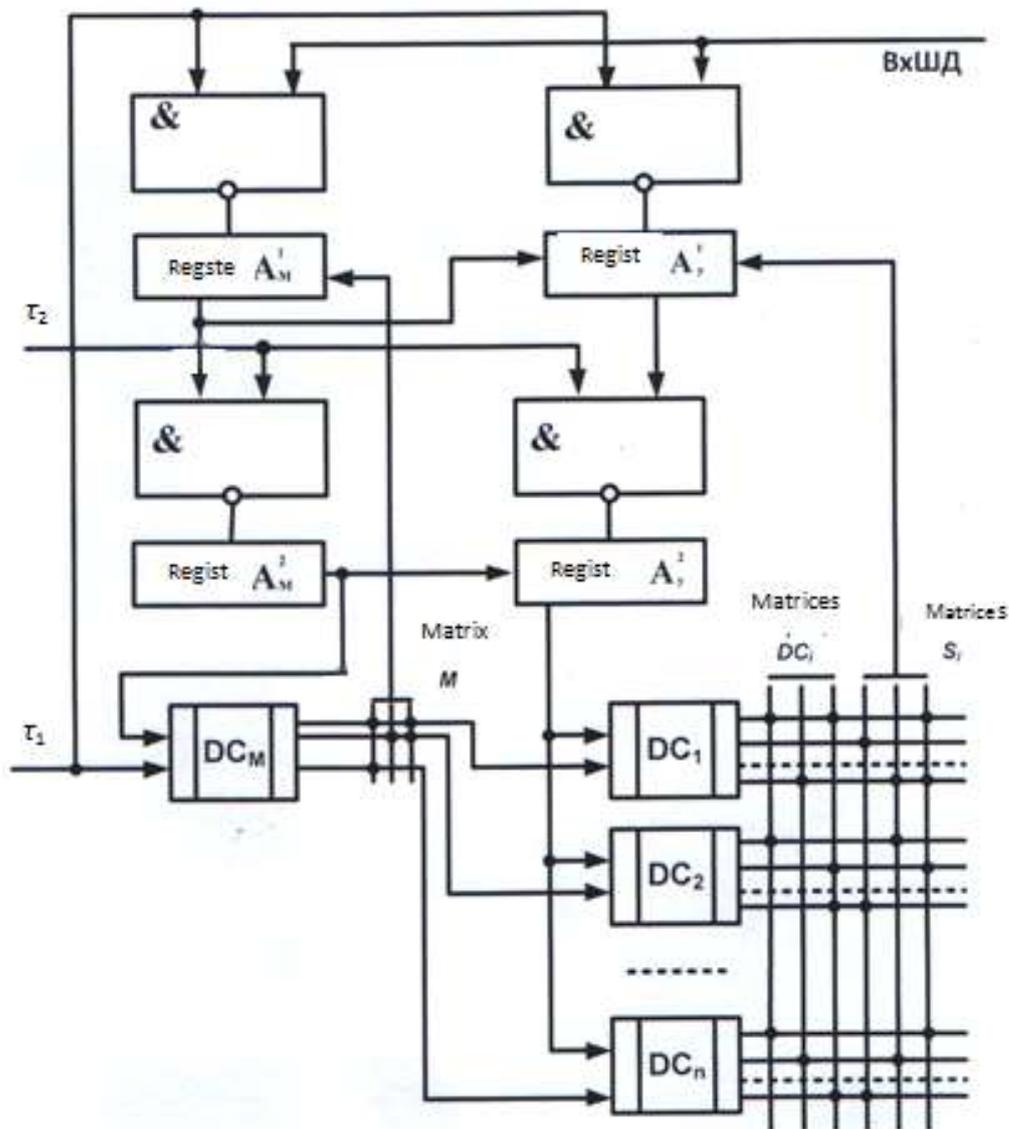
The control device with a matrix structure on the MUSP uses a strategy register  $A_M^1$ , whose outputs are connected to the storing inputs of the register  $A_y^1$  and the input of the NAND element, the second input of which receives the sync signal  $\tau_2$  [2]. The outputs of the register  $A_y^1$  are connected to the input of the second NAND gate, to the other input of which the synchronous signal  $\tau_2$  is input. The outputs of two NAND elements are connected respectively to the input nodes of the strategy register  $A_M^1$  and the controlled register  $A_y^2$ . The outputs of the register  $A_M^2$  are connected to the storing inputs of the register  $A_y^2$  and the input nodes of the decoder  $DC_M$ , on the

input buses of which the clock signal  $\tau_1$  comes. On the output buses of the  $DC_M$  decoder, a matrix  $M$  is created, whose outputs are connected to the input nodes of the strategy register. The outputs of the register and the output buses of the  $DC_M$  decoder are connected to the input nodes of the decoders  $DC_i$  ( $i = 1, \dots, n$ ), on the output buses of which there are created a matrix  $C_i$  of microinstructions and a matrix  $S_i$  of transitions whose outputs are connected to the output nodes of the register  $A_y^1$ . The other input nodes of the registers  $A_M^1$  and  $A_y^1$  are connected through the NAND elements with the nodes of the input bus of the BxIIIИ and the input of the synchronization signal  $\tau_2$  (Figure 8.1).

Structurally, the CU is built on the MUSP differs from the CU built on triggers, because the CU on the MUSP (Figure 8.1) stores the general information of the algorithm in the strategy registers  $A_M^1$  and  $A_M^2$  the particular information of the algorithm in the registers  $A_y^1$  and  $A_y^2$ , has a decoder of the strategy register with the matrix  $M$  to which the clock signal  $\tau_2$ , intended for organizing the transition functions in the general part of the algorithm, and a number of  $DC_i$  decoders ( $i = 1, 2, \dots, n$ ), on the output buses of which the  $C_i$  micro-arrays and  $S_i$  -transitions matrices are organized, transitions in a separate part of the algorithm.

Functionally, in CU, an algorithm can be used for changing the time (depending on the general input information) to the own response to those or other states of the registers, simultaneously processing general and local information.

The strategy register  $A_M^2$  contains the address of the general information that determines the subset of register  $A_y^2$  states controlled by the strategy register  $A_M^2$  and the  $DC_M$  decoder selects the corresponding  $DC_i$  decoder ( $i = 1, 2, \dots, n$ ), the contents of the register  $A_y^2$  in this subset contain the address of the current macro that is in the process of execution. At the beginning of the next cycle, the synchronous signal  $\tau_2$  arrives, during which the contents of the registers  $A_M^1$  and  $A_y^1$  are transmitted through the corresponding NAND valves to the registers  $A_M^2$  and  $A_y^2$  to decode them and then select the next micro-command.



**Fig. 8.1.** Device management in multi-level memory circuits

Functionally, in CU on an MUSP, you can use an algorithm that changes over time (depending on the general input information) your own response to those or other states of the registers, processing both general and private information.

The strategy register  $A_M^2$  contains the address of the general information that determines the subset of register  $A_y^2$ , states controlled by the strategy register  $A_M^2$  and the  $DC_M$  decoder selects the corresponding  $DC_i$  decoder ( $i = 1, 2, \dots, n$ ), the contents of the register  $A_y^2$  in this subset contain the address of the current macro that is in the process of execution. At the beginning of the next cycle, the synchronous signal  $\tau_2$  arrives, during which the contents of the registers  $A_M^1$  and  $A_y^1$  are transmitted through

the corresponding NAND valves to the registers  $A_M^2$  and  $A_y^2$  to decode them and then select the next micro-command.

When using such a scheme, the sequence of microinstructions is determined either by the matrix that defines the sequence of actions, or by the introduction of the initial address of the general and private information at a fixed moment of the machine cycle through the input bus of the BxIIIД, which is determined by the clock signal  $\tau_2$ . This initial address can be the code of a subset of operations when it is perceived by the register  $A_M^1$  and the operation code when it is perceived by the register  $A_y^1$ , or by a binary group in the code field of the operation specially assigned to it to distinguish it from the other remaining operation codes.

### **8.3. Conclusion to the 8th chapter**

In this chapter, the structure of the device is controlled in multi-level memory circuits, capable of changing the structure of the microprocessor's instructions without losing performance for one machine clock, and also process the general and local information simultaneously.

## Chapter 9

### SUMMERS ON THE MATRIX SCHEMES OF THE MUSP

#### 9.1. Basic concepts

The adder is a node performing the operation of arithmetic addition (summation) of two numbers (words). By addition, we mean the formation of a word with numerical values  $S (S_1, S_2, \dots, S_n)$ , equal to the sum of the numerical values of the two original words  $x (x_1, x_2, \dots, x_n)$  and  $y (y_1, y_2, \dots, y_n)$  [1]. These words are stored in the registers of the processor. Usually in binary parallel processor registers.

The values of the sum of the  $i$ -th digits of the word and the  $i$ -th transfer are formed in accordance with the rule of ordinary arithmetic:

$$\begin{cases} S_i = x_i + y_i + p_{i-1}, & p_i = 0, \text{ at } = x_i + y_i + p_{i-1} < q: \\ S_i = x_i + y_i + p_{i-1} - q, & p_i = 1, \text{ at } = x_i + y_i + p_{i-1} \geq q, \end{cases}$$

where  $S_i$  is the value of the sum in the  $i$ th digit;

$p_{i-1}$  - transfer from the next minor digit;

$p_i$  - transfer to the next highest digit;

$q$  is the base of the number system.

The low bit of the adder has no input carry and its sum value is formed in accordance with this rule:

$$\begin{cases} S_i = x_i + y_i, & p_i = 0, \text{ at } = x_i + y_i < q: \\ S_i = x_i + y_i - q, & p_i = 1, \text{ at } = x_i + y_i \geq q, \end{cases}$$

Depending on the base of the number system  $q$  and the adopted coding system, the adder: binary, ternary, decimal, binary-decimal, etc., and, depending on the number of inputs, half-summers with two (HS) and total adders with three (SM) inputs are

distinguished. According to the method of organizing the summation process by a single-digit summing circuit, the summaters are: combinational, accumulating and combinational-accumulating type. By the method of processing multi-digit numbers distinguish such types of adders: sequential, parallel and parallel-sequential adder. According to the method of organization of transfer chains between bits, summers with consecutive, through, group and simultaneous transfers are distinguished [39].

A feature of modern adders is the use of binary triggers in registers when storing the original numbers and the result of addition. This feature also defined methods for constructing combinational combinators: binary and binary-decimal [39; 40].

The processing time, for example, of a 32-bit parallel adder is basically determined by the delay time of the carry from the low to the high. The delay time  $t_{\text{delay}i}$  in a single-digit scheme is much less than the time  $t_{pp}$  in the chains of the formation of the 31-bit transfer ( $t_{\text{delay}S} \ll t_{\text{delay}i}$ ).

So, for example,  $t_{\text{delay}S} = 2\tau_e$ , and  $t_{\text{delay}i} = 31\tau_e$ , where  $\tau_e$  is the delay on one logical element. In this regard, all the forces of developers were aimed at reducing the time of transfer in parallel adder.

In our case, we will use the parallel register bit on the multi-level automatic memory scheme (Figure 6.7), which stores 18 states instead of two. Thus, the number of digits of the 32-bit register on the flip-flops will be reduced to 8, which will drastically reduce the number of transfers in the adder if you create a 16-bit adder and 7 hyphenations in the adder.

The proposal to create transfers at once to two digits, to three digits, and so on, did not find its application because of the awkwardness of the schemes for binary adders [1].

Thus, the application of a sequential full 16-bit adder with transfer and shift register storage to the MUSP or parallel adder using the register on the MUSP reduces the stacking time by a factor of 4.

## 9.2. The adder on the matrix structures of the MUSP

Multilevel memory schemes [2-3], unlike 4 triggers that store 16 states, store 18 states and require 8-bit code (one byte of information) to write to them in the corresponding number. Each state of the MUSP (Figure 5.3.) Is presented in Table 9.1.

Table 9.1

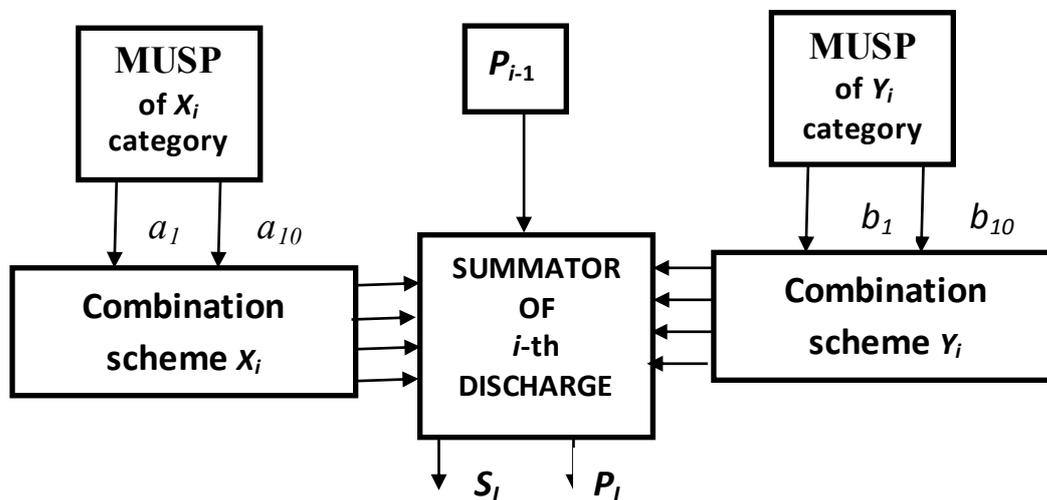
Output signals of the MUSP

Numbers in decimal notation	Value of output nodes	4-digit number code	Schema states $A_i$
	$a_i$	$z_1 z_2 z_3 z_4$	$A_i$
0	0 1 1 0 1 1 1 0 1 1	$a_1 a_4 a_8 \overline{a7}$	$A_1$
1	0 1 1 0 1 1 1 1 1 0	$a_1 a_4 a_{10} \overline{a9}$	$A_2$
2	0 1 1 1 0 1 1 0 1 1	$a_1 a_5 a_8 \overline{a7}$	$A_3$
3	0 1 1 1 0 1 1 1 0 1	$a_1 a_5 a_9 \overline{a10}$	$A_4$
4	Regist 1 1 0 1 0 1 1	$a_1 a_6 a_8 \overline{a7}$	$A_5$
5	0 1 1 1 1 0 1 1 0 0	$a_1 a_6 a_9 a_{10}$	$A_6$
6	1 0 1 0 1 1 0 1 1 1	$a_2 a_4 a_8 \overline{a7}$	$A_7$
7	1 0 1 0 1 1 1 1 1 0	$a_2 a_4 a_{10} \overline{a9}$	$A_8$
8	1 0 1 1 0 1 0 1 1 1	$a_2 a_5 a_7 \overline{a8}$	$A_9$
9	1 0 1 1 0 1 1 1 0 1	$a_2 a_4 a_9 \overline{a10}$	$A_{10}$
10	1 0 1 1 1 0 0 1 1 1	$a_2 a_6 a_8 \overline{a7}$	$A_{11}$
11	1 0 1 1 1 0 1 1 0 0	$a_2 a_6 a_9 a_{10}$	$A_{12}$
12	1 1 0 0 1 1 0 0 1 1	$a_3 a_4 a_7 a_8 \overline{a7}$	$A_{13}$
13	1 1 0 0 1 1 1 1 1 0	$a_3 a_4 a_{10} \overline{a9}$	$A_{14}$
14	1 1 0 1 0 1 0 0 1 1	$a_3 a_5 a_7 a_8 \overline{a7}$	$A_{15}$
15	1 1 0 1 0 1 1 1 0 1	$a_3 a_5 a_9 \overline{a10}$	$A_{16}$
16	1 1 0 1 1 0 0 0 1 1	$a_3 a_6 a_8 \overline{a7}$	$A_{17}$

17	1 1 0 1 1 0 1 1 0 0	$a_3 a_6 a_9 a_{10}$	$A_{18}$
----	---------------------	----------------------	----------

Coding of output signals occurs depending on the number code (Table 9.1) of the memory circuits that make up the structure of the multi-level memory scheme of the class (see Figure 5.3). Active zero output signals are a number from three to four variables. Output signals 4, therefore, in the group of three variables an inverse one variable is added (Table 9.1).

Thus, a full hexadecimal adder receives two numbers from each  $i$ -th bit of the register to the MUSP and carry over from the previous digit of the adder. The structure of such an adder looks like this (Figure 9.2).



**Fig. 9.2.** Structural diagram of the  $i$ -th bit of the adder on the MUSP

The adder itself can be implemented on an MUSP, in which the MFIS consists of as many groups as are determined by the number system  $q$ . For example,  $q = 2$ , then it is sufficient to use an MUSP class  $L_N^B$  (see Figure 5.3), which has a  $2 \times 3$  matrix structure (two groups in the MFIS and a three-bit register in the strategy automaton of each group). The transition from the one group MFIS of states to another one defines a binary number of one digit of the number  $X$ , and the transition in the three-digit trigger of the strategy automaton determines the binary number of one digit of the number  $Y$  and the transfer of  $p_{i-1}$  from the previous digit of the adder.

Thus, the MUSP passes into a state that determines the sum of  $S_i$  and the transfer of  $p_i$  to the next digit of the adder. The transfer sum  $p_i$  can be 0 or 1, depending on the sum of the input signals.

If  $q = 10$ , then it is used for the adder from the MUSP, in which the MFIS has 10 groups of 4 logic elements, which makes up a matrix of  $10 \times 15$ . For  $q = 16$ , it is used for the adder of the MUSP, in which the MFIS has 16 groups of 5 logical elements, which makes up the matrix of  $16 \times 31$ .

The construction of an MUSP, used as an adder, is described in the second section of this work.

Acceleration of the information processing time when using the adder on the matrix structures of the ICMP is carried out due to:

- Firstly, for one machine clock  $T$  the transfer is determined in a decimal or in a 16-bit adder, which in itself increases the speed by 4 times by comparison with a full binary adder for a decimal or a 16-bit adder;
- Secondly, it makes it possible to reduce the number of digits in the registers on the MUSP by 4 times compared to the registers on the triggers, which is very important for increasing the speed with the use of shift registers.

### **9.3. Conclusion to the 9th chapter**

The device control extends the capabilities of the microprogram control unit, processing information common and local simultaneously, which modern microprogram control devices fail to perform.

# **Chapter 10**

## **METHODS FOR BUILDING RECONFIGURED PROCESSORS AND COMPUTERS ON AUTOMATIC MEMORY CIRCUITS**

### **10.1. Basic concepts**

Since the beginning of the twentieth century, a formal classical theory of algorithms has been constructed that specified the possibility of theoretical computation for practical application in cybernetics and programming. Among these algorithms one can single out the most significant ones: arithmetic calculi of Gödel predicates, Post and Turing machines, Markov automata, Janov schemes, block diagrams, learning algorithms systems [1].

The most interesting are learning algorithms that change over time (depending on the preliminary incoming general information) their response to input words [1]. However, they, like all sequential algorithms, have limitations that do not allow simultaneous processing of general and particular information, which reduces the processing speed of the algorithm.

The process of designing processors, the most complex components of a computer and computing circuits (complexes) consists of solving a certain set of tasks, as a result of which the architecture, structure and functional schemes of the processor units are determined [2]. The basis of each structural description of a complex device is the aggregate of components and connections between them. Such a description is often multilevel because each component of the described device itself can have a structural description and then it consists of lower-level components.

With a systemic multi-level approach to this situation, there are methods for building reconfigured processors and computers on automatic memory circuits that can simultaneously process general and local information [3-9].

### **10.2. Methods for building the reconfigured architecture and processor structure on the MFIS and MUSP**

A processor is a device for automatically executing a sequence of operations that are caused by a program for solving a problem. It consists of two devices: operating and control. The fourth-generation processors also include internal (processor) memory devices and input-output control devices [10]. A processor is a device for automatically executing a sequence of operations that are caused by a program for

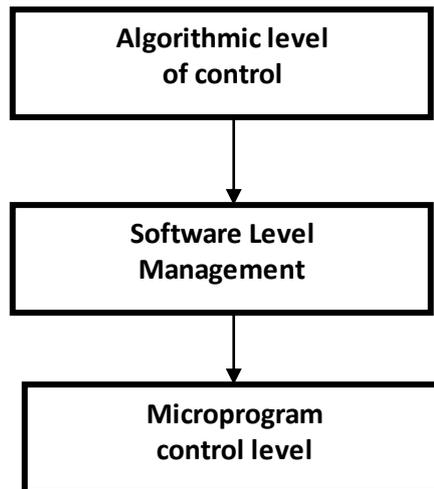
solving a problem. It consists of two devices: operating and control. The fourth-generation processors also include internal (processor) memory devices and input-output control devices [10].

The operating device (OR) performs the transformation of the arithmetic and logical operation (Figure 9.2), and the control device (CU) performs the control of the computing process (Figure 8.1), determines the sequence of operations, directs the selection of instructions from memory, together with the synchronization circuit generates control signals that control the execution of elementary actions (micro-operations). The presence of internal memory (general-purpose registers, cache memory, etc.) in modern microprocessors allows them to implement part of the software (internal). In this connection, the processor means are divided into software (hardware) and hardware (hardware). In modern computers, the interpretation of microprocessor algorithms is performed by microprogramme tools [11-13].

The combination of the characteristics of software and hardware constitute the concept of the architecture of computers and processors. There are four main groups of architectural characteristics of processors:

- characteristics of the internal language and software;
- technical and operational characteristics;
- characteristics of functional modules and extended configuration of the processor and computer;
- characteristics of the interface and interrupt systems.

The characteristics of the first group determine the algorithmic capabilities of the processor. In this connection, in modern computers three levels of the internal language are distinguished, to which three levels of control correspond: algorithmic, program and microprogram [2]. Each of the levels can perform two main functions: to serve as a universal means of displaying the input language (that is, the language on which the task algorithm is formulated) and the means of interpreting some operators through others.



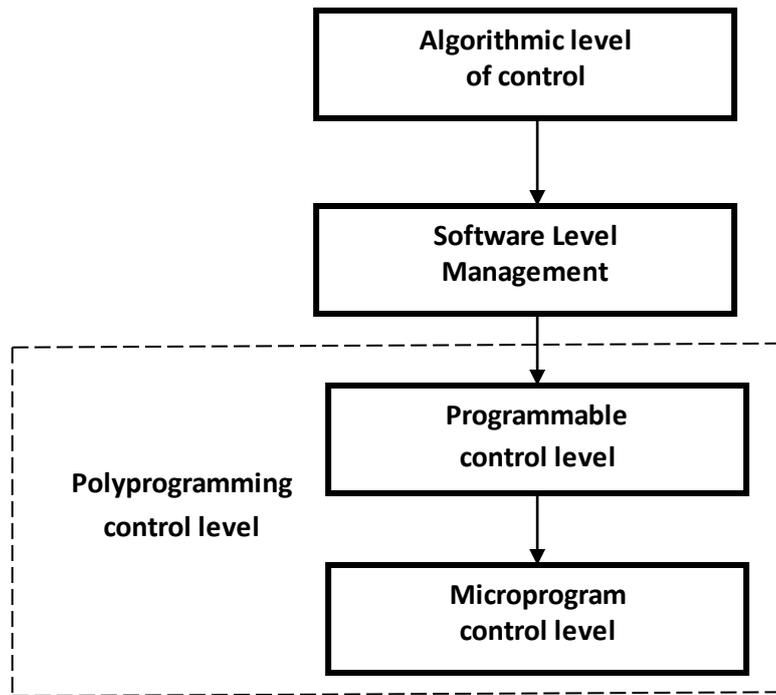
**Fig. 10.1.** Levels of control in the computer

At the same time, all levels of management are in a certain hierarchical connection, which allows you to make an expression of an operator of a higher level through operators of a lower level (Figure 10.1).

The nature of the connections between the control levels, as well as the functions of each of them, most significantly reflects the architecture and structure of the processors.

The principle of hierarchical program management, proposed by L.F. Marakhovsky [3], which breaks the control information into  $n$  levels, allows you to enter the fourth level of management mili-program, which is common with respect to the firmware level. Miliprogram and microprogram levels are combined into a polyprogram level that provides processing of the general and private information simultaneously (Figure 10.2) [3]. This makes it possible to increase the processing speed of information in the class of training algorithms (and other reconfigured ones) that change their response to those or other input words during the time under the influence of general information [14].

The structure of the processor is the aggregate of its functional blocks and the connection between them. The development of the structure, and with it the architecture, was aimed at maximizing the performance of processors, increasing the use of hardware part of software, etc. The principle of microprogramming is realized due to the inclusion in the processor structure of a special memory block for saving microprograms [11].



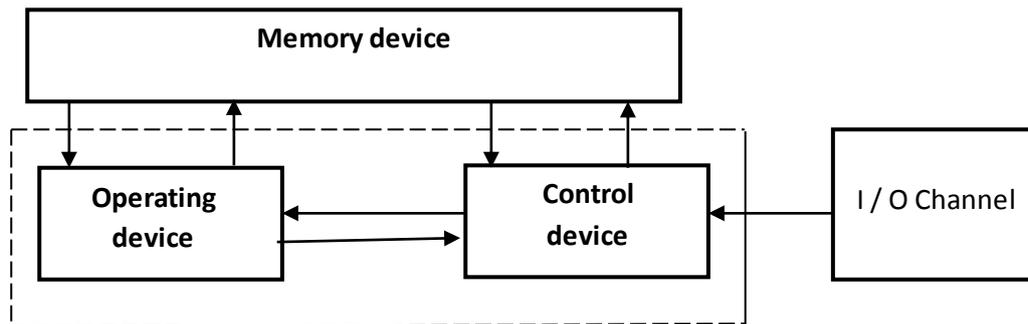
**Fig. 10.2.** Polyprogramming control level

The microprogram processors provide the programmer, in addition to the instruction language, an effective micro-command language, which is located in the main passive memory (ROM). Along with this, the principle of microprogramming simplifies the process of developing, modifying and modifying the command system, and is also a tool for flexibility in the functional orientation of computers and processors for solving entire classes of tasks.

The principle of constructing the polyprogramm processors is realized by including a special memory block in the processor structure for storing general information of the mil- lyprograms. This block provides additional capabilities in microprogram processors in the direction of increasing the modifications and changing the

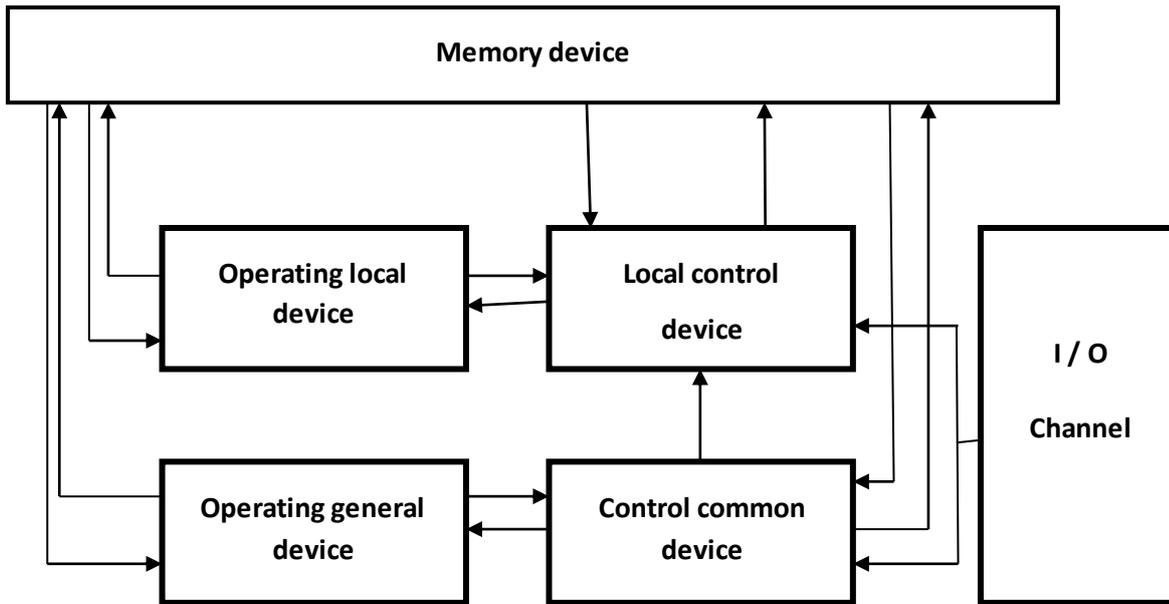
command system and even in the process of work leads to the possibility of simultaneous processing of general and local information.

The classical generalized microprocessor block diagram (Figure 10.3), which changes in accordance with the principle of program management, was proposed by C. Babbage in the 19-th century [2].



**Fig. 10.3.** Generalized scheme of the modern processor

The generalized block diagram of the polyprocessor is shown in Fig. 10.4. The polyprocessor changes in accordance with the principle of hierarchical program control. The structure of the polyprocessor consists of the proposed representation of the structural automaton, which considers multifunctional automata of Marakhovsky 1st, 2nd and 3rd kind [15], is specified as a composition of blocks: a hierarchical control and operational ones, the number of which depends on the number of levels of the control block (Figure 10.4).



**Fig. 10.4.** Generalized scheme of the processor

The development of the structure, and with it the architecture of the processors, was aimed at maximizing their performance, increasing the use of hardware part of the program. The principle of the construction of the processor is realized by including a special memory block in the processor structure to store general information (mili-programs). This provides additional capabilities in microprocessor processors in the direction of increasing modifications and changes in the command system during the operation of the control unit and substantially lead to the possibility of simultaneous processing of general and private information. A generalized block diagram of the polyprocessor on the MFIS and MUSP is shown in Fig. 10.5 [12].

The classical generalized block diagram of the microprocessor with the memory on the flip-flops changes, when the memory and the MUSP are used as the memory. It can become reconfigured. This is explained by the fact that in the program it is possible to divide the control information into general and local information to use it both in the control device [77], for changing the command system, and for processing the general and local information on two arithmetic logic units (ALU) Fig. 10.5).

Changing the command system in the processor (Fig. 10.5) does not require an additional machine clock to change the command system, which allows to increase the speed in the polyprocessor. The main task for the processor designer is to provide

the specified system-algorithmic capabilities of the computer with the help of the best structural solutions. The basis of the architectural and structural organization of the computer's polyprocessors is the use of the principle of hierarchical program management [75].

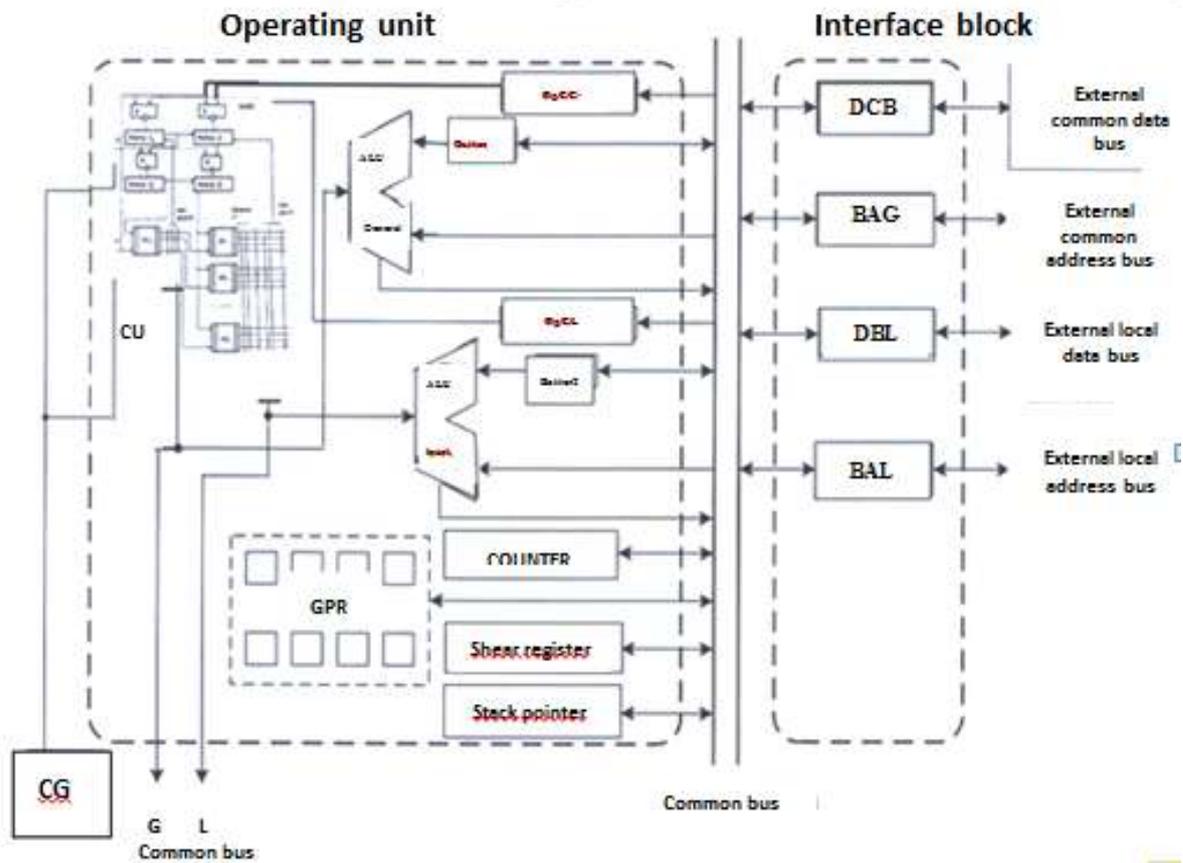


Fig. 10.5. Generalized block diagram of the polyprocessor on the MUSP

### 10.3. Methods for describing a polygram

When applying the principle of hierarchical program control and the theory of the Marakhovsky automatic machines (multifunctional 1st, 2nd and 3rd kind) [3], two lists of interrelated commands belonging to interconnected general and private instructions for hierarchical control of data processing and the data sets themselves being processed by these pairs of hierarchical instructions.

A visual way of specifying the classic Mealy or Moore automata with memory on the triggers based on micro-operations is the problem in the form of their firmware or in a more general form in autograms [11].

For an algorithmic description of the proposed hierarchical automata on the MFIS and / or on the MUSP with a general state consisting of the states of all memory circuits, we introduce the term "polygrams" that combines the term "microprogram" as general information and the term "autogram", as private information. This term "polygrams" has a large capacity for the following reasons:

- Firstly, the concept of "microprogram" and "autogram" and other ways of specifying microcode machines with memory on the triggers [2; 11] are oriented to well-grounded transition functions and exit functions in automata (only during the clock moment  $t$ ) and limit the analysis of the automaton operation only in the automaton discrete time [1; eleven].

- Secondly, the notion of "microprogram" and "autogram" and other ways of specifying microcode automata with memory on the triggers [11] specify the classical automata of the 1st and 2nd kind on the realization of storing the state in the register on the triggers.

- The third and main feature of the concept of "microprogram" and "autogram" and other ways of setting microcode machines with memory on the triggers is that they describe the states of automata of one set that does not change during operation and does not allow to design the configured devices of computer systems taking into account elementary memory circuits.

The notion of "microprogram" and "autogram" and other ways of specifying microprogrammed automata with memory on flip-flops [11] are not suitable for describing Marakhovsky automatic machines on the MFIS and MUSP, which are considered throughout the machine time (in automatic continuous time). They can change states that preserve private information, under the influence of states that process general information, in the process of working for one machine clock  $T$  [3].

The term "polygram" combines the term "milligram", which remembers, processes and provides general information, with the term "autogram" or "microprogram," which processes and gives out individual (local) information simultaneously (in parallel). The term "polygram" will be associated with the schematic implementation of management in multi-level registers with a multifunctional memory organiza-

tion. The polygraph is oriented not only to transforming input information into output, but also to changing subsets of states  $a_i$ , in which an automaton can function for a certain input signal that preserves  $e(\Delta)$ . This feature makes it possible to use a polygram description of hierarchical automata (AI) with memory on the MFIS and MUSP, which allow simultaneous processing of general and particular information for one machine clock cycle  $T$ .

The polygram describes each state of  $a_k$  AI as a union of the  $a_i$ -states of  $S_i$  subautomata (built on the MFIS).

$$a_k = \bigcup_i a_i. \quad (10.1)$$

At each point of the polygram, the operation mode of  $S_i$  subautomata is described for one external cycle  $T_0$  (machine clock) of automatic continuous time. It is used in the polygraph simply as  $T$ . For one external cycle,  $T$  IA A has the ability to accept the input word  $R_k(T)$ , consisting of a set of elementary  $p_i(T)$  input words of  $S_i$  subautomata.

$$R_k = \bigcup_i p_i. \quad (10.2)$$

Under the influence of the input word  $R_k$ , the IA  $A$  can go to a new state  $a_s$  and output the output signals  $Y_k$ , consisting of a set of output signals  $Y_i$  of certain three types of  $S_i$  subautomata of the 1 st, 2 nd and 3 rd kind.

$$Y_k^1(t) = \bigcup_i Y_i^1(t); \quad (10.3)$$

$$Y_k^2(T) = \bigcup_i Y_i^2(T); \quad (10.4)$$

$$Y_k^3(\Delta) = \bigcup_i Y_i^3(\Delta), \quad (10.5)$$

where the output signals  $Y_i^1(t)$ ,  $Y_i^2(T)$ ,  $Y_i^3(\Delta)$  are determined according to the functions of the outputs [3].





Algorithms for changing the general state  $e$  of the point  $KE$  of the polygram with variables that preserve  $e$  input signals are described by the milprogram, which looks similar to the microprogram or an autogram:

$$e. p_j \rightarrow Y_j^i - e_j. \quad (10.11)$$

The logic of a system in which internal connections are more important than external ones often confronts us with the difficulties of formulating a microgram (autogram) in a polygram. The change in information is more effective the more complex and better organized structure of the system it accompanies. The change in the internal structure of memorization of states in the MFIS and the transformation of the input external information are only two interrelated parts of the same information transformation process. These two parts are reflected in the lines of the polygram (10.8) - (10.11).

#### **10.4. The principles of building configured processors and computers that simultaneously process general and local information**

The structure of the reconfigurable system with the memory on the flip-flops is represented in two parts: a constant that processes general information. and a variable that processes local information. The processing of such hierarchical information occurs consistently. First, the general information determines the area of processing of private information in the multifunctional device, and only then the local information is processed.

The structure of the reconfigurable system with the memory on the MUSP allows to process the hierarchical information in parallel, which reduces the processing speed of information.

The development of new principles for building adders, registers, control devices, reconfigured processors and computers on new elementary multi-level memory circuits allows to store both general and local information simultaneously, to increase the processing of information in the adder fourfold and to change the storage structure of local information under the influence of general information, which makes it possible to increase the processing speed of information in the processor.

The theory of design of MUSPs from the MFIS has been considered, which have a vertical connection between their levels [7-9]. On the basis of this vertical connection, the generation of the conserved input signals  $e_j(\Delta)$  is performed in the MUSP, which makes it possible to perform the enlarged transitions.

The hierarchical link between the states of the MFIS in the lower level MUSPs and the subsets of the MFIS states in the upper level MUSPs helps to simultaneously store the general information in the MFIS of the lower layers and local information in the MFIS upper levels [16-18].

The MUSP has a number of advantages in terms of functional and design characteristics compared to multistable triggers.

According to the functional properties, all MFISs are able to work in parallel at each level of the MUSP and to reconfigure (reconfigure) the area of operation of the upper MFISs processing individual (private) information, without reducing the performance of all levels of the MUSP. This allows simultaneous processing of general and private information, which has no analogues in modern processors and computers.

The microprogram control level determines, first of all, the command system, which is subsequently used by the software control level (Figure 10.1).

When using the mili-program control level, together with the microprogram control level, a polyprogramming control level is created that allows the firmware to be changed at the expense of milprograms at the polyprogram level and leads to a change in the command system implemented by the software level in one machine clock cycle  $T$ . Hence, the variety of commands defines the class of functions, which can be most effectively implemented when solving a certain class of problems. A

firmware or a specific system of microprograms is implemented in one block  $\pi_i$  of states of control device circuits.

When constructing parallel hierarchical structures, the control devices of the polyprocessors on the elements of the MFIS and the MUSP can be influenced by the multimanders (input signals  $e(\Delta)$ ), changing the composition of the microprograms, reconstructing the computer command system, and simultaneously orienting its work to more efficient data processing, local and general information.

Such a structural scheme can be used in self-improving algorithms, in which the learning algorithm under the influence of the third level of control changes not only the numerical parameters, but also the structural scheme of the fourth level (working) algorithm.

It can be imagined that the instruction register in the computer consists of an MFIS that is able to rebuild a lot of its states (comandas) under the control of the supervisory register. At the same time, the register on the MFIS effectively implements the working algorithms of the command system for processing special algorithms when solving only one particular class of problems, and in the case of a different class of tasks, the register on the MFIS can be tuned to efficiently handle special algorithms when solving a different class of problems. Thus, at a mili-software control level, the computer can work as computers with different command systems that more efficiently process information from different classes of tasks.

The speed  $V$  of solving problems while simultaneously processing general and local information at the polygram control level can be calculated in the first approximation by the formula:

$$V = \frac{1}{k_1 t_1}, \quad (10.12)$$

where  $t_1$  is the time for retrieving a word from memory when performing an operation in a separate device and simultaneously sampling the word to rebuild the processing algorithm;

$k_1$  - the average number of calls to the memory device during the operation.

With decreasing accesses to memory due to the implementation of multicomputers, the speed of solving self-improving algorithms for word samples is increased by eliminating additional memory accesses for changing algorithms that are implemented on modern reconfigured computers.

In addition, new processing capabilities appear in computers, which in principle can not be used at present, since the memory of registers is based on binary triggers. With mili-program management, you can use the new consolidated transitions, which expand the capabilities of computers, computer systems and networks. This allows you to create reconfigured devices, computers, computer systems and networks that can rebuild the algorithms of their work, depending on the needs of the control object at a higher speed.

The range of algorithms that are self-developed, in practice very large: pattern recognition, information protection, training systems, etc. [1].

The new direction of building computers on the MFIS and the MUSP promotes the progress of computer technology. It can be implemented on modern logic elements used in VLSI, FPGA, RAM, and can also affect the development of reconfigured devices, computers, computer systems and networks.

### **10.5. Methods for building a reconfigured computer with regard to the "elemental" level**

The block diagram of the processor has two levels of control in the polyprogram that are executed simultaneously in each cycle, in which the controlled device implements the firmware, and the control device implements the millyprograms. In the process of work, instructions by milprogres (general teams) are able to simultaneously change the structure of the execution of micro instruction instructions (individual commands), which allows increasing the speed of execution of hierarchical algorithms and increasing their flexibility in solving problems whose algorithms change in the process of solving them. This possibility appears when you use N-level

memory circuits in the computer's devices as registers of elements, which memorize both general and private information.

For each level of control, you can use an operating device (arithmetic logic unit), as well as additional cache memory on elements of N-level memory circuits.

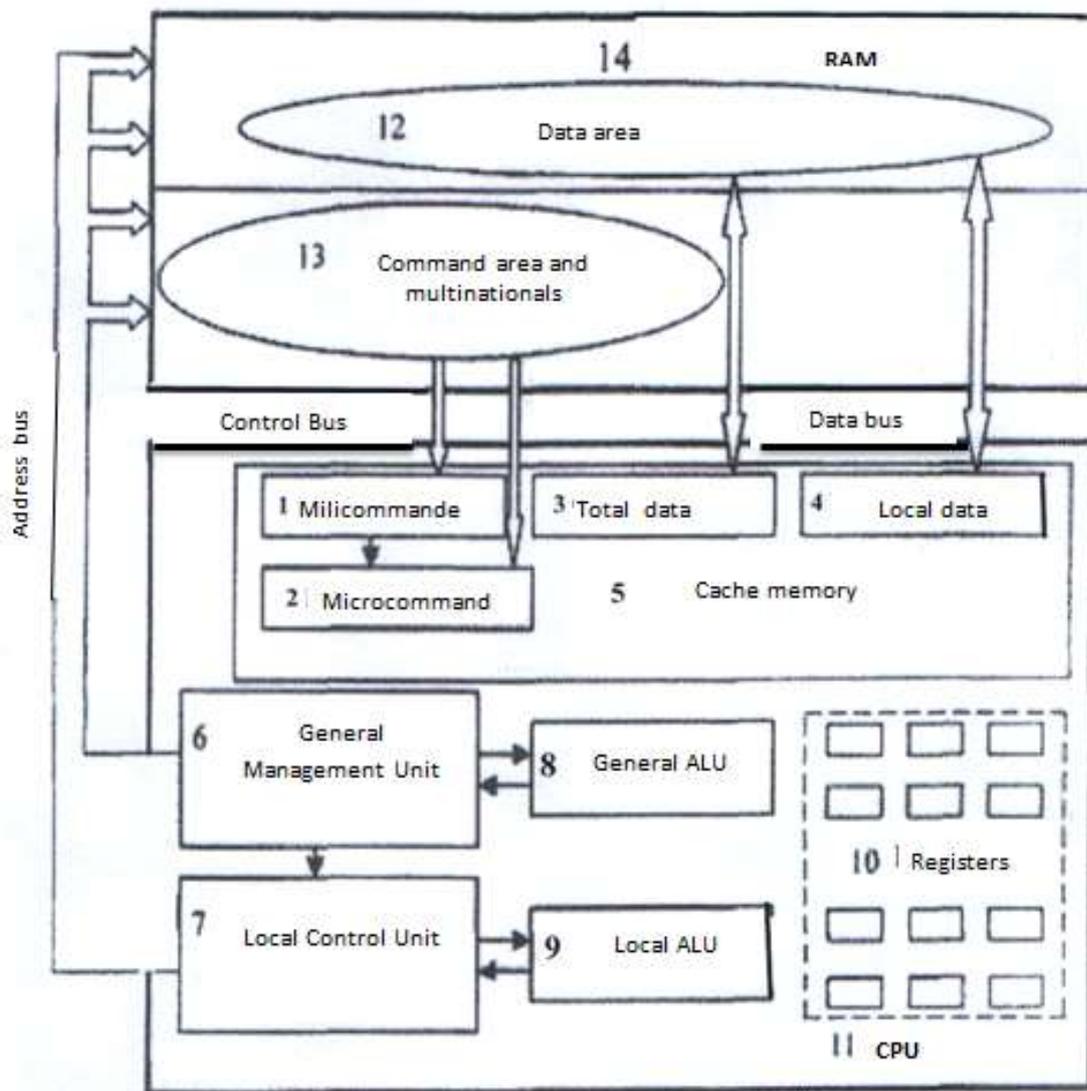
Essential is that the electronic computer consists of two components: the processor and RAM, which are connected by a system bus, consisting of a data bus, a control bus and an address bus [10].

The electronic computer (Figure 10.6) [4] is distinguished by the fact that the processor has N-level memory circuits (for example,  $N = 2$ ) that make up general purpose registers (RON), general arithmetic logic unit (ALU) and a separate ALU, which is associated with the corresponding control units of the general and local level.

The control units are hierarchically linked from the general to the local. The general control unit and the local control unit via the address bus associated with the main memory.

The RAM itself is divided into a data area and the instruction area is respectively connected via a control bus and a data bus with a cache memory having hierarchical registers for the multimand and microinstructions and registers for common data and local data that are respectively associated with the general and local level control units and with blocks of ALU. Cache blocks have feedback on the data bus with the data area.

Functionally, the electronic computer is synchronous. The data and commands are written from the RAM of the registers and cache by the clock signal. The corresponding data for processing is then transmitted via the bus to the ALU. When everything is done, ALU starts to work. After the calculation, the results are transferred to the register data bus. ALU can read and write data to the register for one cycle. Both the data and the poly-commands are processed from the memory cells for processing by the processor.



**Fig. 10.6.** Reconfigured electronic computer

Polymandans are divided into common teams, of which the mili-pro-gram consists, and local teams, of which the microprograms are composed. These two instructions are simultaneously fed to the corresponding levels of N-level memory circuits

(for example,  $N = 2$ ) of the processor control units, which implement hierarchical algorithm changes in one cycle.

One of the main temporal characteristics of processing hierarchical information in this case is faster processing of local information in relation to the general, and one of the functional characteristics of control information is the change in the algorithm for processing local information when certain general control information is processed.

It can be imagined that the instruction register in the computer consists of the MFIs, which rebuild many of their states (commands) under the control of the control register. At the same time, the register on the MFIS effectively implements the working algorithms of the command system for processing special algorithms when solving only one specific class of problems, and in the case of another class of problems, the register on the MFIS has the ability to be tuned to the efficient processing of special algorithms when solving another class of problems.

Thus, at a mili-program control level, a computer can operate as different computers, which collectively more efficiently process information from different classes of tasks.

In addition, new processing capabilities appear in the computer, which can not be used in principle, since the memory of registers is based on binary triggers. With mili-program management, you can use new transitions: enlarged and probabilistic, extending the capabilities of the computer, systems and networks. This allows you to create multifunctional devices, computers, systems and networks that can rebuild the algorithms of their work, depending on the needs of the control object.

General information can also be submitted as a separate (local) and general (control). Such a hierarchical separation of information is finite and possible up to a certain minimum amount of general information.

The main distinguishing feature of the principle of hierarchical program control is that the local control information is divided into blocks (subsets) of states whose operating area is determined by the state of the common control information that generates an input signal for these state blocks (realizing the state conservation function).

## **10.6. Accelerating the execution of algorithms in reconfigured computer systems on multi-level memory circuits**

The possibility for self-improvement can be embedded in any algorithmic system, as Academician V.M. Glushkov [1]. An algorithm that changes over time (depending on the previous input information) its response to certain input words, in modern serial information processing systems is called self-changeable, training or reconfigured.

When algorithm A is some reconfigured algorithm, it determines not one alphabetic operator, but an entire family of such operators. In practice, the reconfigured algorithms are specified in the form of a specially organized system of algorithms. In the simplest case, such a system consists of two algorithms. The first of these algorithms A performs processing of information (conversion of input words to output). This algorithm is called working. The second algorithm B is called the control, training or algorithm of the strategy automaton. Algorithm B affects algorithm A, changing the algorithm of its operation. In modern computers, the influence of algorithm B is carried out after each transformation by the working algorithm A of the next input word to the corresponding output word, as shown in Section 1.

In polycomputers, it is possible to change the sequential order of the system of algorithms B processing general information and A processing separate (partial) information for parallel (simultaneous) operation of algorithms B and A. This possibility is explained by the fact that multicore memory circuits are used in the polycomputer, which simultaneously store the general and local information and are used in the control device (Figure 4.5), in which the general information affects the local information, changing the algorithm of its operation.

In a number of cases it is possible to generalize the concept of a polycomputer that processes only a system consisting of two algorithms B and A, to processing a system of algorithms of different levels. To do this, elementary multi-level memory circuits are used that have more levels than 2. In this case, the algorithm of the first-level strategy machine affects the algorithm of the second-level strategy machine; the algorithm of the second-level strategy automaton influences the algorithm of the third-level strategy machine and introduces some corrections into it, etc. Such a stepped organizational system of memory elements of the polycomputer allows simultaneously processing systems of self-improving algorithms, which allows accelerating the processing of high forms of self-improvement and self-organization in polycomputer systems.

### **10.7. Conclusion to chapter 10**

The principles and methods of structural organization of control devices that allow simultaneous processing of general and local information and give new functionality to these devices are considered in this chapter.

New principles and methods for constructing the architecture and structure of processors with the use of multilevel memory schemes and the construction of polycomputers simultaneously processing general and local information are also considered.

In general, we can conclude that the proposed results of the work give the direction to systematically design the reconfigured devices of computer systems taking into account the automatic memory circuits, which have less hardware costs and greater speed in the reorganization of work algorithms.

## CONCLUSION OF SECTION 3

What is important and fundamental in the approach proposed by Marakhovsky

L.F.:

1. Multifunctional and multilevel [7-9] elementary memory circuits are created and patented, which are not inferior to triggers in terms of speed, and:
  - Have less hardware costs per one memorized state (gain in equipment!);
  - less by an order of internal connections, which is very important in the development of integrated circuits;
  - And most importantly - they are able to change the structure of memorizing states in the process of work and to implement a certain direction of information that triggers in principle can not do;
2. Patented:
  - Electronic computer [4] on multifunctional and multilevel elementary memory circuits;
  - Structural automaton [6], in which the theory of automata of the third kind is protected;
  - Microprogram control device [5], on multifunctional and multilevel elementary memory circuits.

These all reconfigurable devices are able to change the algorithm of their work at the "elemental" level due to the ability of automatic memory circuits to implement their transitions in two variables: setting and storing input signals.

## **SECTION 4**

### **NEURONS AND NEURAL NETWORKS**

#### **Chapter 11.**

#### **Some notions about neurons**

##### **11.1. Introduction**

Currently, artificial intelligence (AI) is regarded as one of the scientific areas of computer science. Work on artificial intelligence unfolded with the beginning of industrial use of computers [1 -2].

Formally, in the 70-80's. the last century it was finally established that the level of development of computer technology does not allow us to speak even of the possibility of approaching the Artificial Intelligence. "Reasonableness" of automatic systems was removed from consideration [3-9].

During these years, many scientists realized that their path had reached a deadlock, due to ignorance of the fundamental principles of building AI devices.

In November 2011, in Portland, Oregon, Supercomputing Conference'09, IBM announced significant progress in creating a computing system that simulates and emulates the ability of the brain to feel, perceive, act, interact, learn, and at the same time comparable with the brain for low power consumption and size [10-11].

Investigating the failures and difficulties of creating AI at the present stage, the author comes to the conclusion that fundamental research in the field of the human brain, its neuron has not yet led to the creation of an appropriate model of the neuron cell, its model of connections that characterize the real object. In connection with this, supercomputers are used, which, according to external characteristics, create a model of "cat action".

In the fourth section, an attempt is made to create a model of a digital artificial neuron, implemented on the basis of the MFIS and MUSP, which can form the basis for a new direction in the development of systems with increased intelligence.

## **11.2. Preliminary concepts of the neuron model**

The profound study of any of the sciences, and even more so of the sciences, directly or indirectly related to the human brain (medicine, psychology, cybernetics, computer technology, etc.), leads to the need to study the processes of human thought. The world within a person can be compared to the world of the universe that surrounds a person by the complexity of the study. From time immemorial and up to now the structure of the neuron cell, the structure of the human brain and its possibilities is an important problem of philosophy, religion and psychology, and is actively discussed in many scientific treatises, often with irreconcilability in judgments.

At present, in the field of computers, an acute problem has arisen in the study of all aspects of the human brain in connection with the need to create higher-quality computer systems of artificial intelligence that should approximate the possibilities of thinking of the human brain. At the same time, a number of scientists are skeptical about the possibilities of creating artificial intelligence systems equal to the human brain [12].

In the era of universal intellectualization, when computer systems will talk to a person in his language, remind, warn and explain possible options for pressing problems, methods and interactive software tools that correspond to the possibilities of human thinking are of great interest. Intellectual systems of computer mentoring (Intelligence Tutor Systems), taking into account the type and level of intelligence of the user, must perceive actions and predict the consequences of the realization of conscious and unconscious goals and motives of human behavior.

In this connection, the relevance and interest in the work of the memory of the human brain and its main element - the neuron from the position of multifunctional and multilevel memory schemes - becomes clear [13-15].

### **11.3. Properties of human thinking**

In the human brain, in addition to the rational and irrational numbers of unambiguous logical thinking, there is a probabilistic, associative and fuzzy thinking. Professor L. Zade argues that the theory of fuzzy sets is more acceptable for human thinking in many cases of making a decision, and especially for complex, multivariate solutions [16-18]. The theory of fuzzy sets is, in fact, a step towards the convergence of the precision of classical mathematics and the pervasive inaccuracy of the real world, to the rapprochement generated by the continuing human striving for a better understanding of the processes of thinking and cognition [17]. Essentially L. Zadeh and his followers agree that the logic of human thinking is based not only on the classical two-valued or even many-valued logic, but also on logic with fuzzy truth values, with fuzzy bundles and fuzzy rules of inference. The theory of fuzzy subsets allows structuring hierarchical structures that are separated by not very precise boundaries. For example, when studying thoughts, languages and perceptions in people.

### **11.4. Fundamentals of short-term memory of the human brain**

In his works, L.A. Khursin [12] reflected the characteristics of the human brain as a product of the functioning of a social system, the main structural element of which is a person who is able to adapt to the influence of the external world and to implement expedient actions within the inherent consciousness and will. This is primarily the process of labor activity, which is by its nature an information process.

Mediated by human labor, the information he called associated information, due to the fact that it is accumulated in the human brain through his training. He described the information with free information in connection with the fact that it is synthesized by the human brain when perceiving the reaction of the external environment to the actions performed by man.

The primary flow of related information ensures the life and survival of each person as an element of the social system, and the secondary flow of related information increases the efficiency of the primary information flow and ensures the survival of the system as a whole.

The state of the system, in which the information flow it creates contains in equal quantities functional and structural information, was called the information equilibrium state of the system.

Considering the system of social type and short-term memory of the human brain, Khursin introduces many definitions and formulas, some of which we will describe for further understanding of our studies.

A system of  $N$  elements in the state of information equilibrium has a structure consisting of  $n$  hierarchical levels. The amount of information that is created by the elements of each hierarchical level of the system is a constant value. The amount of information that each element of the system contributes to the information flow is called the information capacity of the system element. In the hierarchical structure of the system, all its elements are ordered from the highest (private) to the lowest (general) level in order of decreasing their information capacity.

In the field of psychology, it has been established that the possibilities of perceiving the absolute difference in stimuli and the speed of motor reactions are limited by the amount of information transmitted. It was determined that the number of symbols or "pieces" of information is constant and equal to "seven plus or minus two". Khursin showed that the number of hierarchical levels of the information structure of the operational memory of the human brain as a number is 7.6, that is, it ranges from 7 to 8.

The product of human thought activity is the flow of free information, representing a variety of interrelated images, various information complexity. The information characteristic of the human brain determines the upper limit of the number of levels of complexity of images that a person can perceive and which can operate in the process of thinking. The upper limit of the amount of information carried by images of each difficulty level is approximately 735.1 nits / element. This constant sets the limits between which a person is still able to establish connections. This restriction, in our opinion, determines the assertion of Gödel's theorem on the incompleteness of deductive systems [97], and also characterizes the properties of short-term memory of the human brain.

From this limitation follows the consequence that the information flow in the RAM can not exceed the value 5586.76 nits / element.

The structure of any system is formed from a set of elements and the links between them. In the process of thinking, the links between the elements are established on the basis of a set of attributes (properties) that characterize the elements of the system. The number of signs that a person can operate in the process of perception and thinking is 54 nits / element. This number is divided into an equal number of structural and functional characteristics, the number of which is 27 nits / element. These results were confirmed in the study of the social structure of the working class: "Out of 54 signs, 27 characteristics were selected in the questionnaire, which were considered to be the most significant for the classification of individuals ...".

These constants are usually called the "mental abilities" of a person. "A person can not develop his mental abilities the way he can develop muscles. The only thing that he can do is to perfect his art in applying mental abilities. This is an important difference that a layman does not distinguish. " The essence of this improvement is the acquisition of information of certain levels of complexity and mastering the methods of its logical processing.

These constants are usually called the "mental abilities" of a person. "A person can not develop his mental abilities the way he can develop muscles. The only thing that he can do is to perfect his art in applying mental abilities. This is an important

difference that a layman does not distinguish. " The essence of this improvement is the acquisition of information of certain levels of complexity and mastering the methods of its logical processing.

The total number of images constituting the information flow, able to access the operative (short-term) memory of the human brain is approximately determined up to 1121, and the information capacity of the threshold for both functional and structural information is determined approximately 27. In the alphabets of most developed modern languages (including the section between the letters ) is the number 27.

The knowledge of the world by man is limited within arbitrarily intersecting areas not only by the volume of short-term memory, but also by the volume of long-term memory.

### **11.5. The problems of creating a model of the human brain**

The human brain has a number of advantages over all technical and cybernetic devices for such important properties.

First, input signals coming from the external environment affect the eyes, ears, body, taste of food, etc.

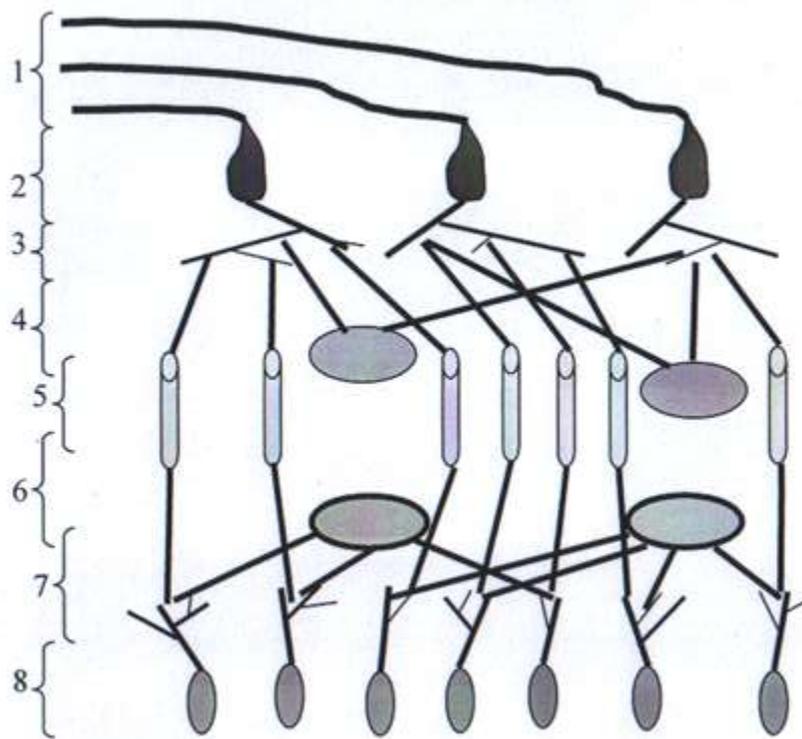
As the Bible says - "In the beginning there was a word", which sets up the brain, as well as other environmental influences.

In this regard, to create a model of the human brain, appropriate sensors should be developed and preferably their output signals are best represented in digital form, for better interfacing with digital devices.

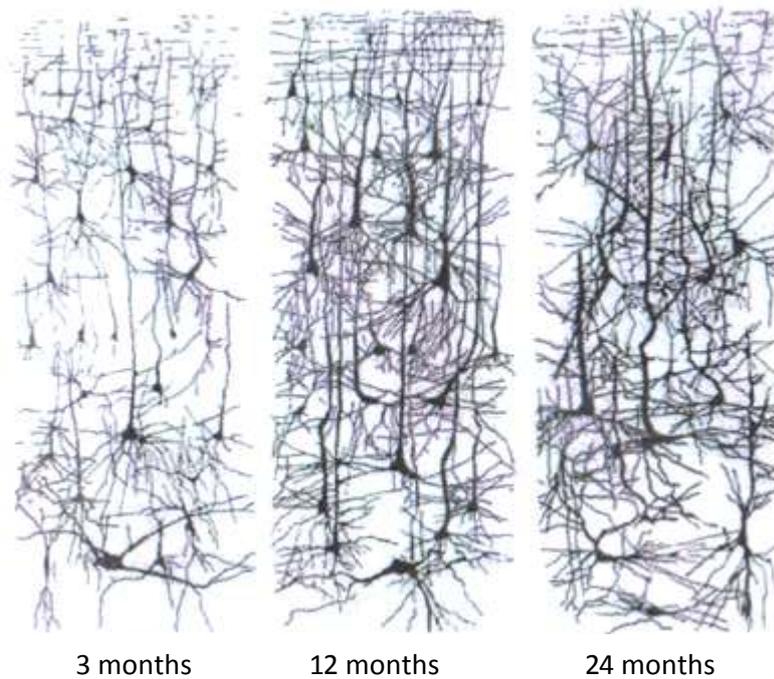
Secondly, information that comes from the external environment is generalized. This is evident from the example of the eye. Receptors in the eye are of the order of 18-20 million, and the cones that generalize the visible information through the eye receptors are of the order of 72,000. That is, the information is compressed approximately 256 times already at the second level (Figure 11.1). This problem of information compression is important to understand and solve technically.

In Fig. 11.1 shows the biological scheme of horizontal organization of information compression and, what is important, shows 8 levels, which corresponds to the levels of the human brain [12]. Thus, the problem is to create control devices that have at least 8 levels, and at each level the information should be generalized at least 256 times.

Thirdly, it is necessary to consider the natural expanding connection between neurons of the human brain, which is depicted in Fig. 11.2.



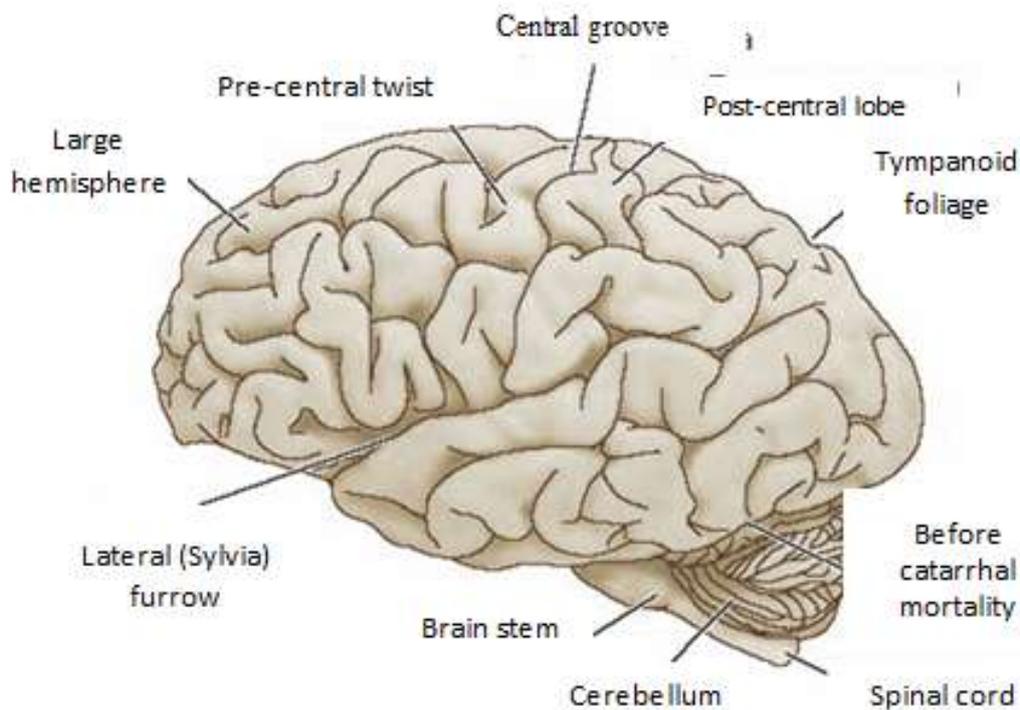
**Fig. 11.1.** Biological scheme of horizontal organization of information compression



**Fig. 11.2.** The process of development of the human cerebral cortex

This image shows how the child's connections gradually begin to be established from three months to two years, necessary to generalize the information obtained, to build the appropriate templates and models that reflect the real world of the individual. Thus, the problem is to create opportunities to expand the links between artificial neurons with increasing information or its generalization.

Fourth, the human brain, possessing from 14 to 20 billion neurons. This is a fairly large structure by the number of neurons, which is difficult to create physically at the present stage of the development of technology, and even more so to manage it (Figure 11.3).



**Fig. 11.3.** The human brain

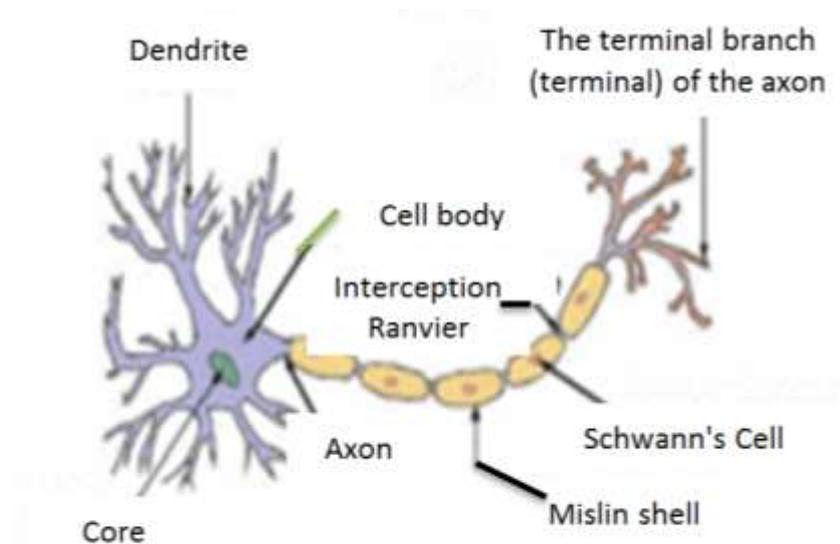
The talented mathematician Frank Plumpton Ramsay proved that complete disorder is impossible in such large structures as the human brain, the universe, etc. Every sufficiently large set of numbers, points or objects necessarily contains an ordered structure. Work in this direction has confirmed this important result [19]. However, the problem of creating ordered structures in the model of the human brain remains.

And fifth, the brain does not have a computer, logical theories, or postal system of numbers, but only its own logic for obtaining information, compressing information, choosing the path of communication with other cells, and generalizing this information. Calculations, reasoning, number systems and any other algorithms are derivatives of those models that have already been generalized and are of interest to a person, as Nikitin NA writes. in an interesting paper, "The Logic of Cell Control" [20].

## 11.6. Information Characteristics of the Human Brain Neuron

The material, taken from Wikipedia, the free encyclopedia, describes the characteristics of the neuron of the human brain as follows.

Neurons (from the Greek *neuron*) are nerve cells, structural and functional units of the nervous system (Figure 11.4). The neuron consists of a body and outgrowths from them - relatively short dendrites and a long axon. Neurons conduct nerve impulses from receptors to the central nervous system and from the central nervous system to the executive organs. Neurons interact with each other and with cells of the executive organs through synapses. Some synapses cause neuron depolarization, others - hyperpolarization; the former are inhibitory. Usually, excitation of the neuron requires stimulation from several exciting synapses.



**Fig. 11.4.** The typical structure of a neuron

Dendrites - as a rule, short and highly branched processes, serving as the main site of formation of excitatory and inhibitory synapses affecting the neuron (different neurons have a different ratio of the length of the axon and dendrites). A neuron can have several dendrites and usually only one axon. One neuron can have connections with 20,000 other neurons.

The place of generation of excitation in most neurons is the axon hillock - the formation at the point of the axon from the body. For all neurons this zone is called trigger.

Depending on the intensity of the functional load, neurons form one or another type of fiber.

The difference between a biological and an artificial neuron is as follows.

Neural networks built on artificial neurons show some signs that allow us to make an assumption about the similarity of their structure to the structure of the brain of living organisms. Nevertheless, even at the lowest level of artificial neurons, there are significant differences. For example, an artificial neuron is an inertial system, that is, a signal at the output appears simultaneously with the appearance of signals at the input, which is completely uncharacteristic for a biological neuron having a memory.

From the considered information characteristics of the neuron of the brain of living organisms, we distinguish such:

1. input signals arriving at the neuron, consist of two types: exciting and inhibitory;
2. The neuron remembers the information;
3. Output signals of the neuron, depending on the intensity of the functional load, form one or another type of fiber, i.e. has a certain functional direction.

Based on these information characteristics of the neuron, in the future we will consider our proposed digital artificial equivalent.

## **11.7. The principles of building an artificial intelligence system**

Full Member of the IAPSP, IEEB V. Govorov in his work "The Paradigm of the New Science of Russia" wrote about the universe as a system, the following.

"Any Creator, when creating the System, uses the basic principles (canons) of its construction. First of all, the system is heterogeneous - the principle of diversity is used, but these heterogeneities are connected by a consistent interaction. The system is multifunctional - and each part of it is able to perform not only its functions, but also duplicate the functions of other parts of the system. This particularly applies to the management functions of the System - it is necessary to duplicate control commands, have a separate "feedback", independent control over the execution of control commands. The presence of a single System Management Center is fraught with the consequences of its failure - another backup center is needed. The system is obliged to ensure its sustainable development, self-learning, resilience, have the ability to adequately respond to external influences, possess the necessary "safety margin", or "unsinkability" of the System. In the event of the destruction of a part of the System, it is mandatory to restore (regenerate) it, and, as a last resort, if the entire System is destroyed, its disaster recovery (Phoenix effect), which requires a separate Emergency Recovery Center (Phantom).

All these conditions are feasible if:

- - Basic and Reserve mathematical systems;
- - Basic and Reserve programming and management languages;
- - Complete and duplicated interrelationships between Blocks of the System with the system of recognition "Svo-Alien";
- - Independent Centers for Analysis and Forecasting.

And one more Main Condition - all this should be described by the Unified Language, which does not allow a double interpretation. "

In this work V. Govorov gave quantitative characteristics of the linguistic Slavic system of language, which already now has more than 700 Bukov. This number of

letters is in the limit of the information characteristic of the short-term memory of the human brain, which is equal to 735 elements.

In addition, it is mentioned that our Language consisted of more than a thousand Letters of Slavic Praalphabet, which corresponds to the maximum area of perception of short-term memory of the human brain, which is limited to a number of elements equal to 1121 elements.

The quantitative characteristics described by V. Govorov are interesting, that the Pantheons of the Orthodox Gods - the 9 Pantheons with 108 Gods - were created to manage the System of the Universe.

### **11.8. IBM's achievements in building a supercomputer on artificial neurons**

IBM specialists have developed an advanced processor that mimics the work of the human brain. "This is our first cognitive computer core, which combines calculations in the form of neurons, memory in the form of synapses, and communications - in the form of axons," the head of the research of Dharmendra Modhu told CNET.

The "biological" processor is based on the BlueMatter algorithm, which was developed in 2009 when trying to identify the relationship between cortical and subcortical structures of the brain. The development of this algorithm was necessary in order to understand how this body processes and exchanges information, noted Modha [21-23].

In article [21] it is pointed out that "modern computations are based on the stored program model, traditionally implemented in digital synchronous serial centralized circuits of general purpose with explicit memory addressing that indiscriminately overwrites the data and creates a boundary between computations and data. In contrast, cognitive calculations such as those performed by the brain will use repetitive computational blocks, neurons and synapses implemented in mixed analog-digital asynchronous parallel distributed reconfigurable specialized and fault-

tolerant biological substrates with implicit memory addressing, which is updated only when information is changed , blurring the boundaries between computation and data. "

According to Modh, such processors can replace the von Neumann architecture, on which most modern computers are built. When the first commercial applications for "biological" chips appear, Modha has not reported. According to analyst Rick Doherty, head of the Envisioning Group, such programs can be ready by 2015 or 2016 [21-23].

To create such chips, IBM combined achievements in nanotechnology, neurobiology and supercomputers. The initial stage of research in the field of neurosynthetic chips was funded by the US Agency for Advanced Scientific Research DARPA, allocating \$ 21 million for the project. The research was conducted within the framework of the scientific project Systems of Neuromorphic Adaptive Plastic Scalable Electronics (SyNAPSE). Now the researchers say that the chips created so far are able to analyze the data, but are not capable of self-restoring, and the memory is not yet in the neurons. These abilities, as the researchers hope, they will appear in the future.

Future applications will impose increased demands on computers and either we will have to significantly increase the computing capabilities of chips or make them more intelligent, "said Dharmendra Modha, the head of this project at IBM Research.

Investigating the failures and difficulties of creating AI at the present stage, the author comes to the conclusion that fundamental research in the field of the human brain, its neuron, has not yet led to the creation of an appropriate model of the neuron cell, its model of connections that characterize the real object. In this regard, and used super-computers (Fig. 2), which by external characteristics create so far only the model of "action of the cat."

In the field of creating a cognitive computer that works similarly to the brain, cortical simulation is an extremely important technology for testing hypotheses about the structure, dynamics and functions of the brain [21].

## 11.9. Conclusion to the 11th chapter

The work of the human brain rests on improving the capabilities of research tools and the very concept of its work. There is indirect evidence that a human neuron is a multifunctional structure capable of reconstructing its work, storing a larger amount of information having two sets of input signals: setting  $x(t)$  - exciting signals and rearranging the structure of subsets of memory  $e(\Delta)$  at which the established state - inhibitory signals, it is possible to communicate with one of the  $18^4$  neurons, which is about 104976 states of memory circuits and much more. Being at a certain level of understanding the functions of the neuron, we are offered our vision of the structure of the neuron, which can partially realize its capabilities. This approach is somewhat different from the generally accepted, but, according to the authors, may have the right to exist to exist.

## **Chapter 12.**

### **ARTIFICIAL ELEMENTARY NEURON**

#### **12.1. Designing an artificial elementary neuron**

##### **12.1.1. IBM created a chip with artificial neurons**

The architecture and behavior of neurons and transistors are fundamentally different. You can create an application that demonstrates behavior similar to neurons, but due to certain basic principles, such an application will be ineffective. However, IBM researchers managed to create a chip that works the same way as a human neural network.

A group of scientists from Cornell University created a fundamentally new chip. He simulates the behavior of neurons. It is based on thousands of cores, capable of carrying out computational operations, they are asynchronous. Each core demonstrates bursts of activity and binds to other cores in a complex way.

As is known, transistors work in binary mode: there is a current, there is no current. The state of one transistor can only change the state of another transistor connected to it. Neurons function differently. They can receive signals from a large number of other neurons. This happens through a system called dendrites. Neurons can also send signals to a large number of other neurons. This happens through axons. Thus, their signals are not binary, like those of transistors. The language in which neurons "talk" is a series of peaks (bursts) of activity that differ in frequency and duration. The processor, created by IBM, simulates the work of the human neural system.

TrueNorth, the so-called new processor, contains 4000 individual cores. To build it, it was necessary to place 5.4 billion transistors on the chip. Each of the nuclei is a set of microcircuits, which in their functions correspond to neurons. The memory of one core is 100,000 bits. This is enough to store the following information: the state of the neuron, the address from which it received the signal, the ad-

dress of the neuron to which it sent the signal. The memory also stores the signal level - everything, like real neurons. Each neuron receives signals from 256 other neurons and transmits them to the same number of neurons.

In the course of experiments, it turned out that TrueNorth surprisingly consumes little energy, despite the fact that it contains a neural network. Researchers say that the chip architecture can be scaled. This means that it was possible to create a super-computer based on a neural network. The next step, which researchers plan to undertake, is to connect TrueNorth chips to the network. As a result, they intend to build a system that will contain hundreds of thousands of nuclei and hundreds of millions of neurons.

This, of course, is a great leap of American scientists and developers in the field of creating machine-based artificial intelligence.

Investigating their achievements, it turns out that the neurons themselves have no memory. The memory is stored in dendrites. They use in their devices automatic discrete time and binary memory. The neurons themselves do not reconfigure. Without detracting from the merits of the TrueNorth processor, it can be noted that it has drawbacks: binary memory, a non-tunable neuron that does not have memory. This platform also caused the use of automatic discrete time [1-2].

Consider an artificial neuron on the MUSP, which is devoid of these shortcomings.

### **12.1.2. Formulation of the problem**

The biological neuron has a number of properties that it is desirable to produce in an artificial neuron. These are the properties:

1. He must store information. In other words, to have memory.
2. Have the property: to rebuild the structure of your memory in the process of work.

3. Have two sets of input signals: setting  $x(t)$  - excitation signals and rearranging the structure of subsets of memory  $e(\Delta)$ , at which the established states - inhibitory signals are remembered.
4. Possess some set of transitions: single-valued, aggregated, probabilistic and fuzzy.
5. Have the opportunity to contact one of the  $18^4$  neurons, which is 104976 states of the memory circuits.
6. Possess the properties of self-control. In other words, in case of catastrophic failures, disconnect yourself. With non-catastrophic failures, narrow down the number of possible connections to other neurons.

The work of the biological neuron has not been fully investigated. There is information that he remembers a large amount of information. For example, the flight of an astronaut to the moon. In addition, in the works of Andrei Viktorovich Nikitin, the idea is expressed that a neuron is a complex biological machine that performs many functions, and not just a transition from one state to another [3-4].

Despite the incomplete understanding of the functions of the biological neuron, we nevertheless undertake to create it at the level of its known properties, which expand the work on neurons of IBM.

### **12.1.3. General principles of designing a single category of an artificial neuron on multi-level memory circuits**

Consider the general principles of designing  $N$ -level  $L_N$  memory devices from the MFIS.

$N$ -level memory circuits ( $N \geq 3$ ) with a multifunctional organization system will be identified with one digit of the neural register into four digits. This MUSP is able to work as an elementary multifunctional automatic machine of Marakhovsky 2nd and 3rd kind. As was discussed earlier, these memory schemes are able to work in deterministic, probabilistic and fuzzy regimes.

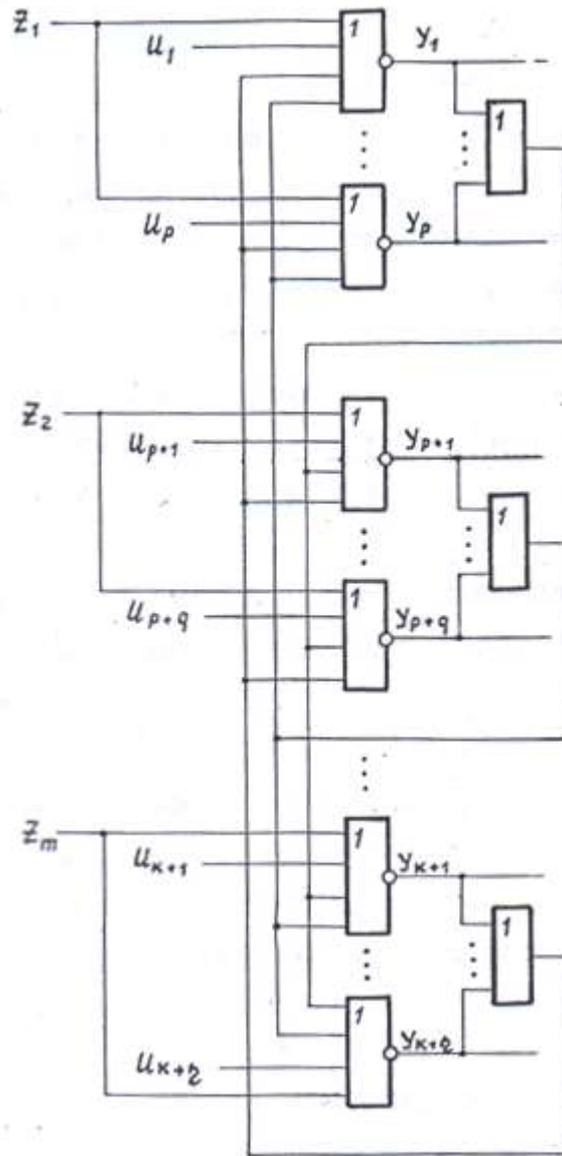
The structure of the  $N$ -level MUSP of the  $L_N$  class includes the controlled MFIS  $A_y^N$  and the  $(N-1)$ -level multifunctional automaton of the strategy  $A_M^{N-1}$ , which consists of the controlled MFIS  $A_y^{N-1}$  and  $(N-2)$ -level multifunction machine strategy  $A_M^{N-2}$ , it.d. until the multifunction machine of strategy  $A_M^1$  becomes one-level.

The free inputs  $n_j$  of the  $BA_j$  (logical elements NOR) of the MFIS  $A_y^j$  ( $j = 2, 3, \dots, N$ ) and the multifunctional automaton of the strategy  $A_M^1$  are connected to the input setting buses  $j$  ( $j = 2, 3, \dots, N$ ).

To reduce the intergroup relations in the MFIS, the outputs of BA only  $i$ -groups, in which  $BA_i > 1$ , are connected to the inputs of the BA of the remaining groups via the logical element OR (Figures 4.3, 4.4 and 12.1).

Each  $(N-1)$ -level automaton of the strategy  $A_M^{N-1}$  equipped with a combination scheme of outputs  $I_{N-1}$ , consisting of  $N$ -input elements AND. Inputs of the elements AND of the automaton  $A_M^{N-1}$  connected to the outputs  $BA_i$  of the  $i$ -th groups, in which the quantity  $BA_i = 1$ , or with the outputs of the OR elements of the  $i$ -th groups, where  $BA_i > 1$  in the MFIS. Each input of the AND element is connected to the output of one of the MFIS  $(N-1)$ -level strategy automaton  $A_M^{N-1}$ , realizing the output function of the automaton  $A_M^{N-1}$ . The outputs of the elements AND of the automaton  $A_M^{N-1}$  are connected in an appropriate way with the free inputs of the  $BA_i$  controlled by the MFIS  $A_y^N$  of those  $i$ -th groups in which the quantity  $BA_i > 1$ .

One input of each element AND of all the  $N-1$  combinational circuits  $I_j$  ( $j = 1, 2, \dots, N-1$ ) of the strategy automata  $A_M^j$  with the control input  $z_0^j$ , which solves the interconnection of the MFIS  $A_y^{j+1}$  with the automaton  $A_M^j$ .



**Fig. 12.1.** MFIS with combined entries of elements in each group

The number  $r_j$  of the elements AND of the scheme  $I_j$  the necessary number  $r_e$  of the input MFIS signals  $A_y^{j+1}$  that preserve  $e(\Delta)$  by the relation:

$$r_j \geq r_e - 1 \quad (12.1)$$

where  $r_j$  is the number of elements of AND circuit  $I_j$ ;

$r_e$  is the number  $e(\Delta)$  of the input signals of the MFIS  $A_y^{j+1}$ .

The same number  $r_j$  of elements of AND is determined by the product of the numbers  $m_j$  of groups of the  $i$ -th MFIS  $A_y^j$ , included in the strategy automaton  $A_M^j$ .

The semi-closed  $N$ -level structure of the MUSP is shown in Fig. 12.2, and the open  $N$ -level structure is shown in Fig. 12.3.

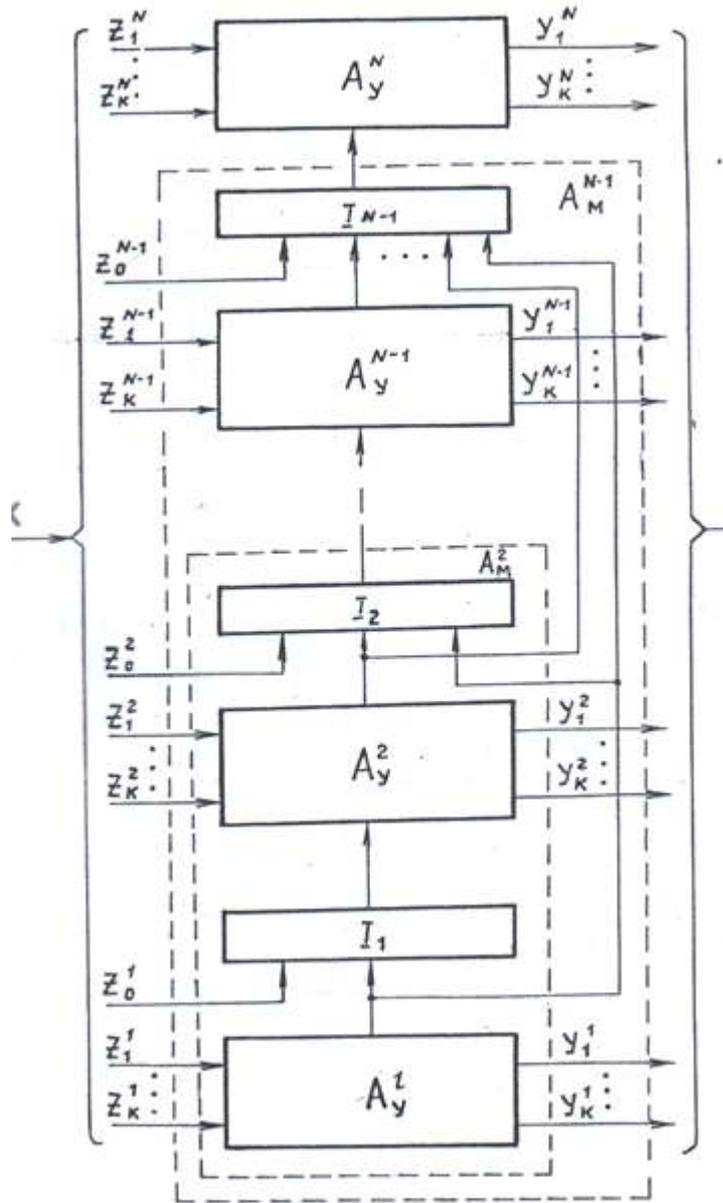
When using the OR elements for realization of connections between groups in the MFIS  $A_y^j$  (Figure 12.1), the minimum signal delay necessary for memorizing the state, respectively, increases from  $2\tau_3$  to  $3\tau_3$ .

The minimum delay required to generate the output signals of the  $A_M^j$  strategy machines, taking into account the combination schemes of the outputs  $I_j$ , also increases correspondingly to  $3\tau_3$ . The structure of the  $N$ -level structure of the MUSP (Figure 12.2) shows that its performance is reduced by the delay of one logical element in comparison with the speed of the  $RS$  flip-flop, which is a small time, commensurate with the spread of the delay of one element.

The number  $M_N$  of the remembered states of the  $N$ -level structure of the MUSP is determined by the product of the numbers  $m_j$  of the BA groups of every  $j$ -th ( $j = 1, 2, \dots, N$ ) MFIS  $A_y^j$ .

$$M_N = \prod_j m_j \quad (12.2)$$

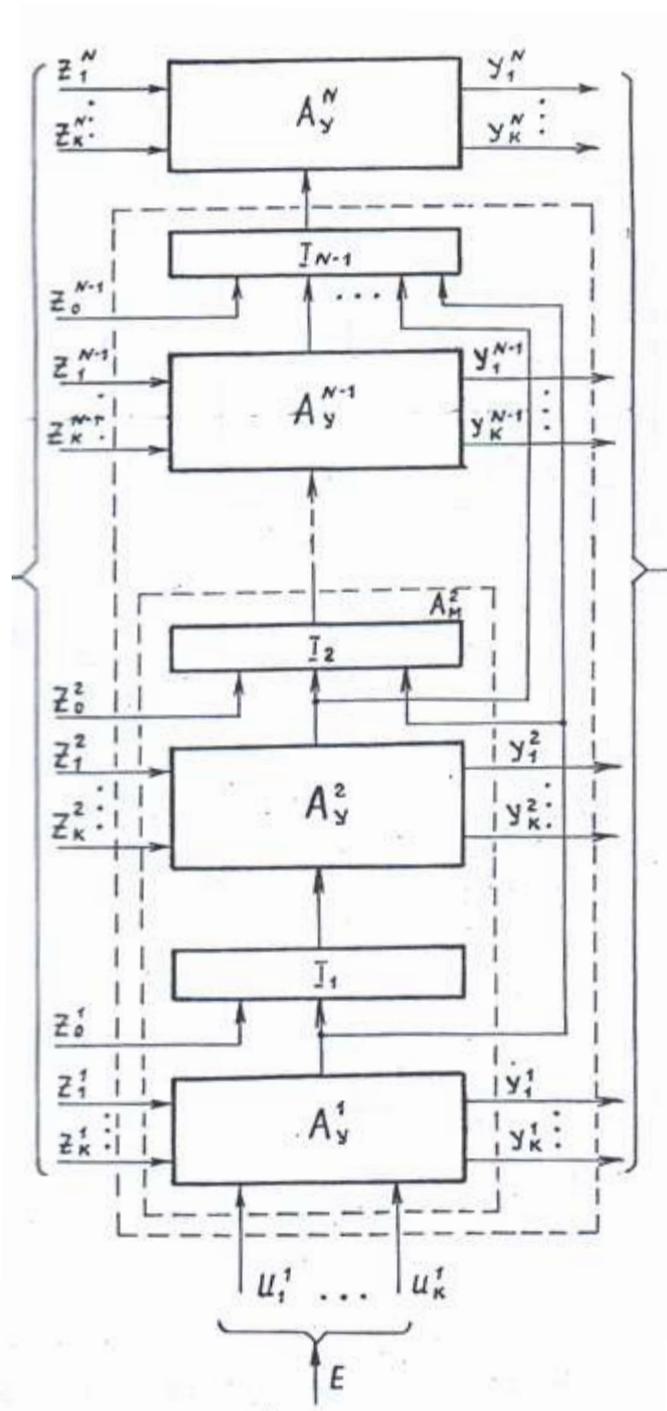
The work of the  $N$ -level structure of the MUSP depends on the values of the control inputs  $z_0^j$ . With the value of all  $z_0^j = 1$ , the  $N$ -level memory operates as a single  $M_N$  stable element in deterministic, probabilistic and fuzzy modes. For all  $z_0^j = 0$ , the  $N$ -level structure of the element is converted to an  $N$ -bit parallel register, each  $j$ -th bit of which is able to store  $m_j$  states.



**Fig. 12.2.** Semi-closed structure of the MUSP

When organizing the structure of  $N$ -level memory, it is possible to reduce the number of interlayer links by using in each combinatorial scheme  $I_j$  the connections from the combination scheme  $I_{j-1}$  (Fig. 12.3). With such an organization, the number of inputs of the AND elements in all combination schemes

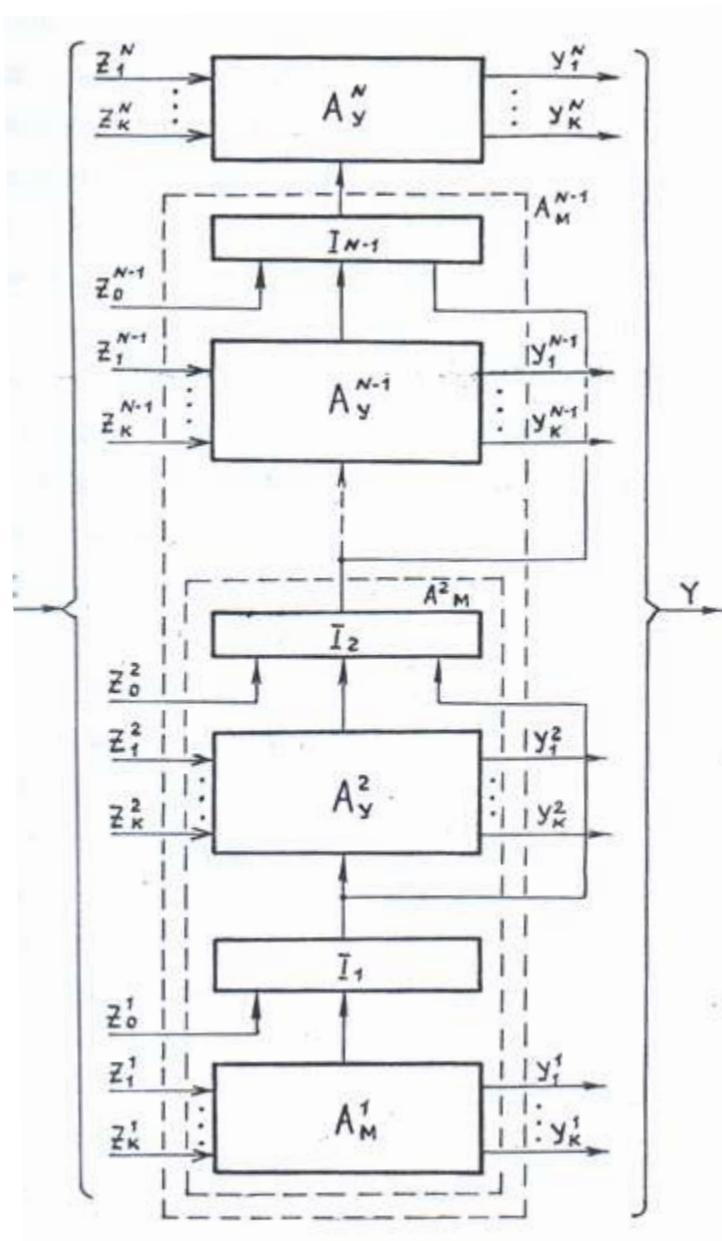
When organizing the structure of  $N$ -level memory, it is possible to reduce the number of interlayer links by using in each combinatorial scheme  $I_j$  the connections from the combination scheme  $I_{j-1}$  (Fig. 12.3).



**Fig. 12.3.** The open structure of the MUSP

With such an organization, the number of inputs of the AND elements in all combination schemes  $I_j$  of the outputs does not exceed three, and the number of elements in each  $j$ -th combination circuit is equal to the number  $M_j$  of the remembered states of the strategy automaton  $A_M^j$ , that one element of  $I$  in each output circuit  $I_j$  is larger, than in the previous  $N$ -level scheme. of the outputs does not exceed three, and

the number of elements in each  $j$ -th combination circuit is equal to the number  $M_j$  of the remembered states of the strategy automaton  $A_M^j$ , that one element of  $I$  in each output circuit  $I_j$  is larger, than in the previous  $N$ -level scheme.



**Fig. 12.4.** Modified structure of the MUSP

In the modified  $N$ -level memory (Figure 12.4), the maximum delay  $\tau_{\text{зад}}$  of the output signal of the strategy automaton  $A_M^{N-1}$  increases and is determined by the formula:

$$\tau_{\text{зад}} = N \tau_0 \quad (12.3)$$

## 12.2. Methods for designing a single category of an artificial neuron on multi-level memory circuits

Consider the scheme of a three-level elementary artificial neuron on the MUSP (Figure 12.5) and its operation.

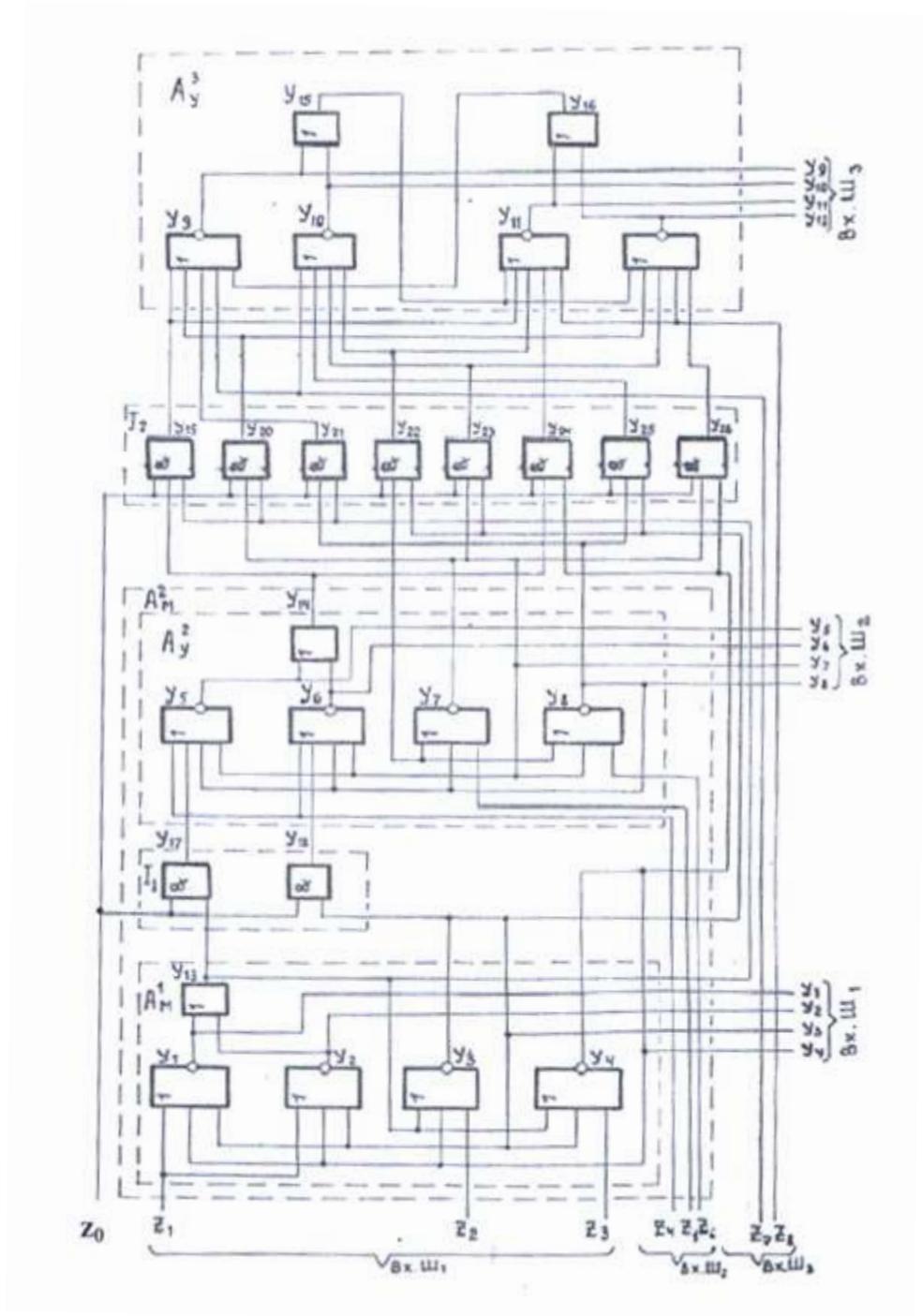


Fig. 12.5. Three-level device of an elementary neuron

Each MFIS  $A^j$  ( $j = 1, 2, 3$ ) consists of four ( $n = 4$ ) logical elements NOR divided into  $m_j$  ( $2 \leq m_j \leq 4$ ) groups. Elements  $A_M^1$  and  $A_Y^2$  are divided into three groups so that in one group there are two logical elements NOR, and in the other two groups one by one. Elements of the MFIS  $A_Y^3$  are divided into two groups of two NOR elements in each. The number  $M_j$  of memorized states in each MFIS  $A^j$  is determined by the formula (4.4) and (4.10):

$$M_1 = M_2 = (2^2 - 1) + (2^1 - 1) + (2^1 - 1) = 5;$$

$$M_3 = (2^2 - 1) + (2^2 - 1) = 6.$$

In those groups where the BA number is greater than 1 ( $R_i > 1$ ) i, the outputs of the BA are interrelated with the inputs of the BA of other groups through the logical element OR, which reduces the intergroup relations in the MFIS  $A^j$ .

The outputs of the BA of those groups where  $R_i = 1$  and the logical elements OR of those groups where  $R_i > 1$ , the MFIS  $A_M^1$  and  $A_Y^2$  go to the logical elements AND of the combination schemes  $I_1$  and  $I_2$  clocked by the control input  $z_0$ .

The numbers  $r_e$  of the conserved input signals in the MFIS  $A_Y^2$  and  $A_Y^3$  are determined by the formula (4.14):

$$r_{e2} = (2^2 - 1) \times (2^1 - 1) \times (2^1 - 1) = 3;$$

$$r_{e3} = (2^2 - 1) \times (2^2 - 1) = 6.$$

Knowing the number of  $r_{ej}$  preserving input signals  $e(\Delta)$  that control the MFIS  $A_Y^j$ , it is possible to determine the necessary number  $r_{yi}$  of logical elements AND of combinational circuits  $I_1$  and  $I_2$  by the formula (5.9).

$$r_{y1} = 3 - 1 = 2; r_{y2} = 9 - 1 = 8.$$

The number  $M_3$  of memorized states of the three-level memory is determined by the formula (5.3):

$$M_3 = 3 \times 3 \times 2 = 18.$$

The outputs of the combination schemes  $I_1$  and  $I_2$  are fed to the inputs of the controlled MFIS  $A_y^j$  of those groups where the number BA is greater than one ( $R_i > 1$ ).

The connections between the outputs of the elements AND of the circuit  $I_1$  and the inputs of the BA MFIS  $A_y^2$  and between the outputs of the elements AND of the circuit  $I_2$  and the inputs of the BA MFIS  $A_y^3$  are determined from the values  $e_j(\Delta)$  of the input signals of the MFIS  $A_y^2$  and  $A_y^3$ . Preserving  $e_j(\Delta)$  input signals are characterized by the fact that at least in two BA groups on the input nodes the value should be equal to one logical zero. The input signals of the MFIS  $A_y^2$  and  $A_y^3$ , which preserve  $e_j(\Delta)$ , are presented in Table. 12.1 and 12.2.

Table 12.1

Preserving  $e_j(\Delta)$  input signals of the MFIS  $A_y^2$

$e_j^2$	The value of the signals $e_j^2$ at the inputs of the elements $y_i$			
	$y_5$	$y_6$	$y_7$	$y_8$
$e_1^2$	1	0	0	0
$e_2^2$	0	1	0	0
$e_3^2$	0	0	0	0

In accordance with Table. 12.1 and 12.2, the outputs of the AND components of the circuits  $I_j$  ( $j = 1, 2$ ) and the inputs of the corresponding MFISs  $A_y^{j+1}$  ( $j = 1, 2$ ) are determined by the unit values  $e_j(\Delta)$  of the input signals of the MFIS  $A_y^{j+1}$ . For example, the input  $y_{17}$  of the AND gate  $I_1$  is connected in accordance with Table. 12.1 with the input of the element  $y_5$ , and the output  $y_{18}$  of the second element AND of the circuit  $I_1$  is connected to the input of the element  $y_6$  of the MFIS  $A_y^2$ .

Thus, in accordance with Table. 12.2 (Figure 12.5) connect the outputs of the elements  $y_9 - y_{12}$  of the MFIS  $A_y^3$ . The  $x_i(t)$  input signals set at the input of only one group have a logical zero value, and at the inputs of all other MFIS groups the values of logical units. The input signals  $x_i(t)$  of the MFIS  $A^j$  are presented in Table. 12.3, 12.4 and 12.5.

Table 12.3

The  $x_i(t)$  input signals of the MFIS  $A_M^1$

Input signals $x_i$	Value of input nodes $z_j$			Value of output nodes $y_j$				States $A_j$
	$z_1$	$z_2$	$z_3$	$y_1$	$y_2$	$y_3$	$y_4$	
$x_1$	0	1	1	1	1	0	0	$A_1$
$x_2$	1	0	1	0	0	1	0	$A_2$
$x_3$	1	1	0	0	0	0	1	$A_3$
$x_4$	1	1	1	0	0	0	0	$A_4$

With the value of the control input  $z_0 = 0$ , the three-level memory operates as a three-digit parallel register. The signals  $x_1, x_2, x_3$  set respectively the memorized states  $A_1, A_2$  and  $A_3$  (Table 12.3), which are stored with one input signal when the values are equal to the logical zero at all input nodes  $z_j$  ( $j = 1, 2, 3$ ) ( $x_1 = x_2 = x_3 = 0$ ). The state  $A_4$ , uniquely determined by the input signal  $x_4$ , is not stored for any input signals. Therefore, in the deterministic mode of operation, the input signal  $x_4$  is forbidden.

Signals  $x_5, x_6, x_7$  set the memorized states  $A_5, A_6$ , and  $A_7$  respectively (Table 12.4), which are stored with the input signal when  $z_4 = z_5 = z_6 = 0$ . The state  $A_8$  set by the input signal  $x_8$  is not stored with other input signals. Therefore, in the deterministic mode, the input signal  $x_8$  is forbidden.

Table 12.4

The  $x_i(t)$  input signals of the MFIS  $A_y^2$ 

Input signals $x_i$	Value of input nodes $z_j$			Value of output nodes $y_j$				States $A_j$
	$z_4$	$z_5$	$z_6$	$y_5$	$y_6$	$y_7$	$y_8$	
$x_5$	0	1	1	1	1	0	0	$A_5$
$x_6$	1	0	1	0	0	1	0	$A_6$
$x_7$	1	1	0	0	0	0	1	$A_7$
$x_8$	1	1	1	0	0	0	0	$A_8$

Input signals  $x_9$  and  $x_{10}$  set respectively the memorized states  $A_9$  and  $A_{10}$  (Table 12.5), stored at the input signal  $z_7 = z_8 = 0$ . The input signal  $x_{11}$  is forbidden because the set state  $A_{11}$  is not stored for any input signal of the MFIS  $A_y^3$ .

Thus, for  $z_0 = 0$ , each MFIS  $A_j$  ( $j = 1, 2, 3$ ) works as an elementary automaton of the second kind in a parallel three-digit register and their work does not depend on each other. The total number of  $M_N$  memorized states of a given register can be determined by formula (6.2):

$$M_N = 3 \times 3 \times 2 = 18.$$

Table 12.5

The  $x_i(t)$  input signals of the MFIS  $A_y^3$ 

Input signals $x_i$	Value of input nodes $z_j$		Value of output nodes $y_j$				States $A_j$
	$z_7$	$z_8$	$y_1$	$y_2$	$y_3$	$y_4$	
$x_9$	0	1	1	1	0	0	$A_9$
$x_{10}$	1	0	0	0	1	1	$A_{10}$
$x_{11}$	1	1	0	0	0	0	$A_{11}$

For  $z_0 = 1$ , the three-level memory is transformed into a combined elementary automaton A in which the operation of the controlled MFIS  $A_y^j$  in certain blocks of

their states depends on the states of the strategy automata  $A_M^{j=1}$  and their output functions realized on the combination circuits of the outputs  $I_1$  and  $I_2$ .

Imagine in Table. 12.3 the MFIS state blocks  $A_y^2$ , which are conserved under well-defined MFIS conditions  $A_M^1$ .

In Table. 12.4 blocks of MFIS states  $A_y^3$ , which are preserved for completely defined combined states of the strategy automaton  $A_M^2$ .

The combined three-level elementary automaton  $A$  is able to function as nine *RS*-triggers (Figure 12.6). When functioning according to the *RS*-trigger algorithm, the MFIS  $A_M^1$  and  $A_y^2$ , which implement the automaton of the strategy  $A_M^2$ , do not change their states. In this case, the trigger states can only change under the influence of the input signals  $x_9$  and  $x_{10}$  arriving at the input nodes  $z_7$  and  $z_8$  of the controlled MFIS  $A_y^3$ .

The states of all nine triggers can be stored with one input signal  $e(\Delta)$  when the values are equal to logical zero at all input nodes  $z_j$  ( $j = 1, 2, \dots, 8$ ). If the state of the strategy automaton  $A_M^1$  is unchanged, the combined three-level automaton is able to function in deterministic mode as a six-stable memory element in the following blocks  $\pi_j$  ( $j = 1,2,3$ ) states:  $\pi_1 (A_{21}-A_{26})$ ,  $\pi_2 (A_{27}-A_{32})$ ,  $\pi_3 (A_{33}-A_{38})$ ,

Table 12.6

The stored combined states of the automaton  $A_M^2$

States MFIS $A_M^1$	Output signals MFIS $A_M^1$				Output signals MFIS $A_y^2$				The combined states of an automaton $A_M^2$
	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$	$y_7$	$y_8$	
$A_1$	1	1	0	0	0	1	0	0	$A_{12}$
	1	1	0	0	0	0	1	0	$A_{13}$
	1	1	0	0	0	0	0	1	$A_{14}$
$A_2$	0	0	1	0	1	0	0	0	$A_{15}$
	0	0	1	0	0	0	1	0	$A_{16}$
	0	0	1	0	0	0	0	1	$A_{17}$

$A_3$	0	0	0	1	1	1	0	0	$A_{18}$
	0	0	0	1	0	0	1	0	$A_{19}$
	0	0	0	1	0	0	0	1	$A_{20}$

Transitions from one state to another in each of the blocks  $\pi_1 (A_{21} - A_{26})$ ,  $\pi_2 (A_{27} - A_{32})$ ,  $\pi_3 (A_{33} - A_{38})$ . They can occur only under the action of the input signals  $x_i(t)$  arriving at the input nodes  $z_j (j = 4-8)$  of the controlled MFIS  $A_y^2$  and  $A_y^3$  (Fig. 12.7 - 12.9).

Table 12.7

The combined states of a three-level automaton A

Combined state of the automaton $A_M^2$	Output signals MFIS $A_M^1$				Output signals MFIS $A_y^2$				Output signals MFIS $A_y^3$				Combined state of the automaton A
	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$	$y_7$	$y_8$	$y_9$	$y_{10}$	$y_{11}$	$y_{12}$	
$A_{12}$	1	1	0	0	0	1	0	0	0	1	0	0	$A_{21}$
	1	1	0	0	0	1	0	0	0	0	0	1	$A_{22}$
$A_{13}$	1	1	0	0	1	0	1	0	0	1	0	0	$A_{23}$
	1	1	0	0	0	0	1	0	0	0	1	0	$A_{24}$
$A_{14}$	1	1	0	0	0	0	0	1	0	1	0	0	$A_{25}$
	1	1	0	0	0	0	0	1	0	0	1	1	$A_{26}$
$A_{15}$	0	0	1	0	1	0	0	0	1	0	0	0	$A_{27}$
	0	0	1	0	1	0	0	0	0	0	1	0	$A_{28}$
$A_{16}$	0	0	1	0	0	0	1	0	1	0	0	0	$A_{29}$
	0	0	1	0	0	0	1	0	0	0	1	0	$A_{30}$
$A_7$	0	0	1	0	0	0	0	1	1	0	0	0	$A_{31}$
	0	0	1	0	0	0	0	1	0	0	1	1	$A_{32}$
$A_{18}$	0	0	0	1	1	1	0	0	1	1	0	0	$A_{33}$
	0	0	0	1	1	1	0	0	0	0	0	1	$A_{34}$
$A_{19}$	0	0	0	1	0	0	1	0	1	1	0	0	$A_{35}$
	0	0	0	1	0	0	1	0	0	0	1	0	$A_{36}$

$A_{20}$	0	0	0	1	0	0	0	1	1	1	0	0	$A_{37}$
	0	0	0	1	0	0	0	1	0	0	1	1	$A_{38}$

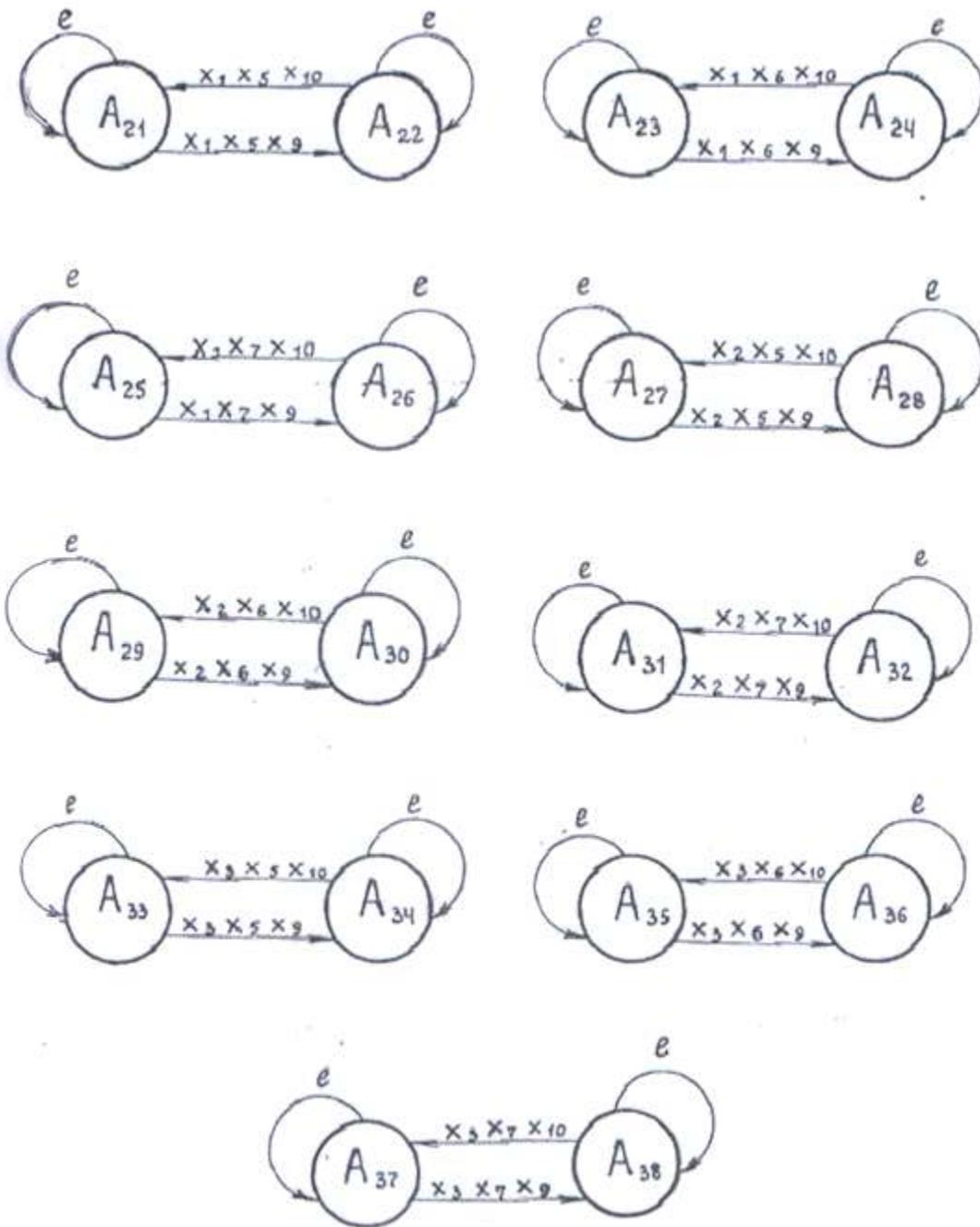
Setting the input signal  $x_1$ ,  $x_2$  or  $x_3$  of one of the three states of the automaton  $A_M^1$ . The automaton  $A$  is able to function in certain blocks  $\pi_j$  ( $j = 1, 2, 3$ ) of states under the influence of the signals  $x_5, x_6, x_7, x_9$  and  $x_{10}$ .

Under the simultaneous appearance of the corresponding triads from the input signals  $x_1, x_2, x_3, x_5, x_6, x_7, x_9$  and  $x_{10}$ , the automaton is able to function as an 18-digit memory element in the whole set of its states  $A_{21}$ - $A_{38}$ . For example, when the input signals  $x_1, x_5$  and  $x_{10}$  appear at the input nodes  $z_j$  ( $j = 1, 2, \dots, 8$ ), the automaton passes to the state  $A_{21}$ , and when  $x_3, x_7$  and  $x_9$  appear, the state  $A_{38}$ , etc.

Thus, the three-level memory device in these deterministic modes functions as an elementary automaton of the second kind and is capable of operating as nine  $RS$ -triggers (Figure 12.6), as three six-element elements (Figure 12.7-12.9) or as one 18-element element memory, using all of its 18 states.

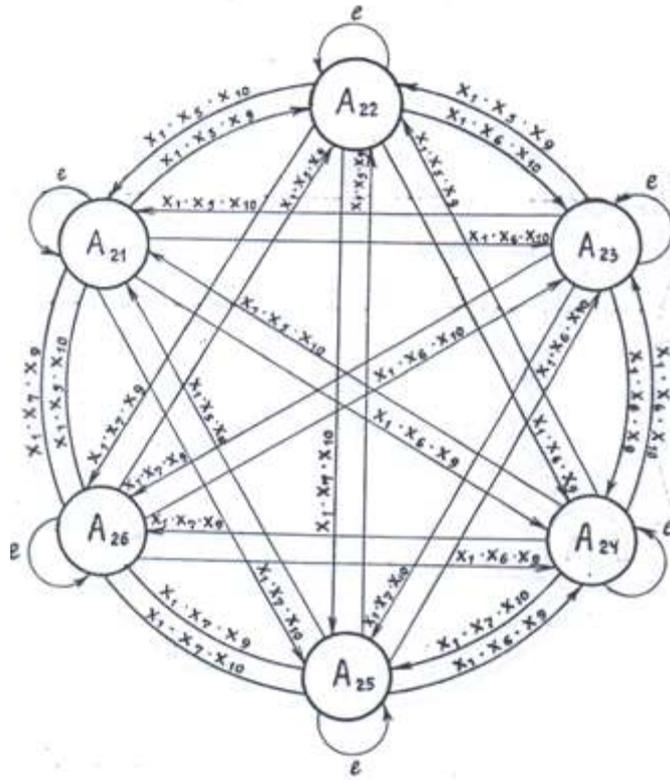
The three-level memory device is able to implement large-scale transitions in the schemes of controlled MFIS  $A_v^j$  with changes in the state of the strategy, under the influence of the input signals of the strategy automata  $A_M^{j-1}$ . that determine the  $x_M(t)$ .

When states change in the two-level strategy automaton  $A_M^2$ , the three-level device can function in two of its blocks  $\mu_i$  ( $i = 1, 2$ ) states: in the block  $\mu_1$  containing nine states  $A_{21}, A_{23}, A_{25}, A_{27}, A_{29}, A_{31}, A_{33}, A_{35}, A_{37}$  and in block  $\mu_2$  containing nine such states  $A_{22}, A_{24}, A_{26}, A_{28}, A_{30}, A_{32}, A_{34}, A_{36}, A_{38}$ .

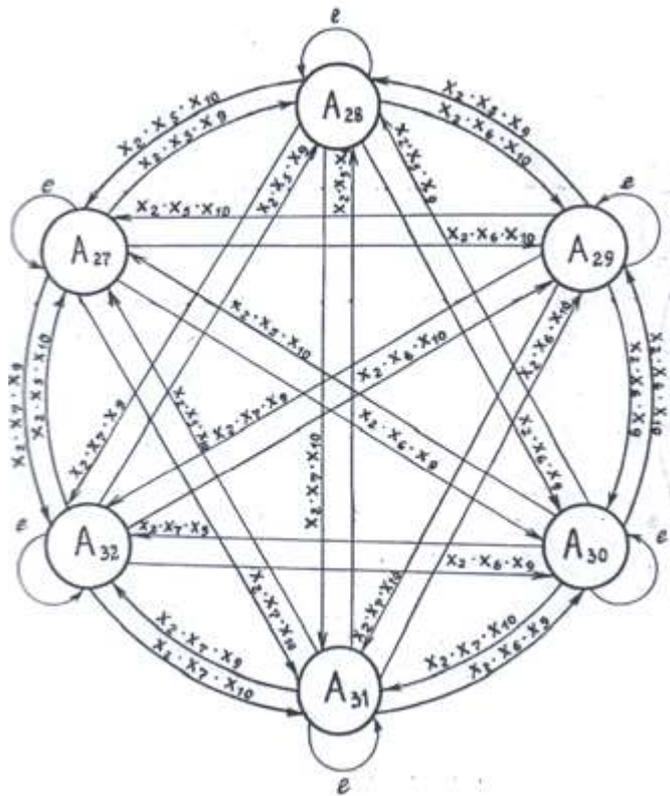


**Fig. 12.6.** The law of operation of a three-level memory device as nine RS flip-flops

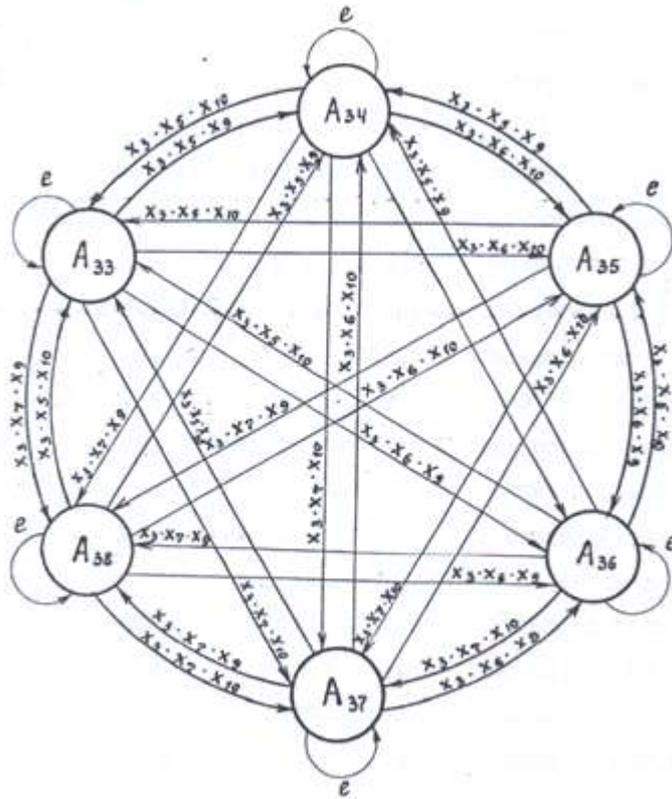
When the states in the strategy machine  $A_M^1$  change state, the three-level memory device is able to function in three blocks  $\mu_i$  ( $i = 1, 2, 3$ ), respectively containing six states  $\mu_1 \{A_{21}-A_{26}\}$ ;  $\mu_2 \{A_{27}-A_{32}\}$ ;  $\mu_3 \{A_{33}-A_{38}\}$ . Enlarged transitions, dependent on the input signals  $x(t)$  and  $e(\Delta)$ , are presented in Table 12.6.



**Fig.12.7.** The law of operation of a three-level memory device, as a six-stable element in the  $e_1$  state block



**Fig.12.8.** The law of operation of a three-level memory device as a six-stable element in the  $e_2$  state block



**Fig.12.9.** The law of operation of a three-level memory device as a six-stable element in the  $e_3$  state block

Thus, during the enlarged transitions in deterministic mode, the three-level memory device functions as an elementary automaton of the third kind [5].

The transitions in all deterministic regimes of the three-level automaton  $A$  occur under the influence of elementary single-valued words  $p_0(T)$  consisting of the input signals  $x_i(t)$ , which uniquely establish the memorized states in the MFIS  $A_M^1$ ,  $A_Y^2$  and  $A_Y^3$ , and one input of the signal  $e(\Delta)$ , that is,  $p_0(T) = x_i(t), e(\Delta)$ .

Table 12.8

Enlarged transitions of a three-level memory device

$x_i$	$A_2$	$A_3$																
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8
$x_1$	$A_2$																	
$x_5$	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
$x_6$	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
$x_7$	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
$x_2$	$A_2$																	

$x_5$	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
$x_2$	$A_2$	$A_3$																
$x_6$	9	0	9	0	9	0	9	0	9	0	9	0	9	0	9	0	9	0
$x_2$	$A_3$																	
$x_7$	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
$x_3$	$A_3$																	
$x_5$	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
$x_3$	$A_3$																	
$x_6$	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
$x_3$	$A_3$																	
$x_7$	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
$x_1$	$A_2$																	
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
$x_2$	$A_2$	$A_2$	$A_2$	$A_3$	$A_3$	$A_3$	$A_2$	$A_2$	$A_2$	$A_3$	$A_3$	$A_3$	$A_2$	$A_2$	$A_2$	$A_3$	$A_3$	$A_3$
	7	8	9	0	1	2	7	8	9	0	1	2	7	8	9	0	1	2
$x_3$	$A_3$																	
	3	4	5	6	7	8	3	4	5	6	7	8	3	4	5	6	7	8

The three-level automaton  $A$ , like the two-level one, is able to function in probabilistic and fuzzy regimes [5-7].

The automaton  $A$  operates in a probability mode under the influence of one of the probabilistic words  $p^B(T)$ , consisting of pairs of input signals  $x_1, x_2, x_3, x_5, x_6, x_7$  of the strategy automaton  $A_M^2$ , establishing the corresponding states  $A_j$  (Tables 12.3 and 12.4), and the input signal  $x_{11}$  of the controlled MFIS  $A_y^3$ , which establishes the unmemorable state of  $A_{11}$  (Table 12.5), as well as the second input signal  $e(\Delta)$ .

The probability transitions in the  $\pi_j$  state block are presented in Table 12.7. The probabilistic transition occurs in one of two states of well-defined blocks  $\pi_j$  ( $j = 4, 5, \dots, 12$ ). The theoretical probability of the transition of  $P_e$  to one of the states  $\pi_j$  is 0.5. In practice, it ranges from 0 to 1 ( $1 \geq P_e \geq 0$ ). In the study of the probability transition, it usually goes over to one of the states at  $P_e \approx 0.3$ , and to another state - at  $P_e \approx 0.7$ .

Thus, probability transitions provide an integrated transition from a certain state  $a_i$  to one of the states of a certain block  $\pi_j$  ( $j = 4, 5, \dots, 12$ ).

In the three-level automaton  $A$  considered, matrix (fuzzy) transitions are presented in Table 12.8. Fuzzy transitions provide transitions from an arbitrary state  $A_j$

to the state  $A_h$  of the matrix block  $e_i(\Delta)$  consisting of completely defined blocks  $\pi_i$ , defined by the two-level automaton of the strategy  $A_M^2$  with a certain measure of the probability  $P_h$ .

Table 12.9

Probabilistic transitions of a three-level memory device

Input signals $x_i(t)$	Transitions to one of the states of the automaton A	State Blocks $\pi_j$
$x_1x_5x_{11}$	$A_{21}, A_{22}$	$\pi_4$
$x_1x_6x_{11}$	$A_{23}, A_{24}$	$\pi_5$
$x_1x_7x_{11}$	$A_{25}, A_{26}$	$\pi_6$
$x_2x_5x_{11}$	$A_{27}, A_{28}$	$\pi_7$
$x_2x_6x_{11}$	$A_{29}, A_{30}$	$\pi_8$
$x_2x_7x_{11}$	$A_{31}, A_{32}$	$\pi_9$
$x_3x_5x_{11}$	$A_{33}, A_{34}$	$\pi_{10}$
$x_3x_6x_{11}$	$A_{35}, A_{36}$	$\pi_{11}$
$x_3x_7x_{11}$	$A_{37}, A_{38}$	$\pi_{12}$

Table 12.10

Probabilistic matrix (fuzzy) transitions in the automaton A

Input signals $x_i(t)$	Output signals AC $A_M^1$				Probabilistic output signals MFIS $A_y^2$				Matrix (fuzzy) blocks $e_i$ of its state			State Blocks $\pi_j$
	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$	$y_7$	$y_8$	$e_1$	$e_2$	$e_3$	
$x_1, x_8, x_{11}$	1	1	0	0	0	1	0	0	$A_{22},$ $A_{22}$			$\pi_4$
$x_1, x_8, x_{11}$	1	1	0	0	0	0	1	0	$A_{23},$ $A_{24}$			$\pi_5$
$x_1, x_8, x_{11}$	1	1	0	0	0	0	0	1	$A_{25},$ $A_{26}$			$\pi_6$
$x_2, x_8, x_{11}$	0	0	1	0	1	0	0	0		$A_{27},$ $A_{28}$		$\pi_7$
$x_2, x_8, x_{11}$	0	0	1	0	0	0	1	0		$A_{29},$ $A_{30}$		$\pi_8$

$x_2, x_8, x_{11}$	0	0	1	0	0	0	0	1		$A_{31},$		$\pi_9$
										$A_{32}$		
$x_3, x_8, x_{11}$	0	0	0	1	1	1	0	0			$A_{33},$	$\pi_{10}$
											$A_{34}$	
$x_3, x_8, x_{11}$	0	0	0	1	0	0	1	0		$A_{35},$	$\pi_{11}$	
										$A_{36}$		
$x_3, x_8, x_{11}$	0	0	0	1	0	0	0	1		$A_{37},$	$\pi_{12}$	
										$A_{38}$		

The theoretical probability  $P_h$  of such a fuzzy transition to the state  $A_h$  is equal to the product of the probabilities of the transitions  $P_{e_i}$ , that is,

$$P_h = P_{e_1} \times P_{e_2} = \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$$

In real  $N$ -level memory devices ( $N \geq 3$ ), the probabilities of fuzzy transitions vary from 0 to 1 ( $0 \leq P_h \leq 1$ ).

### 12.3. Conclusion to Chapter 12

In this chapter, the principles and methods for designing an elementary three-level discharge of a neuron were considered and its unambiguous, enlarged deterministic, probabilistic and fuzzy transitions are shown.

It is shown how to construct such arbitrary elementary neurons with similar properties, which is very important for developers of neurons and neural networks.

This approach is somewhat different from the known biological neurons, but functionally approaches them in terms of their functional qualities of transitions.

However, the biological neuron possesses qualities of connection with other neurons. So the output signals of the neuron can be connected through an axon with

10 000 similar neurons, and the input signals through the dendrites are connected to other neurons, forming exciting and inhibitory signals.

In the next chapter 13 we will consider the structure of a neuron, consisting of elementary neurons, which will solve the problem of communication with other neurons.

## **Chapter 13**

### **ARTIFICIAL NEURAL NETWORKS**

#### **13.1. The history of research in the field of neural networks**

In the history of research in the field of neural networks, as in the history of any other science, there have been successes and failures. In addition, the psychological factor, which manifests itself in a person's inability to describe what he thinks, is constantly affected here.

The ability of a neural network to learn was first explored by J. McCulloch and V. Pitt. In 1943 their work Logical Calculus of Ideas Relating to Nervous Activity was published, in which the neuron model was constructed, and the principles of constructing artificial neural networks were formulated [3].

A great impetus in the development of neurocybernetics was given by the American neuro-physiologist Frank Rosenblatt, who in 1962 proposed his own model of a neuron network, the perceptron [12]. The article was perceived at first with great enthusiasm, but then it was subjected to intense attacks from major scientific authorities. And although a detailed analysis of their arguments shows that they denied not exactly the perceptron that Rosenblatt proposed, large studies on neural networks were curtailed for almost 10 years.

Despite this, in the seventies many interesting developments were proposed, such as, for example, a cognitron that was able to recognize images well enough independently of rotation and zoom.

In 1982, the American biophysicist J. Hopfield proposed an original model of a neural network, named after him. In the next few years, many effective algorithms have been found: the counter-flow network, bi-directional associative memory.

Since the 1970s, the Kiev Institute of Cybernetics has been working on Stochastic neural networks. The neural network model with back propagation.

In 1986, J. Hinton and his colleagues published an article describing the model of the neural network with the algorithm of its training, which gave a new impetus to research in the field of artificial neural networks.

The neural network historically consists of many identical elements - neurons, so let us begin with them the consideration of the work of an artificial neural network.

### **13.2. Basic concepts Urgency of the problem**

In 1992, the program "Calculation in the Real World" was adopted. The main goal of the new program is to enable the computing systems to interact with the real world without the mediation of a person. A fairly large part of the program - 30-40% - is devoted to the study of natural neural networks and the development of artificial neural networks and neural network systems.

Artificial neural networks are computing devices based on the use of a large number of very simple neurons.

Recently, there has been a growing interest in artificial neurons and neural networks from the European Union and the United States, which are financing the largest projects in this field in the history of the latest technologies. Essentially new computers are being created, which are based on the components of the computer hardware element base, without changes in the construction of the first computers for 70 years [1 - 2].

The McCulloch and Pitts article in 1943, where they described a threshold model with an activation function, gave impetus to the creation of virtual neurons and neural networks [3].

In works on neurons and neural networks one should remember such famous scientists who made a significant contribution in this field, such as: N. Wiener [4; 9], D. Hebb [5], A. Turing [6-7], Taylor [8], Walter [10], B. Widrow and M. Hoffom [11], F. Rosenblatt [12], M. Minsky and S. Papert [13], A. Galushkin [14], P. Verbos

[15], A. Ivakhnenko [16], L. Chua [17], L. Zade [18], Mamdani [19; 20], Takagi-Sugeno [21], K. Fukushima [22-23]. S. Grossberg [24; 26], von der Malsburg and Willshaw [25], T. Kohonen [27], J. Hopfield [28-30], Rummelhart, Hinton and Williams [31]), T. Sejnowski [32], Broomhead and Love [33] ], V. Vapnik [34], F. RAEK and V. Maass [35], E. Izhikevich [36].

In 2005, the Manchester project SpiNNaker was launched. The finished computer will have 65,536 identical 18-core processors, giving a total of 1179648 cores. According to the calculation of Manchester scientists this machine can emulate the work of more than a billion neurons in real time. The model of Izhikevich is based on the work of SpiNNaker. [37]. This year also launched a project on computer modeling of the human neocortex - Blue Brain Project.

Since 2008, and up to this time, the project of the US Advanced Defense Research Projects Agency (DARPA) is being developed - SyNAPSE. The project uses the Blue Gene supercomputer, which uses 8192 processors to model 10,000 neurons. To connect the neurons, about 30 million synapses were emulated. "Future programs will raise increased demands on computers and either we will have to significantly increase the computing power of chips or make them more intelligent," says Dharmendra Modha, the head of this project at IBM Research [40-43].

Researchers SyNAPSE, therefore, speak about their shortcomings: the created chips are not yet able to be reconstructed, and the memory is still not in neurons.

Analyzing the achievements and shortcomings of the SyNAPSE project, they themselves noted the following: they use memory for triggers or memristor working in automatic discrete time and perform only one  $x_i(t)$  transition. This limits the capabilities of the devices. These devices are not investigated in automatic continuous time and are not able to carry out a transition over two variables  $x_i(t)$  and  $e_j(\Delta)$  [40-44].

The main disadvantage in constructing neurons and neural networks is the absence of:

1. new memory circuits that are able to be rebuilt during operation without loss of performance;

2. the ability to work in automatic continuous time;
3. the ability to carry out a transition over two variables  $x_i(t)$  and  $e_f(\Delta)$ ;
4. To memorize common and local information at the same time.

Researchers SyNAPSE speak out about their shortcomings as follows: the created chips are not yet able to rebuild, and the memory is still not in neurons.

The problem of finding new multifunctional memory circuits was always relevant [45]. It is worth mentioning the work on this topic by well-known scientists, such as: EVerinev and I.V. Parangishvili [46], A.V. Kalyaev [47], V.A. Mishchenko, V.D. Kozyuminsky, A.N. Semashko [48], E.P. Balashov, V.B. Smolov, G.A. Petrov, D.V. Puzankov [49], S. Tsiramua [50], E.A. Yakubaitis [51], who introduced their authorities to this important problem.

However, unfortunately, in these works the asynchronous RS-trigger was used as the basis for constructing the memory scheme, which did not eliminate such limitations as to work only in automatic discrete time and carry out a transition only over one variable  $x_i(t)$ , which refers to the basic limitations modern element base and, accordingly, to devices of computer facilities and neurocomputers built on this basis.

In 2008, a new element appeared - a memristor developed in the HP laboratory by a team of scientists led by R. Williams [52-54]. The memristor, from memory to memory, and resistor, is a passive element in microelectronics that is capable of changing its resistance depending on what flows through its charge (current integral during operation). It has found its application through its functional properties as a memory circuit in leading US firms for the construction of neurocomputers and integrated circuits [53].

However, unfortunately, in these developments the use of automatic discrete time and the implementation of transitions in only one variable  $x_i(t)$  are used.

Limitations of memory circuits used in devices of computer technology lead to the fact that the construction of reconfigurable devices is carried out at the "automatic" level and requires additional cycles (at least two) to process general and local information [56].

Modern computers built on the basis of von Neumann architecture: both data and programs are stored in the memory of the machine in binary code, with the computational module separated from storage devices, and the programs are executed sequentially, one after another. Progressive in the middle of the last century, such architecture today no longer meets the requirements for computer technology: programs have become more complex, and the volumes of data processed have grown by orders, if not in dozens of orders [55]. Modern computers mostly process information in sequence, and information is hierarchical [57]. An attempt to process hierarchical information in modern computers leads to the sequential processing of general and private information [58].

In 1996 and 1998, qualitatively new memory circuits were created that have the ability to operate in automatic continuous times and carry out transitions in two variables  $x_i(t)$  and  $e_j(\Delta)$ , which simultaneously store general and particular data of information [59-62]. Based on these memory schemes, in 2012, a new information technology was proposed that allows more efficient building of artificial neurons and neural networks [63].

These developments can be carried out in the form of computer hardware, which is very actual at the present time.

### **13.3. Analysis of the structural organization of a neuron**

Investigating the failures and difficulties of creating neural networks (NS), we come to the conclusion that fundamental research in the field of the human brain, its neuron has not yet led to the creation of an appropriate model of the neuron cell, its model of connections that characterize the real object of the brain.

In the introduction, the shortcomings of modern NS implemented on the modern element base with memory on the flip-flops were analyzed [62-64]. A conclusion is drawn with a high degree of probability about incomplete compliance of existing artificial NS - biological prototypes.

The central problem in the study of neural networks is the memory of the neuron, its input and output signals, as well as the implementation of neuronal connections and algorithms for their work [3-4].

In 2000, scientist Erik Kandel received the Nobel Prize for what he proved: when people learn something, neurons and neural connections in their brains change. He demonstrated that even seemingly unimportant information that gets into the brain entails structural reorganization of neurons. The brain constantly learns something and, accordingly, constantly changes physically. It works like a muscle. The more we practice, the more it develops. Everything we do in life affects our brain. Thus, the smallest detail can cause changes in connections and structures.

It is also proved that the structure of neural connections depends on the culture and habitat of a person. But the most interesting is that one and the same activity structures the brain of different people in different ways. In our brain there are billions of neurons, and there are even more chains between them.

Scientists have discovered many factors that affect attention. Four of them have the greatest practical potential of emotion, meaning, multitasking and time frames.

Our brain is designed so that we often remember only the essence of phenomena, missing details. It is important to concentrate on the content, try to find common patterns, connect all the details to the logical system.

By its nature, the brain is able to consistently focus at a certain point in time on only one question. Naturally, we mean the ability of the brain to concentrate attention.

The brain can concentrate only for 10 minutes. Then it requires a pause and a "reset".

Short-term memory is responsible for daily activities. The process of its work with information is divided into four stages: coding, storage, removal and forgetting. And the quality of memorization depends on the first seconds of information processing and when the information is repeated from 5 to 9 times.

Scientists have found that improving coding and, in fact, memorizing three factors: focus on the information and understanding of this information, linking to a person's personal experience and the relevant context.

Most of the information is evaporated from memory in the first minutes of its perception. The information that is stored is at first in a fragile state, but eventually strengthens. This is the nature of long-term memory - it is formed by the constant interaction between the hippocampus and the cortex of the brain. For long-term memory to be more reliable, new information should be gradually learned and repeated at regular intervals.

Legions of neurons send commands to each other, constantly changing the structure of connections in the brain.

In the process of evolution, our senses have learned to closely cooperate with each other, processing the hierarchical information of various sources. For example, vision affects the hearing, and vice versa. This means that the best way to learn is to stimulate several senses simultaneously. Vision dominates all senses and consumes half the energy.

Thus, we will draw some conclusions that will help us to focus in part on the fact that:

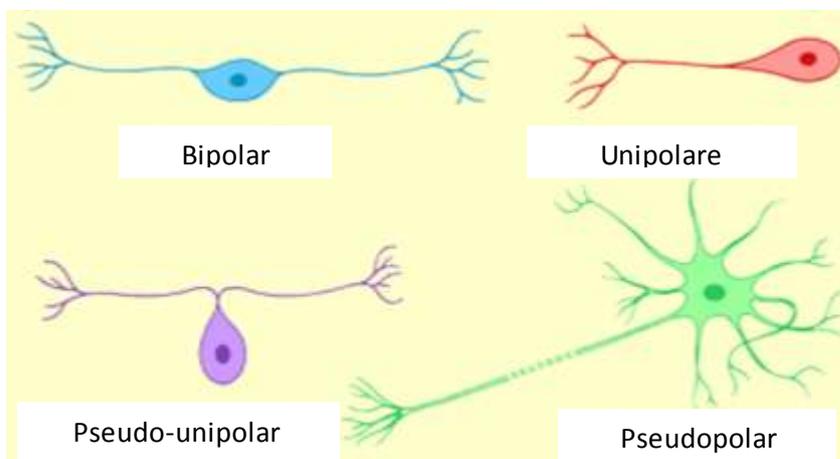
- the brain has both short-term and long-term memory, which have different properties for remembering information. For long-term memory to be more reliable, new information should be gradually learned and repeated at regular intervals;
- neurons and neural connections in the brain change. The slightest detail can cause changes in connections and structures in the brain;
- one and that activity differently structures the brain of different people;
- The brain is designed in such a way that we often remember only the essence of phenomena, missing details;
- it is difficult to concentrate on content, try to find common patterns, to link all the details into the logical system;
- The brain of a man concentrates more on the essence, and the woman's brain - on details.

Of course, these are important functionalities, but not the only ones.

The nervous system consists of neurons and nerve cells and neuroglia, or neuroglial cells. Neurons are the basic structural and functional elements, both in the central and peripheral nervous system.

Neurons are excitable cells, that is, they are capable of generating and transmitting electrical impulses (action potentials). Neurons have different shapes and sizes, form processes of two types: axons and dendrites. In neurons, there are usually a few short branched dendrites through which the impulses follow the neuron's body, and one long axon through which the impulses go from the neuron's body to other cells (neurons, muscle or glandular cells). Transmission of excitation from one neuron to other cells occurs through specialized contacts - synapses.

The brain uses different types of neurons (Figure 2.1). Based on the number and location of dendrites and axon, neurons are divided into non-zero, unipolar neurons, pseudo-unipolar neurons, bipolar neurons, and multipolar (multidendritic trunks, usually efferent) neurons.



**Fig. 13.1.** Different types of neurons in the brain

- Bezaksonnye neurons - small cells, grouped near the spinal cord in the intervertebral ganglia, which do not have anatomical signs of separation of the processes into dendrites and axons. All the processes in the cells are very similar. The functional purpose of bezasone neurons has been poorly studied.

- Unipolar neurons - neurons with a single process, are present, for example, in the sensory nucleus of the trigeminal nerve in the midbrain.

- Bipolar neurons - neurons with one axon and one dendrite located in specialized sensory organs - the retina of the eye, the olfactory epithelium and the onion, the auditory and vestibular ganglia.

- Multipolar neurons - neurons with one axon and several dendrites. This type of nerve cells prevails in the central nervous system.

- Pseudo-unipolar neurons - are unique in their kind. From the body departs one process, which is immediately T-shaped. All this is the only tract covered with a myelin sheath and structurally is an axon, although one of the branches of excitation does not go from him, but to the body of a neuron. Structurally, the dendrites are the branches at the end of this (peripheral) process. The critical zone is the beginning of this branching, which is outside the body of the cell. Such neurons are found in the spinal ganglia.

Neuroglia cells are concentrated in the central nervous system, where their number is ten times the number of neurons. They fill the space between the neurons, providing them with nutrients. Probably, the cells of the neurology are involved in the preservation of information in the form of RNA-codes. In case of damage, the cells of the neuroglia actively divide, forming a scar at the injury site; cells of another type of neurology are converted into phagocytes and protect the body from viruses and bacteria.

Discussion is the question of the presence or absence of a membrane, through which signals from a synapse enter the neuron.

1. The physical theory of Ling is an alternative to the conventional membrane theory that explains the four fundamental properties of a cell by the properties of its plasma membrane;

2. penetration;

3. the ability to selectively accumulate substances;

4. ability to maintain osmotic stability;

5. the ability to generate electrical potentials of different amplitudes.

The fundamental nature of these properties lies in the fact that our understanding of their mechanism also determines our representations from virtually any question of the vital activity of the cell.

The theory of Ling is based not on the properties of the membrane, but on the sorption properties of proteins, wherever they are located in the cell. First of all, we are talking about the binding of the mass components of the cell - water and  $K^+$  ions. The adsorption properties of proteins do not remain unchanged, but depend on the conditions of the microenvironment.

Like any new theory, Ling's theory allows us to take a fresh look at the established ideas in the physiology and biophysics of the cell, and outline new prospects for research. Depending on the representation of the vital activity of the cell, artificial neurons and neural networks are currently being constructed. At present, the construction of neurons by the membrane predominates [40-43].

IBM specialists have developed an advanced processor that mimics the work of the human brain. "This is our first cognitive computer core, which combines calculations in the form of neurons, memory in the form of synapses, and communications - in the form of axons," the head of the research of Dharmendra Modhu told CNET.

Now researchers say that the chips created so far are able to analyze the data, but neurons are not able to rebuild, and also do not have memory that is not in neurons. These abilities, as the researchers hope, will appear in them in the future.

Cognitive calculations, similar to those performed by the brain, will use repetitive computational blocks. Neurons and synapses implemented in mixed analog-digital asynchronous parallel distributed reconfigurable specialized and fault-tolerant biological substrates will be used with implicit memory addressing, which is updated only when information changes, blurring the boundaries between computations and data.

As you can see, the development of reconfigured memory in neurons, which IBM specialists seek, is still relevant for the creation of an AI.

In 1970, L.A. Khursin discovered that the brain consists of water crystals that remember information [73]. From the point of view of physics, living tissue is a solid

body [74]. From this follows, as Khursin argued, that the material basis of short-term memory of the human brain must be the chemical combination of oxygen with hydrogen-water  $H_2O$ , which has the property of retaining in the liquid phase the crystalline structure of the solid phase-ice.

It is noteworthy that the heat capacity of water, depending on the temperature, coincides with the second temperature point of the "melting" of ice and is set in the interval  $37-41\text{ }^\circ\text{C}$ . Khursin showed that a unique correspondence exists between the structural and energy characteristics of water and its components and the structural and information characteristics of the operative (short-term) memory of the human brain [75].

In 1988, the French doctor Jacques Benveniste published his fundamental discovery in the journal "National", which Khursin made in 1970, that water remembers information (Magazine "Abroad", No. 30 (146), 1988). He conducted experiments, pouring a "poison" into the glass of water, and then repeatedly diluted this mixture with water and after that found out the structure of this "poison". Scientists of five known laboratories in the world have studied this phenomenon and confirmed the results. At that time, this greatly embarrassed the scholarly community.

Nobel Prize winner Jean-Marie Le Nom, answering a journalist's question, said that he does not understand how a "fingerprint" of information can be stored in a liquid environment where the movement is extremely accelerated. Especially with such a dilution. Just think of  $10^{120}$  !!! For comparison, he cites data that the number of particles in the universe is  $10^{60}$ .

Collectives of scientists were created who checked this result. About 86% of the results confirmed the fact that the water will remember the information. Khursin L.A. in conversation with Marakhovsky L.F. he commented on the fact that some experiments did not yield results: "... these experiments had to be carried out at the temperature of the human body, that is, in the  $37-40\text{ }^\circ\text{C}$  range and then the results would be 100%."

The tri-unity of matter, energy and information was noted in his works by A.A. Lyapunov [65], Yu.Ya. Kalashnikov [66], V.P. Popov, I. Krainyuchenko [67-68]

and other researchers. Information is equally inexhaustible and hierarchical. Information technologies now largely determine the degree of success of a country in the world economy and politics.

WATER is able to reliably store and transmit the necessary information in both direct and reverse flow modes (Human Earth-Universe, Universe-Earth-Man). Academician V.A. Vernadsky wrote: "Water stands apart in the history of our planet. .... There is no earth substance - a mineral, a rock, a living body, in which there would be no water. Covering about three quarters of the surface of our planet, water represents the matter of life on earth." V.C. Promonenkov in the article asserts that he takes the responsibility to state that the most probable basis of such a rigid hierarchical order that we observe in the mentioned system is information that is stored and transmitted by water. It is water that is the information axis of our System [64].

Based on these studies, we come to the conclusion that:

1. communications in the brain are changing, which entails a structural reorganization of neurons;
2. the structure of the brain consists of a short-term and long-term memory;
3. neurons in the brain have different shapes and sizes;
4. if the neuron is damaged, a scar is formed in its place;
5. Discussion is the question of the presence or absence of a membrane through which signals are sent to a neuron;
6. In works on artificial neurons there are still unresolved questions about memory in neurons and their ability to self-restructure;
7. It is asserted that the water that is part of the neuron cell has a rigid hierarchical order of information that is stored and transmitted.

The formulation of the problem consists in the following: Marakhovsky to consider the development of various types of artificial neurons, the functions of which would approach the neurons of the human brain, and the development of methods for monitoring the work of the neuron and fixing its exit from the working capacity.

### 13.4. Symbols for multifunctional and multi-level memory circuits

Modern computers and neurocomputers built on a modern base with memory on the flip-flops use sequential information in the form of only input information signals  $x(t)$  and process this information in automatic discrete time and are described by Mealy (1-st kind) and Moore (2-th kind) [3-4; 12; 22-25; 31; 35-38; 40-43; 69-73].

The new information technology on the MFIS and the MUSP uses in parallel information in the form of an input information word  $p(T)$  and processes this information in automatic continuous time and is described by the Marakhovskiy automatic machines [44; 59-62; 74-75].

The MFIS is a one-level, multifunctional elementary automaton with a complete transition system and a complete output system for each of the  $r_e (r_e > 1)$  state conservation functions  $\delta_e$ , as described in Chapter 4. The MFIS can be functionally represented as  $r_e$  single-level elementary automata, each of which remembers all their states only for one of the different corresponding sets  $e_j(\Delta)$  of the input signals.

A digital language for describing asynchronous single-level memory schemes was proposed, beginning with a single-level scheme of an asynchronous  $RS$ -trigger and ending with an MFIS with 10 groups of 10 logical elements each [76].

A digital description of the elementary one-level memory is represented as a positional number (decimal or hexadecimal), which represents a structure of either an asynchronous  $RS$  trigger or an asynchronous MFIS. This number characterizes the structure so that using this number, it would be possible to formally calculate the main parameters of the memory scheme, on the basis of which to choose the optimal structure in the designer's opinion and build it on logical elements in the form of a functional memory scheme. In the digital description of the MFIS, it is advisable to enter a decimal number whose number of bits corresponds to the number of logical element groups in the MFIS, and each digit is the number of logical elements in the  $i$ -th or other group. The maximum number of digits of the decimal number in the

MFIS is 10, which corresponds to the restriction of the number of groups to 10 in the structure of the MFIS. These restrictions are purely conditional, although they correspond to some extent to the limitations of many asynchronous elements of OR-NOT (AND-NOT) integrated circuits.

The number that defines the structure of an MFIS in decimal notation has limitations on the number of NAND gates (NOR) in each group up to 9. This number corresponds to the real limitation of integrated circuits. The number of digits that characterize the structure of the MFIS in decimal notation has limitations on the number of possible inputs in the logical elements used (in this numerical description, up to 10). In the case of a digital description, the MFIS structure is specified by the number of logical elements in each  $i$ -th position of the number and the number of groups (digits) in the number itself.

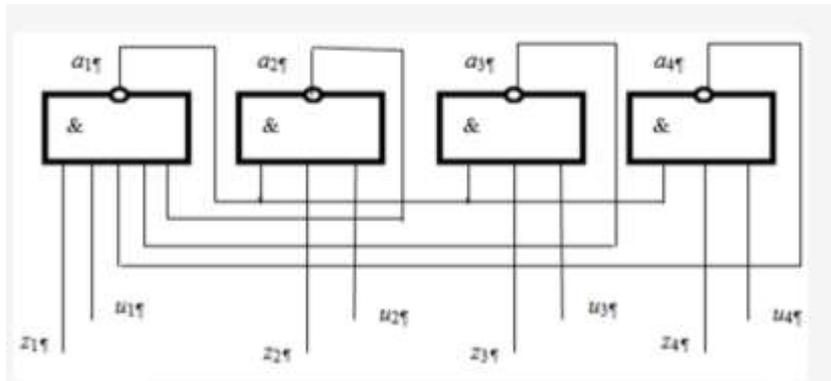
For example, a single-level scheme of an asynchronous *RS*-flip-flop in digital form is recorded as 11, since it has two groups of logic elements, one in each. The generalized block diagram of a multistable memory scheme (MSP) in digital form is written as 11 ... 1, because  $n$  groups of logical elements are located one in each. To obtain the results obtained by using the digital description when selecting the MFIS, you can perform the following steps:

1. In the case of a digital description of the MFIS, its basic parameters can be determined:  $M$  is the number of memory stable states;  $r_x$  is the number of input signals set by  $x_i(t)$  and  $r_e$  is the number of input signals  $e_j(\Delta)$ , and also select the necessary criterion that satisfies the obtained results of the basic parameters for the logical design of the MFIS and is necessary for creating promising devices for computers and networks.
2. Selecting the basic parameters of the MFIS, you can build its functional scheme based on the logical elements (AND-NOT, OR-NOT, AND-OR-NOT), as well as find its description in the form of a system of logical equations, when necessary, for simulation memory circuits.

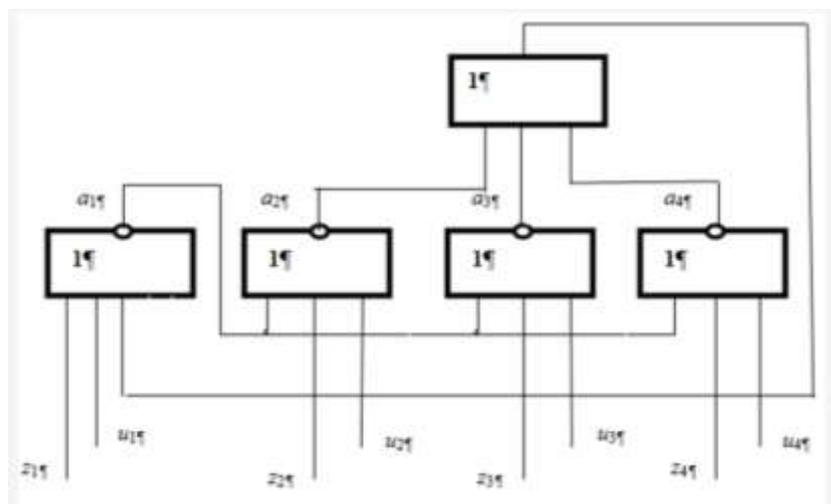
IFRS has two classes of structural solutions to functional schemes: the memory class is  $L$  and  $L^M$  [76]. The class  $L$  of the MFIS is created as follows: if the links be-

tween groups of logical elements are performed directly from the outputs of elements of one group to the inputs of elements of other groups (Figure 13.2).

The MFIS  $L^M$  class is defined as follows: if the links between groups of logical elements are carried out from the outputs of the elements of one group through the logical element OR to the inputs of elements of other groups (Figure 13.3). At the same time, the number of logical elements OR in the LM class increases by the number of groups in the MFIS, but the number of connections between groups in the MFIS decreases significantly, as is clearly seen in Fig. 13.3.



**Fig. 13.2.** Functional diagram of the MFIS class  $L$

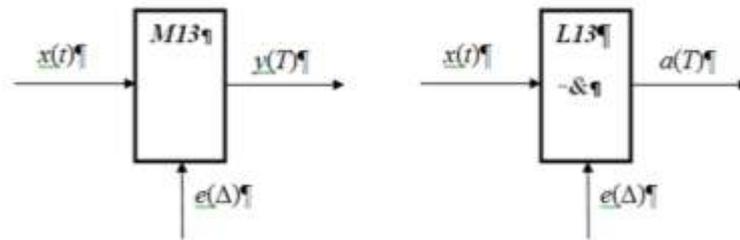


**Fig. 13.3.** Functional diagram of the MFIS class  $L^M$

In the new information technology for memory circuits, you need to create your own conventions that would allow to match these analogues of functional schemes. It is convenient to do this in the legend, when the digital code of these circuits and the

type of the logical element (NAND or NOR) will be displayed. Let's demonstrate this with examples of the memory schemes already discussed.

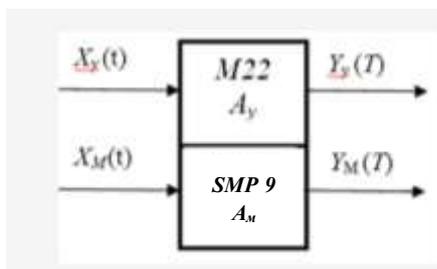
Within the most conventional designation of the MFIS, the designation  $M13$  is given, which reflects the structure of the MFIS of the  $L^M$  class (Figure 13.3). With the MFIS class  $L$ , the symbol is replaced by  $L13$  (Figure 13.2). If necessary, it is necessary to indicate in the symbolic designation on which logic elements the MFIS scheme is implemented, then under the numeric code it is possible to put the  $\&$  sign when the logical NAND elements are used, and in the absence of the  $\&$  sign, the logical NOR elements are used.



**Fig. 13.4.** Symbols of MFIS schemes

Based on the MFIS, multi-level memory schemes are constructed [76]. The principle of their construction lies in their breakdown into control  $A_M$  and  $A_y$  controlled multifunctional memory circuits, interconnected accordingly. The MUSP of the  $L_N$  class with the common strategy automaton  $A_M$  for the MISP and the class  $L_N^B$  with the strategy automata  $A_M$  for each group of the MFIS operate in automatic continuous time. In the MFIS, the input signal  $e_j(\Delta)$  preserves certain subsets of  $\pi_j$  states that allow creating new transition functions, extends the functionality of the MUSP using the MFIS in its structure. Single-stage MUSP uses several vertically interconnected MFIS, which can be synchronized with a single signal.

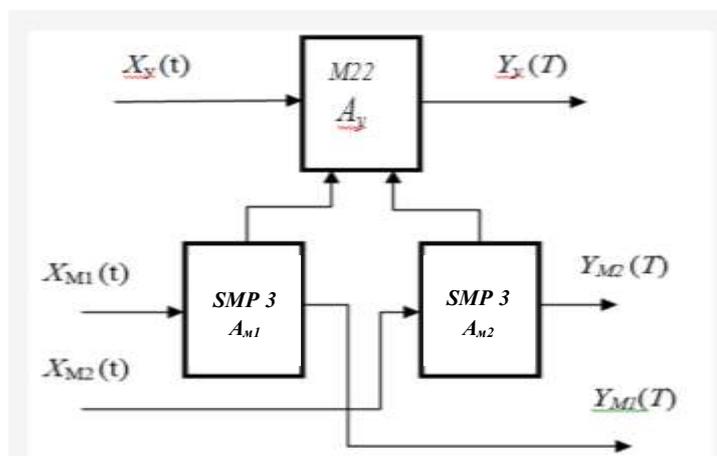
Conventional designation of MUSP class  $L_N$  can be represented in one rectangle divided into two parts: the MIPS and the strategy machine ( $A_M$ ), which includes the corresponding input signals  $x(t)$  (Figure 13.5).



**Fig. 13.5.** Symbols of the MUSP scheme of the  $L_N$  class

In Fig. 13.5 the designation  $M22 A_y$  reflects the MFIS, consisting of two groups, each of which has two logical elements OR-NOT and is built on the principle of the  $L^M$  class, and the designation of the  $SMP 9 A_u$ , reflects a SMP consisting of 9 groups, each of which has one OR-NOR. The relationship between them along the vertical is constructed according to the rules, reflected in [76].

Symbolic descriptions of MFIS and MUSP ( $n > 2$ ) are key when describing any class of MUSP. For MUSP class  $L_N$ , each symbolic number  $A_i$  of the MFIS reflects the relationship of the output signals to the MFIS of all lower levels that generate sets of input signals  $e_j(\Delta)$ , with the input nodes of the saving buses of the upper MFIS. For the  $L_N^B$  class MUSP, the symbolic number  $A_i$  of the MFIS reflects the relationship of the output signals to the MFP of all lower levels. In this case, the sets are generated saving  $e_j(\Delta)$  of input signals with input nodes for one group of logic elements of the upper MFIS, in which the number  $q$  of logical elements has a value greater than one ( $q > 1$ ).



**Fig. 13.6.** Symbols of the MUSP scheme of the class  $L_N^B$

## **13.5. Structural organization of a neuron on automatic memory circuits**

### **13.5.1. The main functional properties of the brain and neuron**

Analyzing the material of the monograph [77], one can see that the main functional properties of the brain and its main cell (neuron) are laid, which lay the foundation for the design of an artificial neuron and neural networks.

Let us dwell on the main functional properties of the brain and neuron, which characterize the level of our knowledge about them.

First, the neuron appears to be a complex automaton, which has a hierarchical, approximately eight-level structure, the maximum area of perception is limited approximately by the number of elements equal to 1121, that is, the number of states [78].

Secondly, on the one hand, the neuron cell has an internal core that remembers information and at rest perceives information through two channels: exciting and inhibitory (tuned), and on the other hand it has the ability to rebuild and establish connections with other neurons, which creates appropriate templates and images, which man operates.

Thirdly, the brain has a high functional reliability of work (if the brain is damaged by approximately 50% it continues to function normally and the person is able to perform qualified work at the same time!). In addition, neuron cells partially fail and are replaced by new ones, which indicates a strict control over the working capacity of a living cell, and if it is found to be partially incapable, it is replaced, and a scar is placed in the neuroglial cells surrounding the neuron.

Fourthly, the neuron has a large number of exits, reaching up to about 20 000, which allow it to communicate with other neurons, sometimes located at a rather large distance. Moreover, it is suggested that the active signal from the neuron in

case of violation of a certain neuron has the ability to bypass it and go to a certain neuron, and not immediately to all the neurons with which it has a connection. The address link, again in our opinion, can implement  $e(\Delta)$  input signals (other input signals that determine the direction of the active signal, which is not yet clearly visible!).

Fifthly, all living matter - from virus to man is provided with a reliable conveyor belt, a hereditary information thesaurus. It consists of extremely rational, communication systems, with incredible accuracy and reliability of information transfer, fantastic security against interference. And all this, in an incredibly small amount. Universal and the language of communication. He is one and the same, for all living organisms.

Sixth, in studying the processes of signaling activity, an important phenomenon was identified, which is of primary importance for the processing of information. The sense organs are arranged in such a way that the reaction to a long-acting, repeating signal fades. At the same time, a new signal is very sharply perceived, different from the others. Due to this feature, the system filters out, from the message flow, the signals, new information. The new information, appears, first of all, with changes in the environment. The ability to form temporary connections, under the action, even of minor irritants, is of decisive importance for the cognitive activity of the brain, in studying the laws of the surrounding world.

Seventh, according to the first signal system, any living organism receives signals from the external world, the state of the environment, from the Cosmos, through the wave radiation. The second signal system (language), this invention is the human brain. It is completely dependent on the first, receiving, interpreting and abstracting input signals. To transmit and receive, processed information, a person can, only in his environment, where the generally accepted code of communication: words, language, language of signs. Communicate with the same system of development, the human system will be able to, only after studying the language of this system. Thus, the second signal system is the superstructure over the first one. Eighth, it seems that after all, all the experience accumulated by a person, all the information received during life, does not pass without a trace, but is recorded and stored at a subconscious

level and transmitted, genetically, to descendants. Nature has taken care of the duplication of brain functions, the protection of the human information thesaurus. Nature reliably controls and protects the brain.

Summarizing, we can quote the words of V. Gagin [79]: "When considering the development of animate and inanimate nature, it is evident that the law of the golden section, the adaptation of living organisms, the hierarchy of information processing in the human brain, high functional reliability of the brain the brain continues to function functionally 50%! and the genetic memory of many generations of people. "

### **13.5.2. Models of an artificial neuron on automatic memory circuits**

The most acceptable device at present for building a digital model of an artificial neuron is the open structure of multifunctional memory circuits, which as a neuron have two sets of input signals: establishing (excited) and saving (inhibiting) ones. The exciting input signals of the biological neuron allow you to accumulate (accumulate) information and overcome its threshold for the output of an active signal, which can be replaced with a digital coincidence circuit from several outputs of other neurons. Such schemes of coincidence of input signals, which can be more than one, could be realized on conventional logical circuits NOR (NAND).

To create a semi-closed digital artificial neuron (DAN) scheme for the MFIS, you can use the strategy machine  $A_M$ , which, in conjunction with the MFIS, implements the MUSP. The general information is recorded in the MUSP strategy machine, which determines the output signal and its direction to the MFIS.

Neurons, together with neuroglia, self-monitor their performance. The automata of the fourth kind considered by us and the corresponding functional self-monitoring schemes allow us to carry out self-monitoring of catastrophic failures in the basic memory circuits [76; 80-81].

Neurons rebuild their structure. The proposed prototype of the DAN neuron also has the possibility of rearranging the structure of the sets of  $\pi_i$  states, which are remembered when the general information in the strategy automaton  $A_M$  changes.

In order to realize the maximum MFIS structure in DAN by the number  $M$  of memory states and the number of re input signals  $e(\Delta)$ , one can use a structure that is described in the digital language in the form of a maximum decimal number of 9999999999.

The number  $M$  of states of the MFIS is determined by formulas (4.9) and (4.10). The number  $K_i$  of the memory states in one group of the MFIS consisting of nine NAND gates is equal to  $2^9 - 1 = 511$  (4.9), and the number  $M$  of states of the entire MFIS is 5110 (4.10).

The number of re input signals  $e(\Delta)$  for the MFIS is 5110, which can be realized in a ten-bit register, each digit of which would consist of nine-digit SMPs.

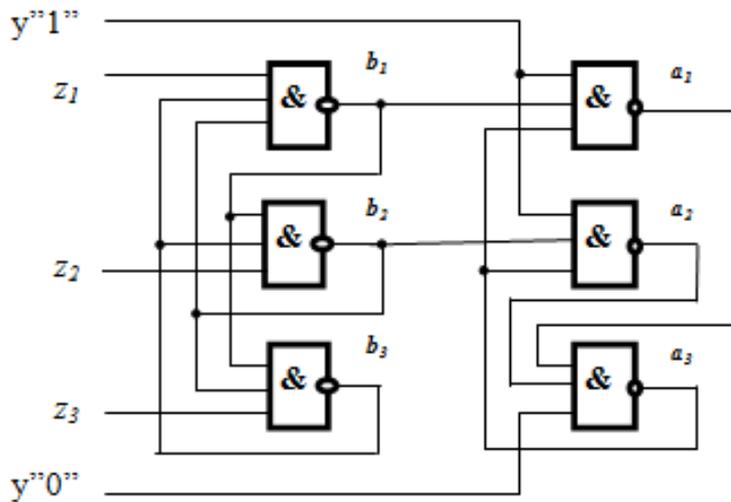
The minimum symbolic number 21 describes the MFIS by the number of logical elements  $n = 3$ . The number  $M$  of the memory states of the entire MFIS is 4 ( $M = 4$ ), and the number of re input signals  $e(\Delta)$  for MFRS is 3 ( $r_e = 3$ ). The active state of a group in which one logical element will characterize the MIPS in the zero state, when the DAN is not excited, and its three states of another group in which at least one logic element can be active, will characterize the excited states of the DAN

To create a DAN with a sufficiently large number of states, it is desirable for the MFIS to use a register with at least eight digits. Then the number of memory states is  $3^8 = 6561$ .

### **13.5.3. Structural organization of an elementary two-stage neuron on automatic memory circuits**

The structural diagram of the model of digital artificial neuron (DAN) is presented in the form of a single digit register, which can be represented as its axon, consisting of eight MUSPs with monitoring schemes to detect catastrophic failures in each MUSP. The scheme of each register bit consists of an MFIS containing two

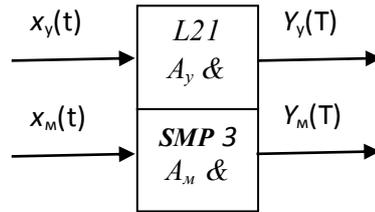
NAND gates (NOR) in one group, and in the other group one NAND gate (NOR). Control schemes for detecting catastrophic failures in the MFIS and the  $A_M$  strategy machine are presented as a three-digit SMP. Consider a functional diagram of a single digit of the DAN model without a control scheme (Figure 13.7).



**Fig. 13.7.** Functional diagram of one digit of DAN model

The conventional designations of such a discharge of the DAN model have this form (Fig. 13.8).

The choice of DAN, where the MFIS is described by the symbol number  $L2I$  with the & symbol, and the  $A_M$  strategy machine is described by the  $SMP3$  with the & symbol, is not accidental for the implementation of the DAN. The ratio of the two elements of the group of MFIS and three elements of the  $A_M$  strategy automaton has a value of 0.6 (6), which is close to the number  $\varphi \approx 0.618$ , which is characteristic for the "golden" section. However, this does not limit the DAN developer for designing an arbitrary structure on the MUSP consisting of an arbitrary scheme of the MFIS and the corresponding  $A_M$  strategy machine.



**Fig. 13.8.** Legend of one digit of DAN model

The input nodes  $y "1"$  and  $y "0"$  in all digits of the DAN register are respectively combined with the input nodes  $Y "1"$  and  $Y "0"$  by the input settling bus  $BIIIY_j$  ( $j = 1, 2$ ) of the MFIS, and the input nodes  $z_1, z_2, z_3$  are respectively connected to the nodes of the input bus  $BIIIZ_i$  ( $i = 1, 2, \dots, 8$ ) of the strategy automaton  $A_M$ .

The functioning of the scheme of one digit of the DAN register is carried out in the following way: the parallel installation of the  $A_M$  strategy machine in one of the storage states and the setting in a single state of the MFIS. According to the state of the  $A_M$  strategy machine, the MFIS generates three states, which were discussed in the analysis of the MUSP. The output signals of a single digit of the DAN are the output signals of the MFIS  $a_1$  or  $a_2$ , or the joint message  $a_1$  and  $a_2$  constitute three states. Each of the DAN states can be viewed as output signals with different weights. When the MFIS is set to zero, the output signal does not appear at the output of the entire DAN.

Thus, the condition is fulfilled that only single signals of one group, in which there are more than one logical elements, in each digit of the DAN register are codes of output states. In this DAN register, there can be  $3^8$  states that can be transmitted for excitation to other DANs. The purposefulness of connecting one DAN to another will be determined from the targeted source code. The configuration of the DAN source code for communication with other DANs is determined by the states of the DAN setting automata.

When you turn on the register of the control circuit in each digit, it is possible to replace a faulty discharge of the DAN register with a working one if it is physically replaced or replaced during simulation in the computer memory. The DAN register

can be compared with a biological neuron in this way. Neuron excitation occurs when summing signals coming from the outputs of other neurons to the synapses of a given neuron. In the DAN register, the excitation (setting in the MFIS unit) occurs when the code from the other DAN register arrives at node "1".

When a braking signal arrives at node  $y^0$  of the DAN register (setting all the MFISs to zero), the DAN register does not output the output signal and the DAN register does not communicate with another DAN register, as it does in the biological neuron.

The automata of the  $A_M$  strategy in the DAN register play the role of dendrites, which determine the direction of the source code of the register necessary for communication with a particular DAN register.

The eight-bit source code of the DAN register can be compared to a long axon in a biological neuron.

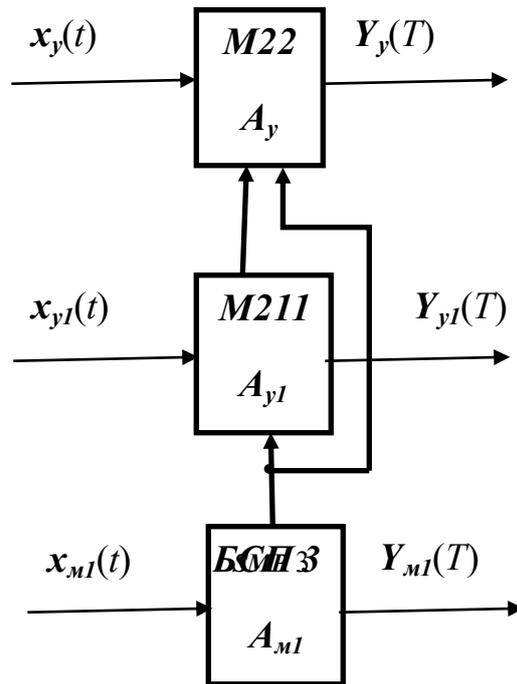
Of course, one must understand that its analogies do not cover the entire complexity of the biological neuron, but sufficiently fully describe its known functions. In addition, such a neuron implemented on automatic memory circuits will be more adequate to the neuron than the proposed neuron circuits built with memory on the triggers.

Due to the fact that the neurons in the brain may be different, we suggest looking at the DAN conventions on the three-level memory scheme (Fig. 12.5), which has more extensive properties.

#### **13.5.4. Symbols of the elementary three-stage neuron on automatic memory circuits**

In the machines of Marakhovsky 3rd kind, transitions are considered that expand the capabilities of computer hardware devices built on triggers [63-64]. Graph-schemes of these transitions are used in the elementary artificial three-level neuron. These are single-valued, enlarged, probabilistic and fuzzy transitions, presented respectively in Fig. 1.5, Fig. 1.6, Fig. 1.13 and Fig. 1.14.

The conditional designation of the scheme of a three-level elementary artificial neuron on MUSP (Figure 13.9) and its operation.



**Fig. 13.9.** The conventional designation of the scheme of a three-level elementary artificial neuron on the MUSP

Each MFIS  $A_y$  ( $M22$  and  $M211$ ) consists of four ( $n = 4$ ) OR-NOR logical elements, divided into  $m_j$  ( $2 \leq m_j \leq 3$ ) groups. Elements  $A_{M1}$  (**SMP 3**) are divided into three groups so that in one group there is one logical OR-NOR. The number  $M_j$  of memory states in each  $j$ -scheme of memory is determined by the formula (5.3).

In those groups where the number  $q$  of logical elements is more than 1 ( $q > 1$ ) and, the outputs of the  $LE$  are interconnected with the inputs of the  $LE$  of other groups through the logical element OR, the intergroup relations in the MFIS  $A_y$  are reduced.

The numbers  $r_e$  of the input signals  $e(\Delta)$  in the MFIS  $A_y$  ( $M22$  and  $M211$ ) are determined by the formula (4.14).

The number  $M$  of memory states of a three-level memory is determined by the product of the number of groups in each DAN memory scheme, which is clearly seen from the symbol (Fig. 13.9).

$$M = 2 \times 3 \times 3 = 18$$

The connections between the outputs of the  $A_M$  circuitry of the  $A_{MI}$  circuit (**SMP** 3) and the inputs of the MFIS  $A_y$  ( $M22$  and  $M211$ ) are determined from the  $e_j(\Delta)$  values of the input signals of the MFIS  $A_y$  ( $M22$  and  $M211$ ). The input signals  $e_j(\Delta)$  are characterized by the fact that, at least in two groups of  $LEs$  at the input nodes, the value is equal to one logical zero.

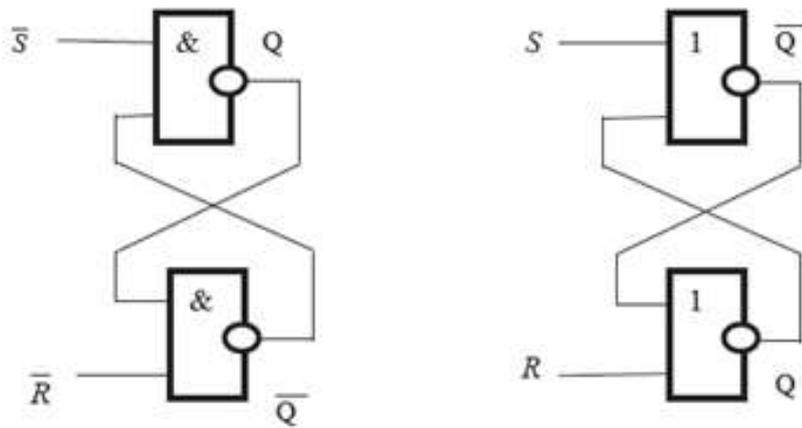
The combined three-level elementary DAN is able to function as nine different  $RS$ -triggers (Figure 5.6).

The three-level DAN in deterministic modes operates as an elementary multifunctional automaton of Marakhovskiy of the second kind and is capable of operating as nine  $RS$ -triggers (Figure 5.6), as three six-element memory circuits (Figure 2.20-2.22) or as one 18th memory element, using all of its 18 states of the memory scheme.

## **13.6. Control schemes for an artificial neuron and communication between neurons**

### **13.6.1. Introduction**

Basic memory circuits (or elementary automata with memory) is called a schema, which consist of  $LE$  and  $n$  are divided into  $m$  groups ( $m \leq n$ ). The input and output nodes in them are connected respectively with the input  $BxIII$  or output  $B_{yx}III$  which the tires of the memory circuits. For example, a basic asynchronous  $RS$ -flip-flop consists of two  $LE$  AND-NOT (OR-NOT), divided into two groups, each consisting of one logical element (Fig. 13.10).



**Fig. 13.10.** Asynchronous *RS*-trigger

On its basis, different triggers are designed, just like: *D*-, *T*-, *E*-, *D-V*-, *JK*-type [82].

If one of the *LE* fails, the trigger loses its memory and is unworkable.

In the human brain, neurons do not have a durable life. In addition, with head injuries, they also fail, but in most cases the brain continues its work [79]. If the neuron is damaged, a scar is formed in its place. This indicates that in its structure there are controlling elements that signal about its incapacity for work.

Neurons are located around the neurons, which are several times larger than the neurons themselves. They perform the control of the neuron's operability. When it fails, they form a scar which gives a signal to the brain that the neuron is out of order. While this problem of monitoring the efficiency of the neuron has not been considered in the scientific literature on neurons and neural networks.

The structure of the brain consists of short-term and long-term memory [78]. Interesting opinion on the account of short-term memory: with the teacher can consider the solution of the problem several times and with the correct decision to write the result into long-term memory. In the long-term memory, communication paths in neurons should be recorded that help implement this algorithm. For this, we need to develop our own language for recording information in long-term memory.

Discussion is the question of the presence or absence of a membrane through which signals enter the neuron, accumulating up to a known threshold [40]. Another

approach to the realization of input signals in neurons is possible, when there is a lack of a membrane in the neuron.

The physical theory of Ling is an alternative to the generally accepted membrane theory that explains the four fundamental properties of a cell with the properties of its plasma membrane:

- 1) semipermeability;
- 2) the ability to selectively accumulate substances;
- 3) ability to maintain osmotic stability;
- 4) the ability to generate electrical potentials.

Fundamental these properties are because our understanding of their mechanism determines our representations and practically on any issue of the vital activity of the cell.

Ling's theory is based not on the properties of the membrane, but on the sorption properties of proteins, wherever they are in the cell. First of all, we are talking about the binding of the most mass components of the cell - water and  $K^+$  ions. The adsorption properties of proteins do not remain unchanged, but depend on the conditions of the microenvironment.

Like any new theory, Ling's theory allows us to take a fresh look at the established ideas in the physiology and biophysics of the cell, and outline new prospects for research. Thus, Na, K-ATPase is not represented by a pump, but by an ion receptor; explains why this enzyme can selectively bind to  $Na^+$  and  $K^+$  ions and why the selectivity to these cations can vary.

It is argued that the water that is part of the neuron cell has a rigid hierarchical order of information that is stored and transmitted [57; 78]. In works on artificial neurons there are still unresolved questions about memory in neurons and their ability to self-restructure.

### 13.6.2. Neuron structures with fixation of disability function

As can be seen, the DAN of a single digit can be two-stage or three-stage. The number of steps can be increased if it is necessary to apply DAN to eight. Each stage of the DAN is an elementary automaton, which can be together with a controlled automaton of the 4th kind [76; 81-82].

New information technology, as a memory, uses multifunctional memory circuits (MFISs) with a matrix state memory structure and multi-level memory schemes (MUSP). They use hierarchical information in the form of an input information word  $p(T)$ , which consists of two input signals  $x(t)$  and  $e(\Delta)$ , processes this hierarchical information in automatic continuous time and is described by multifunctional automata Marakhovsky (1st, 2nd and third kind) and multi-level automata [63].

For the first time, a technique for analyzing operability in case of catastrophic failures in basic elementary memory circuits was developed [80-81].

About performance in the basic memory circuits for catastrophic failures was reported at an international conference and a positive response was received [83].

The performance check for catastrophic failures in the basic memory circuits is performed when the active input signal  $x_p(t)$  appears on the elementary memory circuit (trigger, SMP or MFIS). The input signal  $x_p(t)$  is characterized by the fact that at each input of the NAND gate the active logical 0 is fed, which in the original signal gives the opposite value of 1, and for each input of the NOR gate, an active logical one 1 is fed, which at the initial the signal in gives the opposite value of 0.

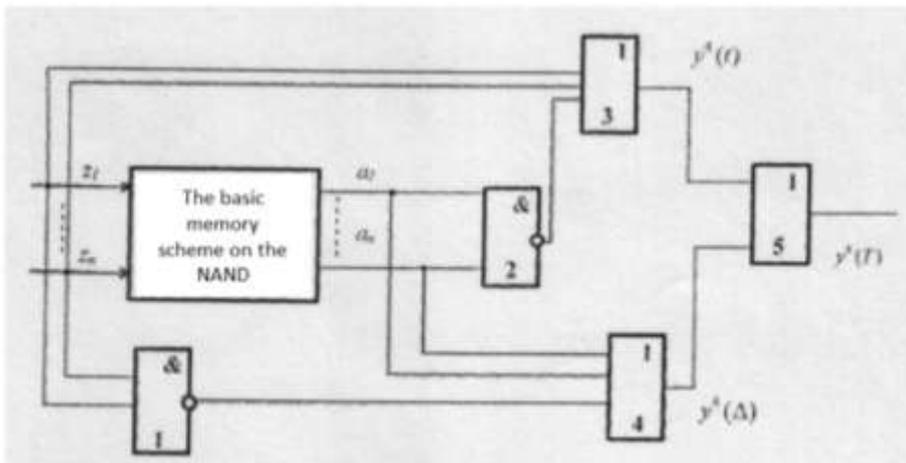
In catastrophic failures in basic memory circuits, at least one output signal is sufficient to have a value of logic 0, so that memory circuits on NAND elements are inoperative, and on the output signal in the output of the opposite value of 0. For catastrophic failures in basic memory schemes, at least one output is sufficient to have a logical value of 1, so that the memory circuit on the NOR elements is inoperable.

Thus, when the input signal  $x_p(t)$  appears on the elementary memory circuit, it is sufficient to have at least one active output signal  $y$  at the output, then the monitoring circuit indicates that this memory scheme is inoperative.

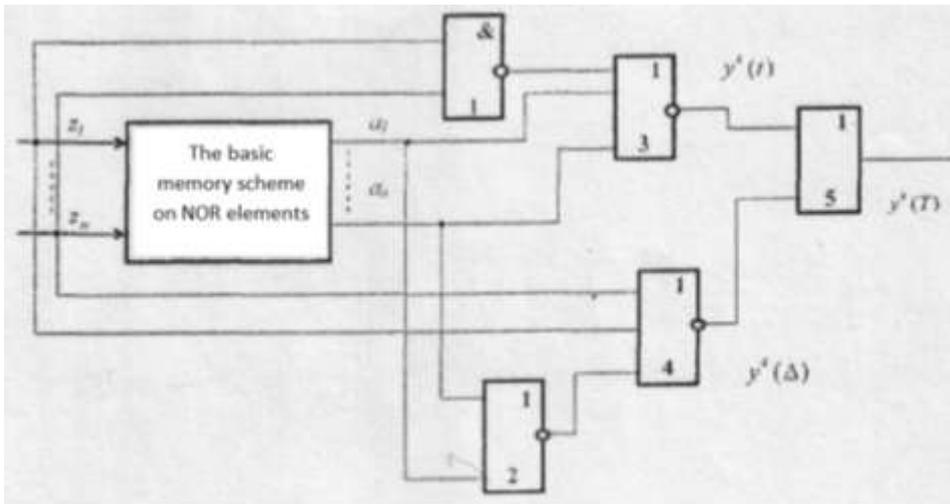
Schemes to verify the operability of elementary memory circuits on NAND logic elements and memory elementary circuits for NOR gates are presented in Fig. 13.11 and Fig. 13.12.

To fix the disability of elementary memory circuits for a multilevel neuron for each level (Figure 3.6), add a trigger to which the results of the verification are recorded. Using the code of the triggers, which are designed to test the disability of elementary memory circuits of a multilevel neuron, it is possible to bypass the disabled neurons in the software of the neural network.

Thus, for a single digit of the N-stage DAN control, a binary N-bit register is required in which each setting of a particular trigger at 1 will correspond to catastrophic conditions in the basic memory circuits on certain elements.



**Fig. 13.11.** The scheme for testing the operability of basic elementary memory circuits on the AND-N logic elements



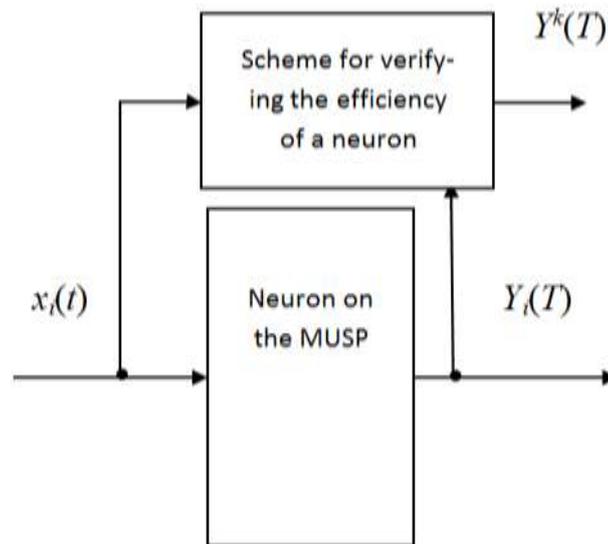
**Fig. 13.12.** A scheme for testing the operability of basic elementary memory circuits on NOR gates

The failure of one of the elementary automata will testify to the narrowing of the capabilities of the DAN, and when the entire DAN discharge occurs, all control triggers will be set to 1.

Thus, for a single digit of the control of the N-step DAN, a binary N-bit register is required in which each setting of a specific trigger at 1 will correspond to catastrophic conditions in the basic memory circuits on certain elements.

The failure of one of the elementary automata will testify to the narrowing of the capabilities of the DAN, and when the entire DAN discharge occurs, all the control triggers will be set to 1.

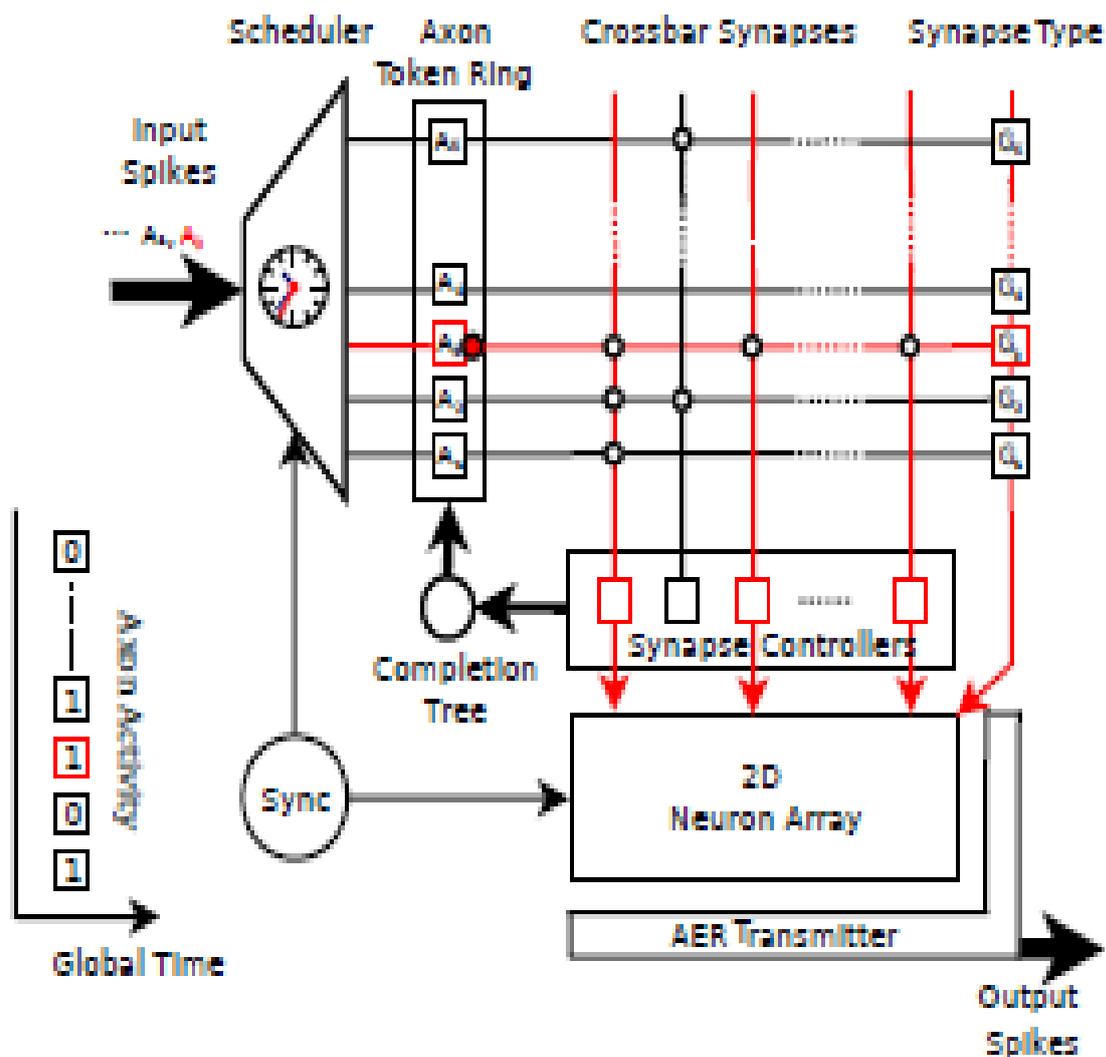
Nero, together with the scheme for verifying the operability of his work in the form of symbols, looks as follows (Figure 13.13).



**Fig. 13.13.** Conventional notation of the neuron with the scheme of its verification

### 13.6.3. Existing functions of the neuron

IBM, from 2008 to 2014, has developed a key unit of a scalable neuromorphic architecture that can manage the surge of neural networks in compact and low-power equipment. In Fig. 13.14 represents the architecture of the neurosynaptic nucleus with  $K$  axons and  $N$  neurons. Each node shown in the matrix is fed between the synapse and axons (row) and dendrites (columns). Each neuron has a special column to enter the matrix. Active synapses are represented in the open circle of the diagram. A sequence of events is used in the kernel.



**Рис. 13.14.** Архитектура neurosynaptic ядра с  $K$  аксонов и  $N$  нейронов

Investigating the achievements of IBM, it turns out that the neurons themselves have no memory. The memory is stored in dendrites. They use in their devices automatic discrete time  $t_i$  and binary memory. The neurons themselves do not reconfigure. Without detracting from the dignity of the TrueNorth processor, it can be noted that it has drawbacks, which the manufacturers themselves pointed out: binary memory, does not rebuild a neuron that does not have memory. This platform also caused the use of automatic discrete time and memory on the triggers, which caused their fundamental limitations [64, 84].

In the nervous system, one neuron receives excitation from a huge number of neurons (their number can reach thousands). It is believed that the human brain consists of approximately  $10^{11}$  neurons, which have about  $10^{13}$  bonds among

themselves. Each neuron transmits excitation through nerve joints (synapses), while the process of information transfer has a complex electrochemical nature. Synapses function as nodes for regulating input information, as a result of which the work of excitation can be amplified or attenuated. As a result, signals come to the neuron, which indicate how to slow down and stimulate its action.

In the artificial neuron of IBM at the input nodes of the neuron, input information is provided from 256 other neurons and is supplied to 256 other neurons, which is due to the limited possibilities of electronic technology.

The artificial neuron receives information from the synapses, each of which corresponds to a specific weight of the synaptic connection, the adder and the activation function.

We introduce the following notation:

$x_1, \dots, x_n$  - input signals coming from other neurons;

$w_1, \dots, w_n$  are the synaptic weights of the neuron;

$b$  - threshold value (threshold)  $y$  - output signal of the neuron;

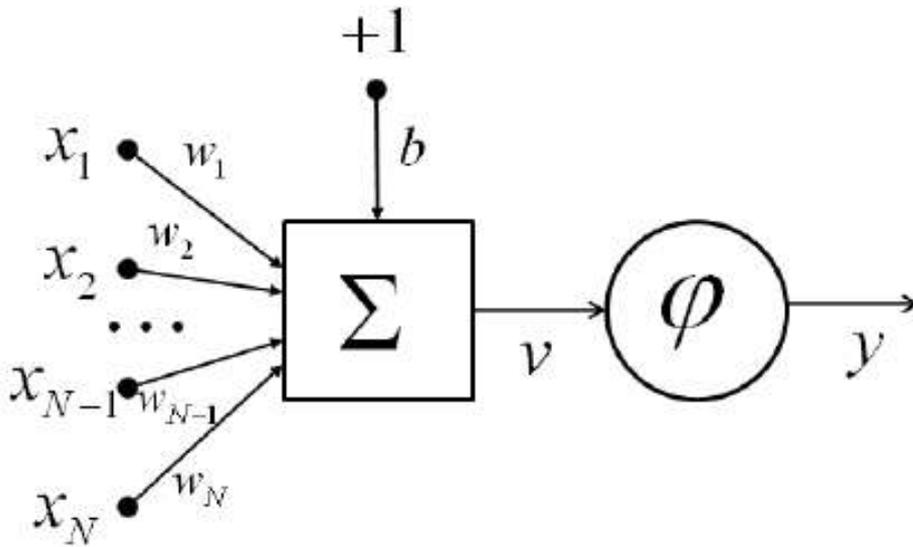
$\nu(\varphi)$  is the activation function.

In Fig. 13.16 schematically represents the model of an artificial neuron.

Mathematically, this model can be written in the form of a formula:

$$y = \varphi \sum_{i=1}^N w_i x_i + b \quad (13.1)$$

The value  $\nu = w_i x_i + b$ , obtained at the output of the adder, is called the induced local field of the neuron.



**Fig. 13.16.** The conditional scheme of an artificial neuron

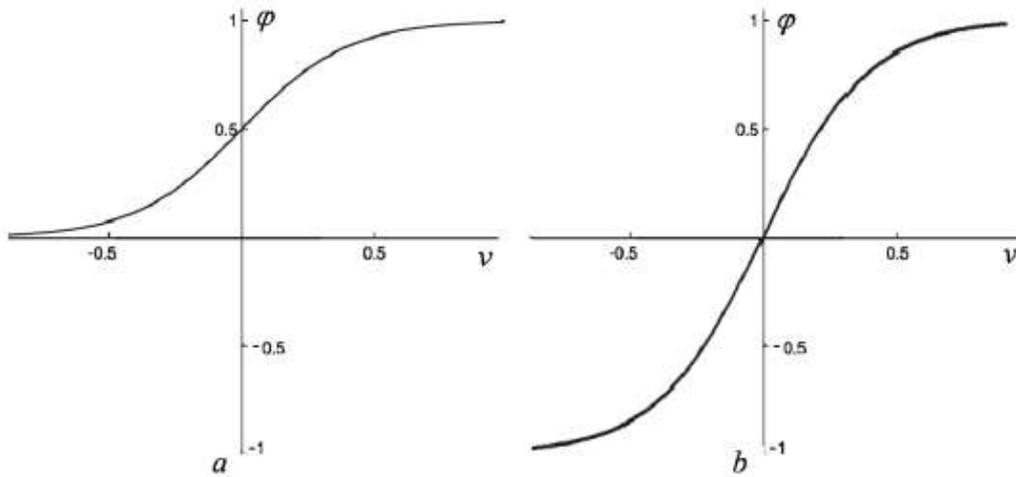
At the initial stage of modeling neural networks, threshold activation functions were used. Currently, the most commonly used sigmoidal activation function, which is determined by the formula:

$$\varphi v = \frac{1}{1 + e^{-\alpha v}}. \quad (13.2)$$

It should be noted that as  $\alpha \rightarrow \infty$ , the sigmoidal activation function tends to a threshold function. As an alternative to the sigmoidal activation function, the hyperbolic tangent function is sometimes used:

$$\varphi v = th \frac{\alpha v}{2} = \frac{1 - e^{-\alpha v}}{1 + e^{-\alpha v}} \quad (13.3)$$

The model of the neuron considered is deterministic. This means that the conversion of the input signal to the output is given by a single-valued function defined on the entire set of input signals. However, in some applications a stochastic neuron model is used, in which the activation function is of a probabilistic nature.



**Fig. 13.17.** Activation functions

In such models, the output signal of the neuron can be +1 and -1 and is determined taking into account the probability of each of the results. Thus, the activation function for the stochastic neuron will look like this:

$$\varphi v = \begin{cases} +1, & \text{with probability } P v ; \\ -1, & \text{with probability } 1 - P V . \end{cases} \quad (13.4)$$

where  $P(v)$  is the probability of neuron activation, and  $v$  is the induced local neuron field (the signal that is formed at the output of the adder).

The probability of activation of a neuron  $P(v)$  can be described by a sigmoidal function of the following form:

$$P v = \frac{1}{1 + e^{-v/T}}, \quad (13.5)$$

where  $T$  is the analog of temperature, which is used to control the degree of uncertainty of switching. It should be noted that the value of  $T$  does not describe the physical temperature of the neural network.

Obviously, as  $T \rightarrow 0$ , the stochastic neuron takes the deterministic form of the neuron with a stepped activation function of the form:

$$\varphi v = \begin{cases} 1, v \geq 0 \\ -1, v < 0 \end{cases} \quad (13.6)$$

#### 13.6.4. Self-Improving Artificial Neuron Systems at the MUSP

The human brain has opportunities for self-improvement. In an artificial neuron, an automatic discrete hour is used in the analysis of its operation and triggers are used as memory [3-50].

V.M. Glushkov in 1961 considered the question of improving systems [85]. He wrote that "... the ability for self-improvement can be laid, in fact, in any rigid algorithmic scheme."

Modification of the action of the self-learning or learning algorithm  $A$  can be considered if the given algorithm is able to convert the word  $p_i$  to the word  $q_i$ , provided that before processing the word  $p_i$  it processes the words  $p_1, \dots, p_{i-1}$  ( $i = 1, 2, \dots, m$ ). In this case, the description of algorithm  $A$  on a word  $p_i$  in the input alphabet  $X$  is determined not only by this word, but also by the history of the previous work of the algorithm. Depending on which sequence of input words has already been reworked by the algorithm before the word  $p_i$  is input to its input, the input word  $q_i$  is changed, into which an algorithm is processed that considers the input word  $p_i$ .

The artificial neuron in the multifunctional memory scheme (Figures 13.7 to 13.9), which has the matrix structure of the memory subsets  $\pi_i$  (Table 13.1), can be described as a modification of the action of the self-learning or learning algorithm  $A$ , which is able to transform the matrix structure of the storage subsets  $\pi_i$ , which is the sequence of the input words  $e(\Delta)$  was re-made simultaneously by feeding the input word  $x(t)$  in the machine clock  $T$ . To eliminate the preliminary delay, the sequence of input words  $e(\Delta)$  in the multifunctional neuron and modify state in a subset  $\pi_i$  simultaneously used multilevel neuron (Fig. 13.7 -13.9).

In multilevel neurons in the controllable elementary automaton  $A_y$  and in controllable automata of the strategy  $A_m$ , the input signals for their rearrangement  $x(t)$

arrive simultaneously. The internal influence of the output signals  $e(\Delta)$  of the automaton  $A_m$  on the automaton  $A_y$  is realized in one machine cycle  $T$ .

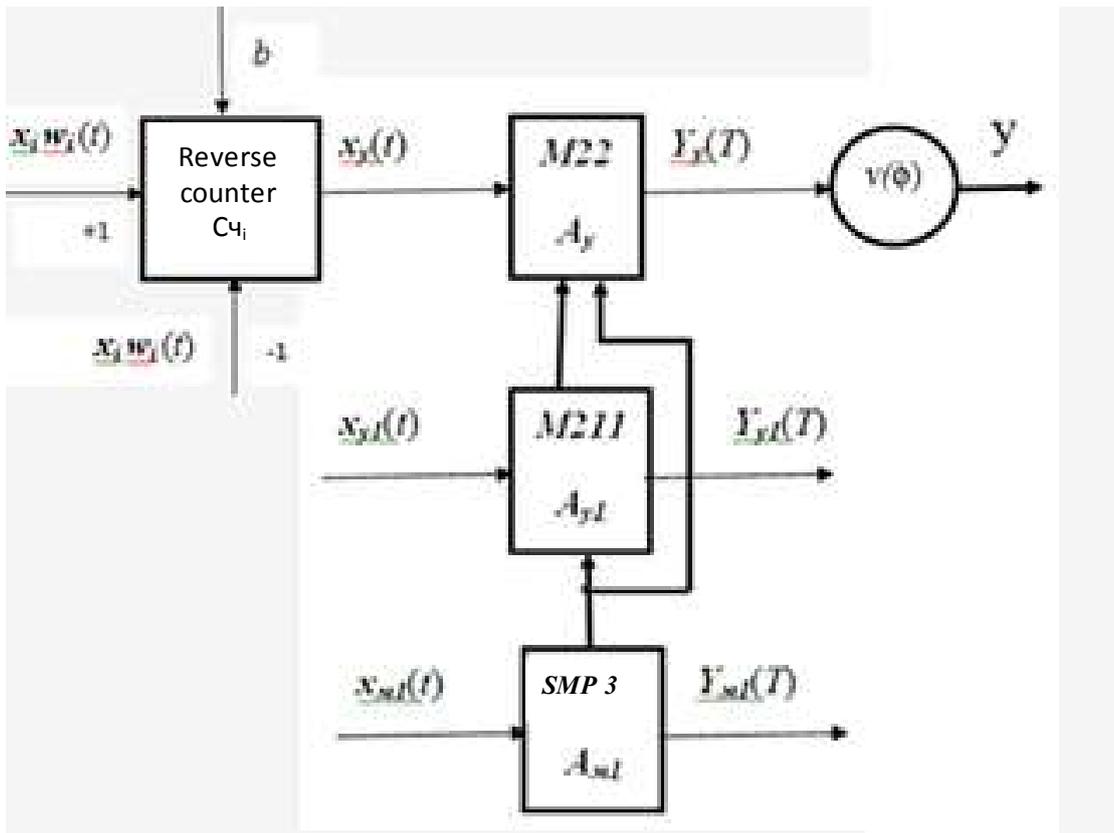
Thus, all the input signals  $x(t)$  from other neurons enter the multifunctional neuron, and the input signals  $x(t)$  of the general type of the controlled automata of the strategy  $A_m$  enter the automaton for tuning this neuron, which rearranges the work in the controlled elementary automaton  $A_y$  of the multifunctional neuron and determines one of the three active output signals:  $a_1(01)$ ,  $a_2(10)$  or  $a_3(11)$ , which can be determined respectively as: exciting output signal, braking or with increased amplitude.

A new look at the neuron is the input of the input signals  $x(t)$ , which set the automaton  $A_y$  of the multifunctional neuron to a new state that belongs to a certain subset  $\pi_i$  (Table 13.1), which is set by feeding the setting input signals  $x(t)$  to the control automata of the strategy  $A_m$ .

Input of the input signals  $x(t)$ , which set the automaton  $A_i$  of the multifunctional neuron to a new state from many output signals of other neurons can be carried out through the adder, as is done in ordinary neurons (Figure 13.9). But the difference in a neuron is that the neuron has memory and can change its state and subsets, where this state will be preserved, which allows modification of the action of the self-learning or learning neuron algorithm on the MUSP (Figure 13.9) and control its performance (Fig. 13.13).

It is proposed to use input signals  $x(t)$ , which set the automata  $A_i$  of the multifunctional neuron to a new state from many output signals of other neurons, to use a reversing counter that has two modes of operation: positive when exciting input signals  $x(t)$  appear and negative when braking input signals  $x(t)$ .

The counter is considered as an automaton, which can be built both on triggers and on the MUSP. And it can work when giving signals equal to 1 or other value (for example, 2, 3, etc.).



**Fig. 13.18.** Input signals with synaptic weights that are fed to a multilevel neuron through a reversible counter

This position is analogous to the membrane of a biological neuron, to which exciting and inhibitory signals are applied, which sometimes have an increased signal value twice. We introduce the following notation:

$x_i$  ( $i = 1, \dots, n$ ) are input signals coming from other neurons;

$w_i$  ( $i = 1, \dots, n$ ) are the synaptic weights of the neuron;

$b$  - threshold value of the reverse counter;

1 - node of excited input signals;

-1 - node of braking input signals;

$x_y$  is the input signal to the neuron;

$Y_y$  ( $y = a_1, a_2, a_3$ ) are the output signals of the neuron;

$v(\varphi)$  is the activation function;

$e(\Delta)$  is the function of the reconstruction of the neuron coming from the strategy automaton.

The arrangement of the neuron on the flip-flops (Figure 13.10) does not have the possibility of realizing a self-muting or napchauchayuchogo algorithm, which is important, since the structure of the trigger does not change [3-43].

The device of the neuron on the MUSP, consisting of the neuron itself built on the MFIS, and the two-level automaton of the  $A_M$  strategy built on the MUSP (Figure 13.13), is naturally called self-defeating or training, which can work in deterministic, probable and fuzzy regimes, as was proved higher.

The three-level memory scheme, as the basis of the neuron, resembles an artificial neuron with enhanced possibilities for transitions and changes in the structure of subsets, memorizing of different states.

- For different sets  $e_j(\Delta)$  of input signals coming from the strategy automaton of the three-level memory scheme, nine *RS*-flip-flops having different subsets of their states can function in the subsets (blocks) of  $\pi_j$  states of the MFIS.

- The three-level memory scheme, in units of  $l_i$  ( $i = 1, 2, 3$ ) states can function as three different elementary six-stable memory circuits.

- In general, the three-level memory scheme functions as an elementary 18-stable memory scheme.

In operation, the three-level memory scheme can work in three modes:

- In deterministic mode, realizing single-valued and enlarged transitions (Fig. 1.5-1.6);

- In the probabilistic mode, realizing probabilistic transitions of the first and second types (Fig. 1.13);

- In the fuzzy mode, realizing fuzzy transitions (Fig. 1.14).

Elementary neuron on the 3-level memory scheme has 5 transitions. These transitions provide functional advantages over known artificial neurons.

The elementary neuron on the 3-level memory circuit has three output signals, which can correspond to the exciting signals, the braking or the double exciting signals.

## **13.7. Structures of axons of an artificial neuron with the ability to self-reorganization for the organization of connections between neurons**

### **13.7.1. Introduction**

In the central nervous system of a person there are from 100 to 1000 types of nerve cells, depending on the chosen degree of detail. They differ in the arrangement of dendrites, in the presence and length of the axon, and in the distribution of synapses in the cell. The cells are strongly interconnected. A neuron can have more than 1000 synapses. Closely behind the functions of the cell are formed clusters: globose or parallel puffs. Hundreds of clusters are isolated in the brain. The cerebral cortex is also a cluster.

The document, published in Nature in March 2016 by the neurologist Wei-Zhong and Alain Harvard University, who works with the team of Koch and his associates, examined the scheme with 50 neurons and more than 1000 connections with other neurons. By comparing this scheme with information on the workings of each neuron in the brain, they received a simple rule - neurons with similar functions connect more to similar neurons than to types of other neurons. Andrei Tolias, a neurologist from the college Baylor, who also works together with the team of Koch and his associates, specifies that some fundamental phenomena in the structure of neuronal connections may not yet be clarified.

These neurologists examined the size of the brain about three millimeters thick, consisting of a series of repeating modules or microchips, similar to an array of logic gates in a computer chip. Each module consists of approximately 100,000 neurons, representing a complex network of interconnected cells. The data indicate that the basic structure of these modules is approximately the same throughout the cerebral cortex. However, the modules in different parts of the brain are specialized for certain purposes, such as vision, movement or hearing, etc. These scientists do not yet

know how the modules look and how they work. A David Harvard neurologist, David Koak, who heads one of the IARPA teams, said: "It seems easy to see - just open your eyes - but it's hard for computers to teach them to do the same thing."

The brain can perform analysis, synthesis of any number of functions in various ways, so you have to study its capabilities. The team, led by neurologists in conjunction with St. George's Church, believes that the brain has a built-in library of parts - bits and pieces of objects and people - and has its own rules on how to put these parts together so that there is a whole picture.

The brain can perform analysis, synthesis of any number of functions in various ways, so you have to study its capabilities. The team, led by neurologists in conjunction with St. George's Church, believes that the brain has a built-in library of parts - bits and pieces of objects and people - and has its own rules on how to put these parts together so that there is a whole picture.

Andrei Tolias and his team found three general rules that govern how cells are connected: some mainly for neurons of their own types; others who avoid their type, mainly associated with other species; and the third group only with some other types. Using only these three rules of connection, researchers can simulate the circuit quite accurately. Now the task is before the researchers to determine how to do it algorithmically. Tolias is interested in the question: "What types of accounts do they perform?"

The rules used in artificial neural networks to change the connections between neurons are almost certainly different from those used in the brain. A vital component of the biological system in the brain is an inverse relationship, both within individual layers, and from a higher order of layers to lower orders. The brain has hundreds of systems for various functions. They are linked together in a single complex system.

### 13.7.2. Structures of axons of an artificial neuron

Axon, one or none in each cell, are long, sometimes more than a meter, the original nerve fiber of the cell. The impulse is generated in the axon hillock. Axon provides impulse conduction and transmission of effects on other neurons or muscle fibers. Closer to the end, the axon often branches.

Akson connects the neuron with more than 10,000 neurons. To implement such a number of relationships that may change, it is proposed to use the register on multilevel memory schemes (MUSP).

The construction of a register on a three-level memory scheme ( $SMP^3$  13.9), which has an MFIS  $M22$ , as a controlled automaton  $A_y$ , with a memory for six states and a two-level strategy machine that consists of the control automata  $A_{y1}$  and  $A_m$ , must have at least  $6^5 = 7776$  states. Thus, an axon can be constructed on five or six parallel three-level memory circuits.

The conventional designation of the axon scheme on three-level radial brakes is given in Fig. 13.19.

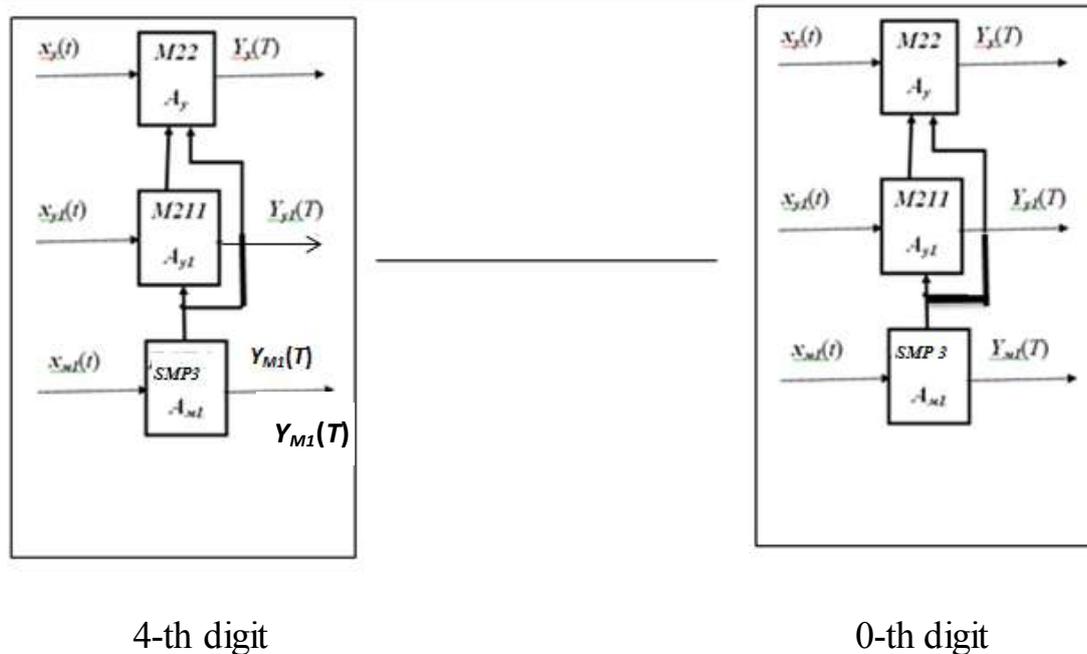
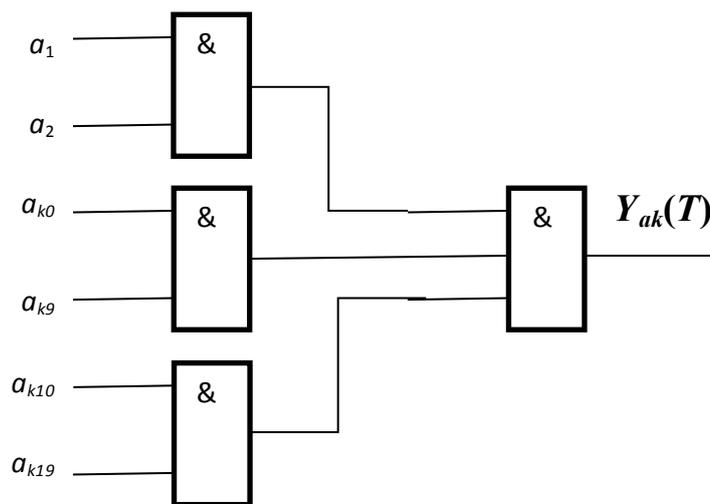


Fig. 3.19. Axon notations on 5 three-level MUSPs

The output signal of the axon register is a 20-bit code that determines the relationship of the output signal of a given neuron to the input of a necessary neuron. The output signal from the neuron has a two-digit code. According to which the signals are propagated:  $a_1(01)$ ,  $a_2(10)$  or  $a_3(11)$ , which are defined respectively as: exciting, braking or with increased amplitude.

When a neuron is close to neurons of the same type, the axon is not used and the output signal from a given neuron is directly transferred to the input of the necessary neuron. However, when a neuron is located at a distant distance from neurons of another type or on another layer, an axon is used and the output signal from a given neuron is directly transferred to the input of the necessary neuron.

When using the axon code and the output signals of the neuron, the coincidence of these signals in the combination scheme is used (Figure 13.20).



**Fig. 13.20.** Functional diagram of the axon output signal

When developing the functional scheme of the output signal  $Y_{ak}(T)$  of the axon, the limitations of the input nodes of the logic elements were taken into account, with the possibility of connecting to them no more than 10 input signals, which are superimposed on the constraints of the NAND logic elements in integrated circuits. The structure of an artificial axon for neurons with the ability to self-restructure for

the organization of connections between neurons lies in the fact that the MUSP is used in each register bit (Fig. 13.18).

When developing the functional scheme of the output signal  $Y_{ak}(T)$  of the axon, the limitations of the input nodes of the logic elements were taken into account, with the possibility of connecting to them no more than 10 input signals, which are superimposed on the constraints of the NAND logic elements in integrated circuits.

The structure of the artificial axon for neurons with the ability to self-adjust for the organization of connections between neurons is that the MUSP is used in each register bit (Figure 13.20).

In practice, self-changing (and, in particular, self-improving) devices consist of two devices: the first of which is multifunctional and processing private (separate) information, and the other device, adjusts the first, affects the change of operation of the first device in one direction or another functions. When the neuron and the axon are applied to the MUSP, the controlled MFIS  $A_y$  is realized as the matrix structure of the memory states (Table 13.1) controlled by the automaton of the  $A_M$  strategy, which changes the structure of the stored states in the MFIS.

## **13.8. Structure of short-term and long-term memory of artificial intelligence and determination of their functions**

### **13.8.1. Introduction**

Khursin L.A. in his works [78; 86] gave fundamental concepts and characteristics of short-term memory of the human brain, within the framework (limitations) of which, in his opinion, the development of any systems created by man is underway. Below are some of its definitions, for a general understanding of the development of systems created by man.

### 13.8.2. Characteristics of information and entropy

Information created by human consciousness in the form of ideal abstract structures is called free information, regardless of where it is located: in the brain of a person or on another material carrier. Free information, mediated by human labor and realized in the form of specific structures, is called additional information.

Free information can be both objective (true and represents people's knowledge of the objective world, that is, scientific information) and subjective (incorrect or distorted notions about the objective world, that is, noise).

Information that a person owns can be represented in such types:

1. Innate information - genoinformation;
2. Free information received by a person after birth, - phenoinformation;
3. Free information that a person has created himself is neo-information.

Entropy, which is determined by the amount of information created on the average by one element of the system, is called static entropy, and entropy, which is determined by the amount of information created on average per unit of time, is called dynamic entropy. Static entropy has the dimension nit / element, and dynamic entropy - nit / hour, if information is measured in natural units.

For the first time, a unit of information was designated by the "thread" ("natural bit",  $1 \text{ bit} = \ln 2 \text{ nits}$ ) A. Rapoport (University of Michigan) in the article "Experimental study of the parameters of self-organization in groups of three tested" in the book "Principles of self-organization" [87].

We give the concept of entropy given by Valery Ganin in the book [79]. Entropy can be defined as a measure of uncertainty, or as a measure of the diversity of states of a system. If the system can be in one of the equiprobable states, then the entropy  $H$  is equal to:

$$H = \log n \quad (13.7)$$

When using the multifunctional memory element proposed by Marakhovsky LF [45], which has 6 states and is capable of operating as 9 RS-type triggers, then for equiprobable subsets of states of 9-type RS-type triggers, the entropy is:

$$H = \log_2 9 = 4 \text{ bit} \quad (13.8)$$

If the number of possible states is 1, then the entropy decreases to zero:

$$H = \log_2 1 = 0. \quad (13.9)$$

Thus, we can conclude that entropy is a measure of the freedom of the system: the more entropy, the more states in the system, the more degrees of freedom it has. When states have different probabilities, then, in this case, entropy is defined as the average logarithm of probability, taken with the opposite sign:

$$H = - \sum_{i=1}^n p_i \log p_i, \quad (13.10)$$

where  $p_i$  is the probability of the  $i$ -th state,

$n$  is the number of states.

In the particular case when the probabilities of all states are the same, they are equal,

$$p = 1/n. \quad (13.11)$$

then

$$H = - \sum_{i=1}^n \frac{1}{n} \log \frac{1}{n} = \log n. \quad (13.12)$$

Khursin L.A. formulated the principle - the maximum entropy [78]:

Entropy is maximal, namely, with a uniform probability distribution. Any deviation from uniformity leads to a decrease in entropy.

The use of the extreme principle allows us to find a stable equilibrium for a very wide class of systems: physical, biological, social, etc. Let us write down this principle - the maximum of entropy in the form:

$$H(x) = \sum p(x) \log 1/p(x) = \max p(x_i) \quad (13.13)$$

Variational variables in this case are the probability of various states  $p(x_i)$ . A maximum is reached, as a rule, a conditional, since in the system there are always limiting conditions that prevent the infinite growth of entropy.

The restrictions can be different. The most typical, important and universal constraints on "resources" are the combined constant  $U(x_i)$ .

$$U(x) = \sum_i p(x_i)U(x) \leq \text{const} \quad (13.14)$$

This is a characteristic of the degree of closure of the system.

Taking into account the limitations, the principle of maximum entropy (for closed systems) is written:

$$H(x) = \beta U(x) = \max_{p(x_i)} \quad (13.15)$$

Here,  $b$  is the so-called Lagrange multiplier. It plays the role of a scale factor, allows us to bring both components in the descriptions of the expression, to a single dimension. In addition, he describes that there is a shortage of resources. The importance of the second is in the expression. For example, if the energy reserve in the system is small, then  $b$  will be large, which means that the second component will dominate the behavior - the system will mainly "save energy". If the energy reserve is large, then  $b$  will be small, and in the behavior of the system, the desire for expansion will increase, and the entropy will increase.

### 13.8.3. The model of the elementary structure of human existence

Khursin L.A. built a model of the elementary structure of human existence, consisting of elementary structures of matter and energy. The weight of a speech element in the elementary structure of matter is:

$$W_b = e^{-1}, \quad (13.16)$$

and the weight of the energy element in the elementary energy structure has a value:

$$W_s = \pi^{-1}. \quad (13.17)$$

The mathematical expectation of the quantities that form the elementary structures of matter and, the energy of the elements, have corresponding values:

$$n_b = e \text{ elem.}, \quad (13.18)$$

$$n_s = \pi \text{ elem.}, \quad (13.19)$$

and the entropy of these elementary structures is as follows:

$$h_b = \ln e = 1 \text{ fil /elem.}, \quad (13.20)$$

$$h_s = \ln \pi = 1,26 \text{ fil /elem.} \quad (13.21)$$

The amount of information in the elementary real structure matters:

$$i_b = n_b h_b = e \text{ fil}, \quad (13.22)$$

and the amount of information in the elementary energy structure matters:

$$i_s = n_s h_s = \pi \ln \pi \text{ fil}, \quad (13.23)$$

The unit of information in the speech structures (13.22) identically coincides with the number of generators (13.18), since the thread reflects in the human brain the model of the speech object itself. The unit of information in the energy structures (13.19) is not identified with the number of elements forming the structure (13.19).

The qualitative difference between the structures created by elements of the real nature and elements of the energy nature is reflected in the quantitatively different units of information that represent these structures.

To measure the amount of information of structures formed by elements of an energetic nature, a logarithm with a base  $\pi$  is used, which is denoted by

$$lp = \log_{\pi} \quad (13.24)$$

This unit of information was called a pit and then

$$lp\pi = 1 \text{ pet} \quad (13.25)$$

In pits, the entropy of the elementary energy structure corresponds to

$$h_s = 1 \text{ pet} / \text{elem}, \quad (13.26)$$

and the number of information presented in it

$$i_s = \pi \text{ pet}. \quad (13.27)$$

The dependencies between the bit, bit and pits are determined by the following relationships:

$$1 \text{ pet} = \ln\pi \text{ fil} = \log_2 \pi \text{ bit}, \quad (13.28)$$

$$1 \text{ fil} = lpe \text{ pet} = \log_2 e \text{ bit}, \quad (13.29)$$

$$1 \text{ bit} = \ln 2 \text{ pet} = \ln 2 \text{ fil}, \quad (13.30)$$

or

$$1 \text{ pet} = 1447 \text{ fil} = 1,6515 \text{ bit}, \quad (13.31)$$

$$1 \text{ fil} = 1428 \text{ bit} = 0,8736 \text{ pet}, \quad (13.32)$$

$$1 \text{ bit} = 0,6055 \text{ pet} = 0,6931 \text{ fil}, \quad (13.33)$$

Between the logarithms of the numbers 2, e and  $\pi$  there is a relation,

$$\log_2 e = \ln 2 = 1 = \text{lp } \pi, \quad (13.34)$$

$$\text{lp } 2 \log_2 \pi = 1 = \ln e, \quad (13.35)$$

$$\ln \pi \text{ lpe} = 1 = \log_2 2, \quad (13.36)$$

with the help of which there is a transition from some logarithms of the numbers 2, e and  $\pi$  to others.

The average value of the number of output elements that make up the elementary structure of human existence matters:

$$n_s = 2,71 = e \text{ elem.} \quad (13.37)$$

On the basis of this it is asserted that the natural unit of information of threads is always identified with elements of any nature - the carrier of information.

The value of this natural unit of information of threads will be:

$$\text{exp} h_s = 2\pi e = 17,079 \quad (13.38)$$

With the formula (13.38), we can conclude that a set of three symbols is optimal in the same sense in which the three-digit code is optimal.

With the conventional designation of a multilevel reconfigured neuron (Figure 13.18), three levels of the neuron are optimally selected, three states of one group of MFIS (*M22*) logical elements, and 18 states of the most three-level neuron, one of which can be taken as the zero state of the circuit, and the last 17 states are for single states.

The limit of man's knowledge of the complexity of phenomena and objects of his being matters:

$$H_n = \Delta h \times \Delta t = 7,022 \text{ fil} / \text{sec}. \quad (13.39)$$

From this it follows that a person can know the surrounding world only a limited area. The maximum size of such a limited area is determined by the maximum number of elements:

$$N = e^h - 1 = 1121 \text{ elem}. \quad (13.40)$$

These formulas mean that the number of output elements of the derivatives of all seven levels of complexity in the sum should not exceed in the closed information complex of the value (13.40). The number of output elements in a closed information complex of maximum possible dimensions is exactly the information characteristic of short-term memory of the human brain, that is exactly 735 elements that form the first level of information complexity. The number of derived elements that fill the other 6 levels have the meaning:

$$\Delta N = 1121 - 735 = 386 \text{ elem}. \quad (13.41)$$

Thus, a person's knowledge of real being is limited by the volume and structure of the short-term memory of his brain [78].

Neoinformation contains neoinformation in the amount

$$I_{uc} = 8 \text{ fil} / \text{attribute}. \quad (13.42)$$

The average value of the length of the feature set is exactly

$$N_{cp} \approx 7 \text{ attributes} \quad (13.43)$$

The processing by a person of information entering the short-term memory of his brain with the long-term and from the outside is the essence of what is commonly called mental activity of man. In short-term memory, information is synthesized and analyzed in the form of information complexes (models) of various capacities.

#### **13.8.4. Number of hierarchical levels of short-term human memory**

The information characteristics of the system are conveniently expressed in terms of the number of hierarchical levels of the system structure that range from 7 to 8.

Khursin L.A. deduced the number of hierarchical levels of short-term human memory, based on the work of Alfred Lotka, where  $n = 7,7663$ , Taranova IN and Shkaratan AI [88], where  $n = 8$ , Miller, according to the works of which Nevelsky PB. established the volume of short-term memory [78; 86].

The human brain, possessing 20 to 100 million neurons, and according to some sources up to 300,000,000, has significant limitations in structure and functionality. It has about 7 - 8 levels of hierarchy:

- The maximum area of perception is limited by the number of elements equal to about 1121 elements.
- Information characteristic of short-term memory of the human brain is 735 elements.
- The number of derived elements that fill the other 6 levels, as well as 386 elements.

Thus, a person's knowledge of real being is limited by the volume and structure of the short-term memory of his brain.

#### **13.9. Structures of neural networks**

In the opinion of the author, the structure of a neural network should consist of two parts: short-term and long-term memory, which have various functions for processing information. As it becomes clear that the short-term memory of an artificial brain must have its own blocks of brain processing. The data of the research show

that the basic structure of these blocks (modules) is approximately the same. However, the modules are specialized for solving various tasks. The tasks that solve these modules and where they are located, in most cases, are already known. But the algorithm for processing these models of tasks is still unknown in most cases.

For example, to compose an algorithm for solving problems on neurons, a person came up with an adder that is not in the brain, using a mathematical block for processing input signals and a threshold function for activating the output signals of a neuron. So far we can only guess how this is done in the brain or build an algorithm for compiling a specialized module using neurons.

In the short-term memory of an artificial brain, there must be blocks of analysis and synthesis for determining the connections of a specialized module that allow solving specific problems by the developed algorithms. The blocks of analysis and synthesis must be in specialized blocks and have connections with the main block of analysis and synthesis of the artificial brain.

In order to carry out synthesis in short-term memory, you need to have a block (library) of typical neurons, objects from which it is possible to build a complete picture. For example, attach "leaves" to trees and the like.

Creating a common system of artificial brain, we must use the basic principles (laws) of its construction. First of all, the system must be heterogeneous, having specialized modules for solving specific tasks, that is, the principle of the diversity of structures with modules from neurons is used, but these heterogeneities are related by a consistent interaction.

The system is multifunctional and multilevel - and each part of it is able to perform not only its functions, but also duplicate the functions of other parts of the system. This especially applies to the management functions of the system, where it is necessary to duplicate control commands, the presence of a separate "feedback", the existence of an independent control over the operability of neurons and the execution of control commands. The presence of one control center of the system is fraught with the consequences of its failure: another backup Control Center is needed.

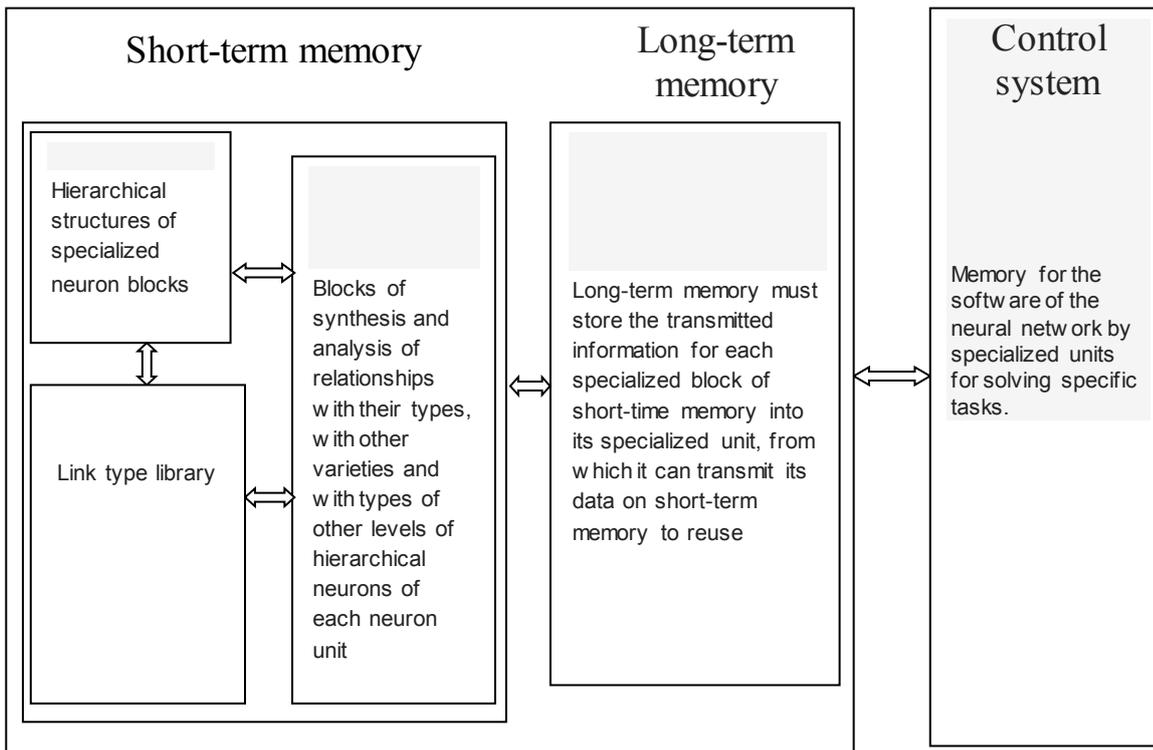
The system should ensure its sustainable development, self-learning, vitality, be able to adequately respond to external influences, possess the necessary "safety margin", or "unsinkability" of the System. In the event of the destruction of a part of the System, it is mandatory to restore (regenerate) it, and, as an extreme case, if the entire system of its recovery is destroyed after a malfunction (Phoenix effect), which requires a separate Emergency Recovery Center (Phantom).

Thus, the structure of a neural network consists of two main devices: short-term memory, long-term memory and a library of typical connections between neurons, between typical modules, between all modules of the system, objects for creating an integral picture and algorithms for solving problems, in addition, to have software for the work of the neural network with the teacher or independently.

Short-term memory has input and output signals, a hierarchical structure of individual blocks of neurons for performing certain functions and algorithms for solving a class of tasks. Blocks of synthesis and analysis of relationships with their types, with other varieties and with types of other levels of hierarchical neurons of each neuron block, a block of transmission of algorithms of neuron connections after a successful solution of the problem.

Long-term memory shall store the transmitted information from each specialized short-term memory unit to its block, from which, upon request, from the control center transmit its data to short-term memory for reuse. At the end of the transfer of information from the short-term memory of certain connections between neurons into the long-term memory, all the involved neurons are set to the initial (zero) state.

The structure of the structure of such a neural network is shown in Fig. 13.17.



**Fig. 13.17.** Structure of neural network structure

## **13.10. Structural diagram of artificial brain on multi-level memory circuits**

### **13.10.1. Introduction**

Efforts to find a specific localization of higher mental, including linguistic, functions in the human brain have a long history [49]. The paradigms that prevailed for the first time alternated depending on the state of scientific knowledge—from narrowly localized, when the areas in the cerebral cortex that provided counting in the mind or singing—were dynamic, when it turned out that almost all the brain involved in all complex functions. At present, despite the huge amount of reliable factual material accumulated over the years, the situation has become less clear and the above-mentioned paradigms continue to coexist or alternate.

Of course, an even more uncertain situation characterizes, as before, our ideas about the principles of the functioning of the brain, despite the indisputable break-

throughs and discoveries of the twentieth century, as well as the growing sophistication of the technique of functional cerebral mapping.

And yet, the basic information on the brain organization of consciousness and language can be used by us in the context of the topic under discussion. First of all, it is the semiotic heterogeneity that is provided by the cerebral hemispheres, its dual nature, the possibility of duplication, the double reading of the external, as well as internal, information [Balonov, Deglin, Chernigovskaya, 1985; Chernigov and others, 1984-1997; Davidson and Hugdahl, 1995]. The very structure of the brain is already the basis for the coexistence of at least two possible principles for the processing of any information - right and left hemispheric, with its organization, priorities and language. It is important that the right and left brains are neighbors and eternal interlocutors, Ego and Alter Ego, who are in constant dialogue, however, in an unknown language. The idea of a "brain dialogue" was emphasized at different times and in different contexts by many authors.

Vygotsky L.S., who argued that it was once a dialogue between different people, becomes a dialogue within one brain.

Ivanov Vjach.V., which emphasized that the human brain is usually regarded as a biological phenomenon, appears as a society in miniature.

Bibler V.S., described the process of "internal Dialogism" as a clash of radically different logics of thinking: the "I" of the mental and the "I" is intuitive.

Bakhtin M.M., who noted that the event of the life of the text (which is understood in the broad sense) always develops on the boundary of two consciousnesses. "Dialogic boundaries," he wrote, "cross the entire field of living human thinking."

Lotman Y.M., for many years never ceased actively developing ideas for dialogue, directly draws a parallel between the two parts of the structure of the human brain and culture, pointing to bipolarity as the minimal structure of a semiotic organization. Intelligence, according to Lotman, arises when there are internal heterogeneities. Moreover, Lotman reasonably believes that in the course of cultural development within the individual's consciousness of a person there are various psychologi-

cal "personalities" with all the complexities of communicative communication between them.

The most important factor in thinking processes, as is well known, is reflection and attempts to express cognitive and linguistic procedures. This ability is provided by some specific parts of the brain. When analyzing the experimental data obtained by us and other researchers, it seems that reflection is generally a function of the left pivil structure. It seems that the "I" and in general the ability to separate oneself from the world, as a figure from the background, (or to realize it) - is fully provided by the structures of the left hemisphere. It is possible, however, that the right hemisphere in this sense is not less developed, but has a different language that is not translated without significant losses to the ordinary language. It's not for nothing that such a perceptive thinker as Yu.M. Lotman was so keen on the idea of the impossibility of translating right hemisphere consciousness, emphasizing the unity of left-hemispheric linearity and, accordingly, discreteness and right hemispheric diffuseness, vagueness, and fundamental metaphoricity; language of associations and analogies, allegories, images and semiotic smears, not sentences.

### **13.10.2. Description of the hierarchical structure of specialized neuron blocks**

The information that comes from the external environment is generalized. This is evident from the example of the eyes. The receptors in the eye are about 18-20 million, and the cones that generalize the visible information through the eye's receptors are about 72,000. That is, information is compressed about 256 times already at the second level of information processing in the brain. This compression problem is important to understand and, if possible, to solve technically.

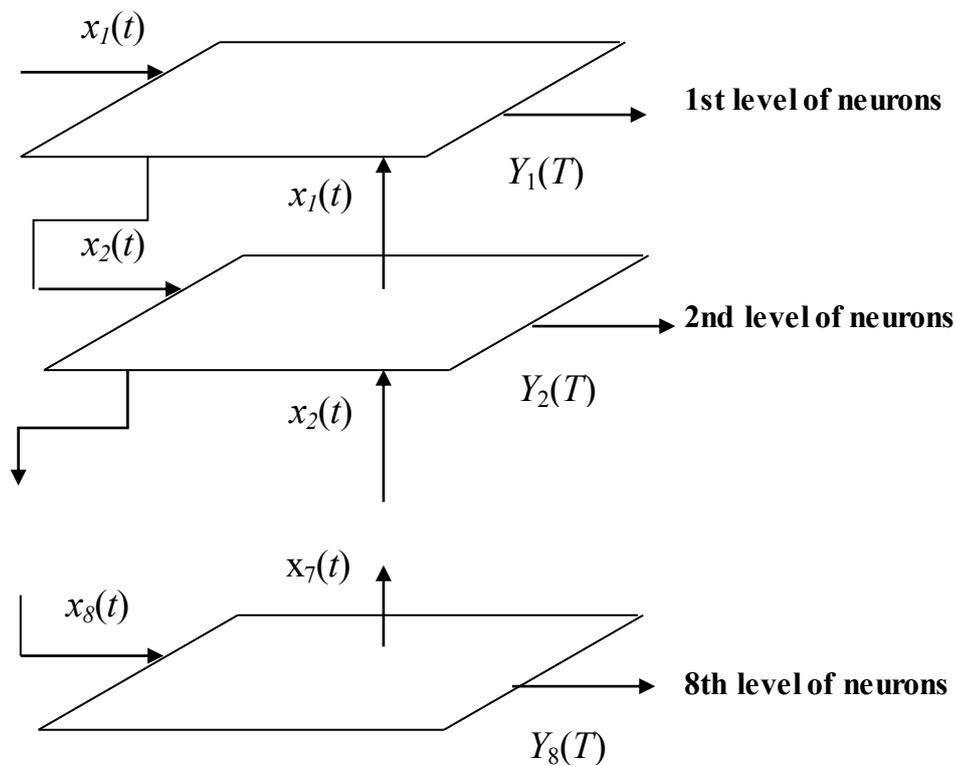
The information that comes from the external environment is generalized. This is evident from the example of the eyes. The receptors in the eye are about 18-20 million, and the cones that generalize the visible information through the eye's receptors are about 72,000. That is, information is compressed about 256 times already at

the second level of information processing in the brain. This compression problem is important to understand and, if possible, to solve technically.

The view of some scientists and developers of IBM's neurocomputers in developing a scalable neuromorphic architecture. uses 6 levels in the brain [49-51].

To construct the hierarchical structure of specialized neuron blocks, the number of neuron planes will not be refined, but we will consider them up to 8 levels, although they are still considered in the literature in the range from 6 to 8.

Fig. 13.18 provides a structural hierarchical scheme of a neural network with connections between layers (layers) of the network.



**Fig. 13.18.** Hierarchical structural scheme of the neural network

In the application of neurons to the MUSP, based on the matrix structure of the multifunctional memory scheme (MFIS), it is possible to generate the output signals of the neuron with the corresponding weights  $w_i$ , which are determined by the strategy automaton in the MUSP. With the use of the MFIS with the symbol  $M2I$ , as we have seen, the values of the output signals have three values (01, 10, 11), but with the use of the MFIS with the symbol  $M3I$ , the values of the output signals have 7 differ-

ent values, which extends the possibility to form the output the value of neurons with a large number of  $w_i$ .

Connections are made from each layer through the axons of neurons in one direction and into the other between adjacent layers of the neural network are reflected in Fig. 13.18.

As can be seen, neurons can be different in their functionality, corresponding to the purpose of biological neurons (Figure 13.1). The construction of digital artificial neurons, based on the used capabilities of multifunctional and multilevel memory schemes, the theory of microsynthesis, is given in [62-63]. Some hardware structural, functional capabilities and application of additional transitions are given in Fig. 13.7, - 13.9, Fig. 1.5-1.6, Fig. 1.13.-1.14. These provided drawings show that the developer of DAN can use them to construct their theory of microsynthesis of multifunctional and multilevel memory schemes, their variants of neurons.

An important stage in the construction of a hierarchical neural network is the possibility of organizing links between neurons of their types or with neurons of other types of type neurons, or with other degrees of the hierarchy of neurons in the neural network. At the same time, it is necessary to consider the connections, constantly expanding naturally between neurons, as it is automatically done in the biological human brain. When new information appears in the biological brain, the connections constantly increase, as was shown earlier.

The image of the expansion of the links shows how the child's connections gradually begin to be established from three months to a year, necessary to generalize the information received, and is constantly expanding, with the construction of appropriate templates and models reflecting the real world of the individual. Thus, the problem is to create opportunities to expand the links between artificial neurons with increasing information or its generalization.

For this purpose, it is proposed to organize blocks of synthesis and analysis of links with their types, with other varieties and with types of other degrees of hierarchical neurons of each block in the artificial model of the neural system (Fig. 13.18).

To this end, when analyzing a neuron, you can apply such indicators:

1. output signals appearing at the output of the neuron (exciting, exciting with doubled force or inhibiting);
2. the condition in which the neuron is located (fully working, partially working or completely unworkable);
3. setting the threshold, which is used in the reverse register to receive the input signal  $x(t)$ ;
4. Establishing a signal that must be delivered to the neuron's strategy machine, so that the signals of the neuron appear on the output of the neuron, exciting with doubled force or inhibiting.

Synthesis blocks are used to construct the output signals of links coming from the axon where it is required to arrange certain input words  $p(T)$ , which determines the connection to a certain neuron is deterministic, before the supposed connection to a neuron from a certain set of states or fuzzy connections to a neuron from an indistinct output subset neuron. In addition, it is necessary to arrange certain input words  $p(T)$ , which allows the connection of a neuron to a certain set of neurons. These characteristic links between neurons of the same type or between neurons of other types, or between neurons of other levels of the hierarchical neural network, should be concentrated in the library of model connections.

Memory for the software of the neural network by specialized blocks for solving certain problems must have a language with which you can create an algorithm with the help of the teacher to solve the problem.

The solution of the problem in short-term memory must be repeatedly checked by the teacher to confirm a positive or negative result. At the same time, at the end of the exercise, the neurons should be installed in an idle state, then to start working again.

## **13.11. Blocks of hierarchical neurons**

### **13.11.1. Introduction**

Blocks of neuron analysis are associated with an algorithm for solving a specific problem. The algorithm for solving a particular problem is first developed by the

teacher, then the result of its performance is checked, but only if it is solved positively from short-term memory to long-term memory.

In the biological brain, as described earlier, if the neuron cells are damaged, the neurons are actively divided, forming a scar at the site of neuron damage. As can be guessed, this scar indicates in the block of analysis of the biological brain about the inoperability of the neuron, and this is taken into account in the synthesis of connections between neurons. The mechanism of this analysis in the biological brain is not known to us, although the notion that it exists exists.

In the case of an artificial brain, as described earlier, it was envisaged to construct a digital artificial neuron with a scheme for fixing the detected DAN damage partially or completely. Scheme of analysis based on these indicators of the possibility or impossibility of DAN performance must calculate its way of connections between efficient neurons and implement these connections around the inefficient neurons.

Thus, each neuron constructs a neural network through the analysis of its data from the fixation register using the link analysis and the link synthesis unit to solve a particular problem in short-term memory with a check of the teacher's positive result. Then such an algorithm for solving the problem is repeated and checked with the previous result. If they coincide, the result of the connections between the neurons is transferred to the long-term memory. After writing the algorithm of the chain of these links to long-term memory, the short-term memory is cleared. In long-term memory, a record is made of the connections between neurons when the problem is solved correctly.

### **13.11.2. Analysis block**

The three-level memory scheme is considered as the base of the neuron, which can function as nine *RS*-flip-flops, three six-digit memory circuits or as one 18-bit memory element, using enlarged transitions, probabilistic transitions and probabilistic matrix (fuzzy) transitions.

An analysis of such a three-level DAN is carried out: firstly, at three levels, when input signals  $x_i(t)$  arrive. The operation of the algorithm of such a three-level DAN can be described by the dependence of the output signals  $Y_k(T)$  on the input signals  $x_i(t)$  as follows:

$$Y_k(T) = f\left(\bigwedge_{i=1}^k x_i(t)\right), \quad (13.44)$$

where  $k$  - determines the number of DAN levels.

With the functioning of the DAN algorithm, as nine *RS*-flip-flops, its operation is considered in the deterministic mode of single-valued and enlarged transitions. Analyze the direction of the links according to the levels of the hierarchical structure of the brain, one can apply one direction for the links within the same type of neurons, the second - the application for connections within the same type of neurons, and the third - for the links with the neighboring level of the hierarchical brain.

In this respect, the analysis consists in determining, for each component of the input signals  $x_i(t)$ , its value when using it later in the synthesis of bonds. With the functioning of the DAN algorithm, nine *RS*-flip-flops, DAN works as a multifunctional elementary automaton of the second kind, which is described by such equations (13.45):

$$\begin{cases} a(t) = \delta_0(a(\Delta - 1), x(t)); \\ a(\Delta) = \delta_e(a(t), e(\Delta)); \\ y_L^2(T) = \lambda_2(a(t), a(\Delta)), \\ a(t), a(\Delta) \in \pi_j; \quad i = 0, 1, 2, \dots; \quad \Delta = 0, 1, 2, \dots \end{cases} \quad (13.45)$$

The input signals  $x_i(t)$  have different pairs of values in the DAN, applied in the ninth *RS* flip-flops. The input signals  $e(\Delta)$  adjust the memorization of two  $a(\Delta-1)$  states of the *RS*-flip-flop in the subsets  $\pi_i$  ( $i = 1, 2, \dots, 9$ ). The subsets  $\pi_i$  determine the output signal  $Y_i(T)$  in the analysis of the DAN operation. Between 9 triggers there are three pairs of states in which the output signals do not intersect. These pairs produce excitatory or inhibitory signals of two values: one unit and zero or two units to-

gether. The significance of these output signals must be analyzed when transmitting their signals.

Thus, the analysis unit should give out signals when determining the disability of neurons in the software for circumventing disabled neurons in the compilation of problem solving algorithms.

### **13.11.3. Typical link libraries**

Typical connections, as we considered, can be of three types:

1. associated with the same type of neurons directly;
2. associated with non-type neurons;
3. associated with neurons through the axon of another level of the brain.

Typical relationships associated with the same type of neurons are directly analogous in their structure to combinatorial circuits without loops.

We agree to call the structure system of neurons a stepwise (hierarchical) system, when all neurons that enter it are divided into pairwise overlapping classes  $K_0, K_1, \dots, K_n$  in such a way that only nodes of class  $K_0$  are synchronous when denoted by input signals of a block of the same type neurons.

For any  $i = 1, 2, \dots, n$  of class  $K_i$ , the neurons take input signals from  $K_0, K_1, \dots, K_{i-1}$ . Neurons of class  $K_i$  are called neurons of the  $i$ -th degree.

In the above multi-stage system of neurons, the input nodes of elementary signals of class  $K_0$  are supplied from the outside. As a result, it turns out that in the correct hierarchical scheme of neurons, the elementary signals at all the nodes of the  $K_i$  classes turn out to be quite definite.

Typical links associated with non-type neurons come through an artificial axon, which is a register on the MUSP. The number of codes in the register of an artificial neuron is  $18^4$ , which determines more than  $10^5$  bonds.

For typical connections that are related to neurons through an axon of another level of the brain, one discharge of the axon register is assigned, which is up to 18 states.

The use of these typical links in the library is possible when synthesizing a system of neurons and the connections between them to solve a particular problem.

In the library, there should also be typical input signals of three-level neurons that determine deterministic, large-scale transitions, probabilistic transitions and probabilistic matrix (fuzzy) transitions. These transitions adjust the transitions in the neurons and their output signals with the corresponding output signals, which is very important in the development of the algorithm for solving the problem.

### **13.12. Conclusion to chapter 13**

In this chapter the following questions are considered:

- The history of research in the field of neural networks and on the basis of these studies a new statement of the problem is made. On the basis of the works of L.F. Marakhovsky to consider the development of various types of artificial neurons, the functions of which would approach the neurons of the human brain, and the development of methods for monitoring the work of the neuron and fixing its exit from the working capacity.
- The notation of multifunctional and multi-level memory schemes is presented.
- The main functional properties of the brain and neuron, artificial neuron models on automatic memory circuits are considered.
- The structural organization of an elementary two-stage neuron on automatic memory circuits is described.
- The conventional designations of the elementary three-stage neuron are presented in automatic memory circuits.
- Descriptions of control schemes for an artificial neuron and the connection between neurons, self-improving artificial neuron systems.
- The structure of axons of an artificial neuron with the ability to self-reconstruct for the organization of connections between neurons is described.
- The structure of the short-term and long-term memory of artificial intelligence and the definition of their functions, the model of the elementary structure of human

existence, the knowledge of a person's real being is limited by the volume and structure of the short-term memory of his brain, the number of hierarchical levels of short-term human memory.

- The structure of neural networks, the structural scheme of an artificial brain on multi-level memory circuits, blocks of hierarchical neurons,

## 14. To the conclusion

# ABOUT NEW SCIENTIFIC DIRECTIONS IN THE FIELD DIGITAL COMPUTER ENGINEERING

### Annotation

The article describes the prospects for the development of computing technology, which are given a clue as to what a new research direction in the field of digital computing, given a brief presentation on the system of the existing knowledge of digital computing in the flip-flop and are defined by their fundamental limitations.

Considering the concept of hierarchical information, the author described the new system of knowledge in the field of computer technology, which allows simultaneous processing of hierarchical information levels together. Given enumeration of features that extend the existing system of knowledge that determine the new scientific direction in the field of computing devices in the multi-functional and multi-level circuitry related to memory.

Keywords: theory of automatic, memory circuits, hierarchical information, neuron, neural networks, the principle of hierarchical software management, pro-software handling hierarchical information.

### 14.1. INTRODUCTION

In order to find the truth, everyone should at least  
once in his life get rid of the concepts  
he has mastered and, completely new,  
build a system of his views

R. Descartes

Any new direction in the field of digital computer technology is a system of knowledge, which can be viewed in terms of its basic criteria.

1. It should always have a theoretical basis, which differs from the previous knowledge and the opportunity to create a new and effective step forward.
2. It must have a distinctive language in which we can discuss the objects of its investigation, the rules by which the observation can be made to bring in a coherent system to show their efficiency in comparison with known science or known trends in science.
3. It must have a method to create new objects with new properties and qualitatively assess their results compared with known similar objects in the field of information systems.
4. It is desirable that this new direction in the field of information systems has been interdisciplinary and would cover such typical areas as: automata theory, methods of construction of the elementary memory circuits, methods of construction of device types and new approaches and methods in building software developed not only deterministic devices but probabilistic.

## 14.2. SYSTEM FOR DIGITAL EXISTING KNOWLEDGE COMPUTER TECHNOLOGY

The system existing knowledge of computer technology has hierarchical interdisciplinary connections described V.M. Glushkov [1–3].

The development of this work was based on a number of fundamental developments such as the theory of pure binary logic, developed by English mathematician John. Boole (1815 - 1864) [4]; the principle of management software, developed by English mathematician Charles Babbage and the engineer (1791–1871) when creating the project mechanical analytical engine, which became the prototype of the modern computer [5].

Viktor Shestakov (1907–1987) - Soviet-logic and theory of tick — American scientist and electrical engineer Claude Shannon (1916 - 2001) in the mid-1930s. An interpretation of the algebra by Bulev of logic shows in the Ladder Diagram [6].

John Atanasoff (1903- 1995) — American physicist, mathematician and electrical engineer of Bulgarian origin, one of the inventors of the first electronic computer, which then was established in the United States on lamps and his ideas have been described by J. Von Neumann in 1945 [7].

Modernization ideas J. Buhl performed. Von Neumann. In the 40-50-ies XX century algebra by Bulev has received special significance in the development of computer technology. J. Von Neumann described the structure of the machine using the trigger as a memory element, and proposed the principle of a stored program [5].

In 1918, Russian scientist Mikhail Bonch-Bruevich made a lamp trigger, capable of maintaining a single binary digit.

This invention laid the foundation for an electronic digital computer. [9] In the future, the United States began to call the trigger Moore automaton with nontrivial memory in honor of the American mathematician, began the study of proper machines [1].

Victor Glushkov (1923-1982) — a pioneer of mathematical theory of computing systems [8]. His theorem on the structural completeness of given rationale for the development of the element base of computer systems.

In this theorem, the main element of Memory offered Moore automaton (trigger) with a non-trivial memory, has a complete system of transitions and outputs a complete system, and any functionally complete set of logical elements (elementary automata without memory), to build arbitrary finite automata [1].

The presentation of the theory of synthesis of digital automata as a unified mathematical theory developed by V.M Glushkov, as he writes, was based on the works of famous

scientists. Among them: D. Hafmen, G. Mealy, VI Shestakov, M. Wilkes, Shannon, George. McCarthy, S.K. Kleene, E. Moore, George. Neumann, D. Aufenkamp, Hong F. et al., V.M. Glushkov and his disciples prepared the theoretical foundation of knowledge on the whole areas of digital computers, such as the theory of digital automata and logical design of discrete devices [1-3]; digital converters [10]; algebra algorithms that serve as a basic knowledge of programming theory [11-12]; theory of data structures and parallel computing [13]; domestic intelligence computers [14-15]; computer architecture [16]; self-organizing systems and artificial intelligence [17-18].

Currently, intensive work is underway to reconfigure the device to process the hierarchical information [21-23] based on the FPGA, as well as work on neural computers [24-26] on the basis of the old element, which is used as a memory trigger or memristor a multi-state, resembling multistable trigger, but these devices have significant fundamental limitations. For example, IBM is the program DARPA SyNAPSE built software-emulated neural network from 5.3 billion neurons and 100 trillion synapses in the computer Lawrence Livermore National Lab (LBNL) Blue Gene / Q Sequoia (1,572,864 processor cores, 1.5 PB memory, 98,304 MPI processes, and 6,291,456 threads), in which information is processed in the automaton discrete time input signals and determined only one variable  $x(t)$ .

In computers and neurocomputers always compare the structure and function of the memory with the memory of living organisms, the memory of which has a better selectivity, higher efficiency and more useful than an immutable structure of the trigger [4-5].

Construction of multi-functional memory circuits (flip-flops) and methods of construction on their basis of computing devices, whose work is considered a discrete automaton time is almost a complete theme [5].

Actual direction lifting restrictions binary memory circuits is the development of multi-functional memory circuits. Research in the field of multi-functional memory circuits are considered by many well-known scientists of the twentieth century. The first thing to highlight the work of the representatives of scientific schools Glushkov [1-3; 28], M.A. Gavrilov [29] A.D. Zakrevskogo [30] E.V. Evreinov and I.V. Prangishvili [31], and many other famous scientists in the world. Trying to use in the construction of Mealy and Moore's memory circuits, which would be carried out one control excitation functions and output functions of binary memory circuits, has not given the expected results. This is due to the fact that the change in the control signals and storing a memory state of the circuit to reconfigure computer systems is performed sequentially, which limits the performance of components of computer systems [32-33].

Performance limitations of modern computer systems and networks are due immutable features of elementary memory circuits (flip-flops), which affect the principles and methods of design theory devices of computers and computer systems.

"The transition from the deductive method that satisfies the existing deterministic systems to inductive methods associated with probability systems requires a different way of thinking. The inertia of the old way of thinking has not yet been overcome. The real obstacle to the development of a new direction is not the complexity of the problems, and conservative people," wrote St. Beer [4].

These interdisciplinary development of theoretical knowledge in the field of modern computers and neuro-computers based and now based on the element base of integrated circuits. This database implements the functions of combinational circuits based on Boolean algebra, and as used in this memory and triggers memristors multi-state, which imposes a fundamental limit on the developed and developing a computer system [19 -20].

The author believes that all the above work on reconfigurable memory circuit conducted by tunable excitation functions and outputs is based RS-trigger. They have fundamental limitations that significantly affect the building of computers and systems, namely:

1. They all operate in the automaton discrete time  $t_i (i = 1, 2, \dots, n, \dots)$ ;
2. The basic memory circuit (*RS*-trigger) does not allow the work to rebuild the memorized states;
3. Describe all these devices Mealy and Moore that define consistent operation of the devices;
4. The transition occurs in the memory circuits for a variable  $x(t)$ ;
5. Used the principle of program management, the proposed C. Babbage, does not allow simultaneous processing of public and private information.

### **14.3. PRESENTATION OF HIERARCHICAL INFORMATION**

Modern computer and neuro-computing systems built on the basis of modern, consistent information is used as input data signals  $x(t)$  and use this information in a discrete automaton time [19-26].

In fact, information is hierarchical and is the third element, that of the universe with matter and motion [34].

In 1948, Claude Shannon (USA) proposed the concept is based on the concept of information as a kind of substance that exists in the real world, regardless of a person. "... In-

formation can be seen as something very similar to a physical quantity such as mass and energy." In the same year in his fundamental work "Cybernetics, or Control and Communication in the Animal and the Machine" Norbert Wiener (1894-1964) identified the information as a "symbol of content obtained from the outside world in the process of our adaptation to it and adapt to our bodies feelings. "Unlike Shannon, he did not consider that the information, matter and energy - it is the category of the same order - "Information is information, not matter or energy". Closest to the understanding of the information came Academician V.M. Glushkov. In its definition of information - a measure of the heterogeneity of the distribution of matter and energy in space and time [35]. It is clearly seen that the concept of the information in those years was not entirely clear.

According to Professor V.K. Promonenkova information — it's all a matter of heterogeneity and processes (form, structure, rhythm, replica). Heterogeneity - a quantitative and qualitative distinctiveness, consisting of a substance means the observer, including your mind. As he spoke in his work, is a hierarchical information ("information about information") and represents the third element of the universe. [34] With this difficult not to agree to the author due to the fact that it creates a trend in the field of digital computer technology, which deals with the storage and processing of hierarchical information, almost forty years [38-48; 52-63].

The author used instead of the input signal  $x(t)$ , supplied to the memory circuit (trigger) computing devices, the input word is  $p(T) = x(t), e(\Delta)$ , consisting of two consecutive signals  $x(t)$  and  $e(\Delta)$ , supplied to the multi-level memory in a single machine cycle  $T$  [47-48]. That difference was enough to have the opportunity and the general handling of personal information at the same time in a single machine cycle  $T$ , which in devices with memory for triggers and memristor principle could not be implemented [19-26].

#### **14.4. NEW SYSTEM OF KNOWLEDGE IN COMPUTER ENGINEERING**

Work on a new direction in the field of computers started in 1977 by the author. In 1986 he reported to work at a seminar of the Institute of modeling problems in power engineering Ukrainian Academy of Sciences, which was approved by Academician Georgy E. Pukhov (1916-1988). On the basis of this work has been deposited with the monographs, which were first set out the principles of a new direction in the field of computer technology [39]. In the 90 years of the twentieth century came two manuals and a monograph on the same subject [40-42].

So what is the new system of knowledge that determine the new scientific direction in the field of computer technology?

First developed by the author and hierarchical abstract multifunctional machines capable of handling hierarchical information (general and private type) simultaneously in a single machine cycle  $T$  and 4 machines of the second kind, which control operation of the elementary memory circuits [39-40; 42-44].

1. Developed multifunctional machines of the 1st, 2nd and 3rd kind that work in the automaton uninterrupted time  $T_i = t_i + \Delta_i$  ( $i = 1, 2, \dots, n, \dots$ ). Monofunctional Automatic Mails and Moore automata operating in discrete time, are a special case of multifunctional machines 1 and 2 of the first kind.

2. Multifunction machines can be seen as deterministic, having two deterministic transition: simple and enlarged, which expands the functional properties of deterministic Mealy and Moore as probabilistic and fuzzy. Multifunction machines are able to make the transition in a matrix structure of states of two variables:  $x(t)$  and  $e(\Delta)$  in one cycle machine time  $T$ .

3. A mathematical model of hierarchical abstract machine with multi-function organizational system memory.

4. Develop machines 4th kind of controlling operation of the elementary memory circuits.

First developed by the author of the theory and analysis microsynthesis multifunctional and multi-level elementary memory circuits [9; 39; 41-43; 45-48].

1. The theory and analysis microsynthesis two classes of multifunctional memory circuits (MFIS). MFIS has a matrix structure storing information and two input variables :  $x(t)$  and  $e(\Delta)$ . MFIS have flexibility, selectivity, improved reliability, reduced cost of logic elements per state and are open structure than and different in a positive direction from an asynchronous RS-latch. The techniques incorporated in this theory, allow the developer according to the criteria and the number of states in the number of re-tunable subsets of states easily design a functional block diagram of the structure and memory.

2. The theory and analysis microsynthesis two classes of multi-level memory circuits (MUSP). MUSP are half-closed structure. MUSP allow to keep the total and private information at the same time, have the flexibility, selectivity, improved reliability, reduced cost of logic elements per state than and different in a positive direction from an asynchronous RS-latch. The techniques incorporated in this theory, allow the developer according to the criteria the number of states or by the number of subsets of states tunable easily design a functional block diagram of the structure and memory.

For the first time the author developed the methods of constructing standard computing devices on MFIS and MUSP [42-43], such as:

1. Reconfigurable registers on MFIS and MUSP.
2. Reconfigurable counters MUSP.
3. Reconfigurable control device.
4. The structure of the reconfigurable computer.

The human brain has several advantages over all the techno-cybernetic devices on such an important, in the opinion of the author, the properties-you.

First, the input signals from the external environment, Air-exist in the eyes, the ears, the body, the taste of food and have multifunctional, a matrix structure.

Secondly, the information that comes from the environment, to compile-is. This is illustrated by the eye. Receptors in the eye of the order of 18-20 million, and cones that summarize the information visible through the eyes of RECEP-tor, about 72 thousand. That is, the data compression occurs at approximately-in 256 times at the second level. This issue information compression is important to understand and solve technical.

Third, you must consider the natural growing links between the neurons of the human brain. The child gradually established connections necessary to generalize the (dis-expandable) information, the construction of relevant templates and fashion-lei, reflecting the real world of the individual person.

Fourth, the human brain has between 14 and 20 billion it-electrons. This is a fairly large structure in the number of neurons that are difficult to physically set up at the present stage of technological development, and the Bo-Lee to manage it.

A talented mathematician Frank Plumpton Ramsey proved that complete disorder is impossible in such large structures as human brain, the universe, etc. Thus, each large enough set of numbers, points, or objects necessarily contains an ordered structure. Work in this direction confirm this important result [64]. However, the problem of creating ordered structures in models of the human brain remains.

Fifth, the brain is not a computer, logical theories, position of number systems, but only a logic of information, data compression, the choice of communication path with other cells, to summarize this information. The calculations, reasoning, number systems, and any other algorithms are derived from those models that have generalized and represent human interest, according to Nikitin A.V. Inta-esting work "control logic cells" [65].

The sixth relates to the structure of the neuron, which has two sets of input signals: excitatory and inhibitory, which triggers and memristoture have, and use only the setting (excitatory) input-WIDE signals  $x(t)$ .

First as neurons and neural networks have been proposed multilevel memory circuits that have two sets of input signal: establishing (excitation)  $x(t)$  and stored (polling)  $e(\Delta)$ . In the area of neurons, neural connections and neural models of architectural ensembles, the author offers the following results, which follow from the proposed new direction [43-44].

Consider the construction of neurons based on MUSP and its characteristics :

1. In the brain neurons have different structures that can be in analog form to create MUSP. In addition, these structures correspond to the laws of nature, which is expressed as the "golden section". This law manifests itself in many structures, such as: people, shells, fish, etc. For example, MUSP, which stores 18 states and consists of 10 gates, is characterized by a number of 1.8, which is close to the number of  $P = 1.618$ .

2. Neuron on MUSP has two sets of input signals: establishing (excitatory)  $x(t)$  and retain (electoral)  $e(\Delta)$ .

3. Describe the function of the neuron in the automaton uninterrupted time.

4. The efficiency of the neuron can be tested controlling machine 4th family.

5. In the event of failure or in the event of non-use in the process, it can be turned off, similar to a biological neuron.

6. Most importantly, a neuron can selectively store information in its memory matrix structure states.

Based on the properties of the three-level memory circuits can be built Register axon, which can selectively connect the output of a neuron, or to one or more neurons in a deterministic, probabilistic or fuzzy modes [44]. This allows you to build models of neural networks, both deterministic and probabilistic and fuzzy.

It also allows in the field of computer software and neuro-computers reconfigured to use microprocessors that are able to change the structure of commands without loss of performance due to the introduction of common code in the address system of commands based on the principle of hierarchical management software [41-42; 60].

At the time, under the guidance of the author described achievements have been applied in specific devices. Theoretical and practical results of these studies are presented in the doctoral thesis of the author [66-67]. Currently, work on the development of this new direction under the guidance of the author continued his post-graduate [52-53; 58-63].

This new knowledge in the field of computer technology led to the following conclusions:

1. All these work in the automaton uninterrupted time  $T_i$  ( $i = 1, 2, \dots, n, \dots$ );

2. The basic memory circuits MFIS and MUSP to tune the work of stateless;

3. To describe all devices with memory for MFIS and machine guns MUSP Marachovsky (multifunction machine guns 1st, 2nd and 3rd kind), which define the nature of the devices reconfigured;

4. The transition occurs in the memory circuits in the two variables  $x(t)$  and  $e(\Delta)$ ;

5. used the principle of hierarchical software management proposed L.F. Marakhovskii allows simultaneous processing of public and private information.

Usually scientific paradigm change among the most dramatic events in the history of science. When discipline is changing one paradigm to another, it is called "scientific revolution" or "paradigm shift." The decision to abandon the paradigm is always at the same time there is a decision to take a different paradigm, and the verdict leading to this decision includes both a comparison of both paradigms with nature and comparing paradigms with each other.

In 1935 Gause established the law, known in biology as the law of competitive exclusion (principle or law of Volterra-Gause). On it are two kinds can not stably exist in the same ecological niche: there is a competitive displacement of one another [49]. The displaced population is under the law gets in conditions different from previous conditions. And here comes into force a law of life will set Chetvertakova in 1905 [50]. Under this law, the size of the population is growing under favorable conditions of existence and plummets under unfavorable. The law is always effective in all populations [50].

## **14.5. CONCLUSION**

Described references to knowledge in interdisciplinary areas of knowledge of the new directions in the field of computer systems allow to raise the level of processing of hierarchical information to a higher level. Favorable efforts to implement these developments, unfortunately, only by leading companies such as Intel, IBM, and the like, or government programs, because the development is necessary to start from scratch.

Firms will allow ahead of its competitors, and the countries - to raise its prestige and economy.

Also, it will make a step forward in the field of computers, and neuro-computers will create competitive devices over existing on the existing integrated circuit technology.

## **15. CONCLUSIONS**

The described references to the obtained knowledge in the interdisciplinary fields of knowledge of a new direction in the field of computer systems will allow to raise the level of processing hierarchical information to a higher level. Favorable efforts to implement these developments, unfortunately, can only be carried out by leading firms such as Intel, IBM and similar or state programs, because development should start from scratch.

Firms will outstrip their competitors, and countries will increase their prestige and economy.

## LIST OF REFERENCES

To chapter 1

1. Baranov S.I. Synthesis of microprogram automata: (graph-schemes and automata). - L.: Energia, 1979. - 232 p.
2. Bukharaev RG Fundamentals of the theory of probability automata. - Moscow: Nauka, 1985. - 288 p.
3. Vavilov EI, Portnoy G.N. Synthesis of electronic digital computer circuits. Moscow: Sov. radio, 1963. - 440 p.
4. Varshavsky V.I. Homogeneous structures: Analysis. Synthesis. Behavior. - Moscow: Energia, 1973.-152 p.
5. Gavrilov MA, Devyatkov VV, Pupyrev EI Logical design of discrete automata. - Moscow: Nauka, 1977. - 352 p.
6. Glushkov V.M. Synthesis of digital automata. - Moscow: Fizmatgiz, 1962. - 476 p.
7. Glushkov V.M. Theory of algorithms. -K.: KVIRTU, 1961. - 167 p.
8. Glushkov VM, Kapitonova Yu.V., Mishchenko AT Logical design of discrete devices. - K.: Science. dumka, 1987. - 264 p.
9. Zade L.A. Fundamentals of a new approach to the analysis of complex systems and decision-making processes // Mathematics today. - M., 1974. - P. 5-49.
10. Zade L.A. The concept of a linguistic variable and its application to the adoption of approximate solutions: Per. with the English .. - M.: Mir, 1976. - 165 p.
11. Kofman A. Introduction to the theory of fuzzy sets: Per. with fr. - M.: Radio and Communication, 1982. - 432 p.
12. Mishchenko VA, Kozyuminsky VD, Semashko AN Multifunctional automata and element base of digital computers / Pod. Ed. V.A. Mishchenko. - Moscow: Radio and Communication, 1981. - 249 p.
13. Multifunctional regular computing structures. Balashov, V.B. Smolov, G.A. Petrov, D.V. Puzankov - Moscow: Sov. radio, 1978.-288 p.
14. Romankevich A.M. Applied theory of digital automata. - K.: Vishcha school, 1987. - 375 p.
15. Handbook of digital computers: (processors and memory) / BN Malinovsky, EI Bryukhovich, EL Denisenko and others // B.N. Malinovsky. - K.: "Техніка", 1979. - 366 p.
16. Chirkov M.K. Fundamentals of the general theory of finite automata. - Leningrad: Leningrad Publishing House. University, 1975. - 280 p.
17. Marakhovsky L.F. Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 c.
18. Marakhovsky L.F. Multilevel devices of automatic memory. I ч. - Kiev: USIM. - № 1.- 1998.- P. 66-72
19. Marakhovsky L.F. Multilevel devices of automatic memory. II ч. - Kiev: Usim. - №2. - 1998. - P. 63-69
20. Marakhovsky L.F. Multifunctional memory circuits. - Kiev: USIM - No. 6.-1996.- P. 59-69

21. Marakhovsky L. F. Foundations of the new information technology. Fundamentals of designing reconfigurable devices of computer systems and artificial neurons: monograph. LF Marakhovsky, NL Mikhno - Germany: Saarbrcken, LAP LAMBERT, 2012. - 347 p.

22. R. L. Graham and V. Rödl. Numbers in Ramsey Theory, in Surveys and in Combinatorics. London Mathematical Society Lecture Notes Series, 1987, No. 2 123, pp.111-153

23. Nikitin A.V. Reading and processing of DNA information. <http://www.trinitas.ru/eng/doc/0016/001c/00161718.htm>

24. Ivanov Vyach. Sun. Even and odd. Asymmetry of the brain of sign systems. Moscow: Sov. radio, 1978.-C.62. 25. Ivanov-Muromsky, K.A. Brain and memory. -K.: Nauk. dumka, 1987.-136 p.

To chapter 2

1. Reference book on digital computers: (processors and memory) / BN Malinovsky, EI Bryukhovich, EL Denisenko and others / Ed. BN Malinovsky. - K.: "Техніка", 1979. - 366 p.

2. V.Tatur, 15th anniversary of the SKIF project: history and results // Academy of Trinitarianism, M., El No. 77-6567, pub. 18063, 09.06.2013

3. A.S. Cassidy, P. Merolla, J.V. Arthur, S. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T.M. Wong, V. Feldman, A. Amir, D. Rubin, F. Akopyan, E. McQuinn, W. Risk, and D.S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in the International Joint Conference on Neural Networks (IJCNN). IEEE, 2013.

4. Marakhovsky L. F. Foundations of the new information technology. Fundamentals of designing reconfigurable devices of computer systems and artificial neurons: monograph. / LF Marakhovsky, NL Mikhno - Germany: Saarbrcken, LAP LAMBERT, 2012. - 347 p.

5. Marachovsky L.F. Basic Concepts to Build the Next Generation of Reconfigurable Computing Systems - International Journal of Applied And Fundamental Research. - 2013. - No. 2 - URL: [www.science-sd.com/455-24170](http://www.science-sd.com/455-24170) (20.11.2013). - 6p.

6. Marakhovsky LF Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.

7. Zade L.A. Fundamentals of a new approach to the analysis of complex systems and decision-making processes// Mathematics today. - M., 1974. - P. 5-49.

8. Zade L.A. The concept of a linguistic variable and its application to the adoption of approximate solutions: Per. with the English. - M.: Mir, 1976. - 165 p.

9. Kofman A. Introduction to the theory of fuzzy sets: Per. with fr. - M.: Radio and Communication, 1982. - 432 p.

10. Bukharaev RG. Fundamentals of the theory of probability automata. - Moscow: Nauka, 1985. - 288 p.

11. Marakhovsky L.F. Finite state machines with a multifunctional memory organization system: Teaching. allowance. -K.: UMK VO, 1991. - 67 p.

12. R. L. Graham and V. Rödl. Numbers in Ramsey Theory, in Surveys and in Combinatorics. London Mathematical Society Lecture Notes Series, 1987, No. 2 123, pp. 111-153.

To Chapter 3

1. Glushkov V.M. Synthesis of digital automata. - Moscow: Fizmatgiz, 1962. - 476 p.

2. Marakhovsky L.F. Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.

3. Reference book on digital computers: (processors and memory) / BN Malinovsky, EI Bryukhovich, EL Denisenko and others / Ed. BN Malinovsky. - K. : "Техніка", 1979. - 366 p.

4. Glushkov V.M, Kapitonova Yu.V., Mishchenko AT Logical design of discrete devices. - K.: Science. dumka, 1987. - 264 p.

5. Marakhovsky L.F, Moskvina M.V., Moskvina V.V. Analiz pratsездatnosti at kotostrafichnih vidmovah at base schemes пам'яти // IX Міжнародна науково-технічна conference «Гіротехнології, навігація, керування рухом і конструювання авіаційно-космічної техніки»: Збірка доповідів. - K. : NTU «КПІ», 2013.- P. 462-468.

6. Baz G. A., Samokhvalov E.A. Fundamentals of building nodes of electronic computers: Proc. allowance. K. : KINH, 1978. - 109 p.

7. Baranov S.I. Synthesis of microprogram automata: (graphs and automata). - L. : Energia, 1979. - 232 p.

8. Vavilov EI, Portnoy G.N. Synthesis of electronic digital computer circuits. Moscow: Sov. radio, 1963. - 440 p.

9. Gavrilov M.A., Devyatkov V.V., Pupyrev E.I.. Logical design of discrete automata. - Moscow: Nauka, 1977. - 352 p.

10. Evreinov E.V., Prangishvili I.V. Digital automata with a tunable structure (homogeneous media). - Moscow: Energia, 1974. -240 p.

11. Mishchenko VA, Kozyuminsky V.D., Semashko A.N. Multifunctional automata and element base of digital computers / Pod. Ed. V.A. Mishchenko. - Moscow: Radio and Communication, 1981. - 249 p.

12. Romankevich A.M. Applied theory of digital automata. - K. : Vishcha school, 1987. - 375 p.

13. Chirkov M.K. Fundamentals of the general theory of finite automata. - Leningrad: Leningrad Publishing House. University, 1975. - 280 p.

14. Bukharaev RG Fundamentals of the theory of probability automata. - Moscow: Nauka, 1985. - 288 p.

15. Kovalenko A.E., Gula V.V. Fault-tolerant microprocessor systems. - K. : Technique, 1986. - 150 p.

16. Kozlov B.A., Ushakov I.A. Handbook on the calculation of reliability of radio electronics and automation equipment. - Moscow: Sov. Radio, 1975. - 472 p.

17. Avizhenis A. Fault-Tolerance - A Property That Ensures the Continuous Performance of Digital Systems // Proc. Institute of Electrical and Electronics Engineers. - 1978. -T. 66.-No. 10.-P. 5-15.

18. Stakhov A.P. On the possible cause of increased accidents in the withdrawal of Russian satellites. // Academy of Trinitarianism, M., El No. 77-6567, publ.17146, 26.12.2011

19. Khetagurov Ya.A. Ensuring the national security of real-time systems. - Moscow: BC / NW 2009; №2 (15): 11.1

To Chapter 4.

1. Marakhovsky L.F. Expansion of the fundamental foundations of the modern elemental base of computer systems.-// "Academy of Trinitarianism", M., El No. 77-6567, pub. 2006, 26.01.2015.

2. Marakhovsky L.F. Multifunctional memory circuits. - Kiev: USIM - No. 6.- 1996.- P. 59-69.

3. The fundamentals of the new information technology. Fundamental foundations of building reconfigurable devices of computer systems and artificial neuron. - Saarbrcken, Germany i.melnic@lap-publishing.ru / www.lap-publishing.ru. - 2013.-369 p.

4. Bukreev I.N., Mansurov V.M., Goryachev V.I. Microelectronic circuits of digital devices. - Moscow: Sov. radio, 1975. - 368 p.

5. Handbook of digital computers: (processors and memory) / B,N, Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others / Ed. BN Malinovsky. - K.: "Техніка", 1979. - 366 p.

6. Earl of the RF Encyclopedia of electronic circuits / R.F. The earl, V. Shiits. - Moscow: DMK, 2007. - 249 p.

7. Marakhovsky L.F. Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.

8. Marakhovsky L.F. Defect-free design of functional circuits by means of mathematical modeling (in the ternary calculus:  $0,1, \alpha$ ) on the computer. / Sat: "Problems of the reliability of control systems," Naukova Dumka, Kiev, 1973, p. 66-69.

9. Marakhovsky L.F., Mikhno N.L.. Definition of input words of elementary multifunctional circuits of automaton memory. // Zbirnik naukovykh prats DETUT, Seriya "Transportni sistemi i tehnologii", 2009. -Vip. 14 - pp. 139-151.

10. Avizhenis A. Fault Tolerance - A Property That Ensures the Continuous Operability of Digital Systems // Proc. Institute of Electrical and Electronics Engineers. - 1978. -T. 66.-No. 10.-P. 5-15.

11. Kovalenko A.E., Gula V.V. Fault-tolerant microprocessor systems. - K.: Technique, 1986. - 150 p.

12. Stakhov A.P. On the Possible Cause of Frequent Accidents in the Release of Russian Satellites. // Academy of Trinitarianism, M., El No. 77-6567, publ.17146, 26.12.2011

13. Khetagurov Ya.A. Ensuring the national security of real-time systems. - Moscow: BC / NW 2009; №2 (15): 11.1
14. Gagin V. System analysis "The blade of life." - Odessa: El. ed., 2001. - 315 p.
15. Evreinov EV, Prangishvili I.V. Digital automatic machines with an adjustable structure (homogeneous media). - Moscow: Energia, 1974. -240 p.
16. Mishchenko V.A., Kozyuminsky V.D., Semashko A.N. Multifunctional automata and element base of digital computers / Pod. Ed. V.A. Mishchenko. - Moscow: Radio and Communication, 1981. - 249 p.
- To chapter 5
1. Marakhovsky L.F. Multilevel devices of automatic memory. I ч. - Kiev: USIM. - No. 1.- 1998.- P. 66-72
2. Marakhovsky L.F. Multilevel devices of automatic memory. II ч. - Kiev: Usim. - №2. - 1998. - P. 63-69
3. Marakhovsky L.F. Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.
4. Bukreev I.N., Mansurov V.M., Goryachev V.I. Microelectronic circuits of digital devices. - Moscow: Sov. radio, 1975. - 368 p.
5. Handbook of digital computers: (processors and memory) / BN Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others / Ed. B.N. Malinovsky. - K.: "Техніка", 1979. - 366 p.
6. Glushkov V.M. Synthesis of digital automata. - Moscow: Fizmatgiz, 1962. - 476 p.
7. Glushkov V.M., Kapitonova Yu.V., Mishchenko A.T. Logical design of discrete devices. - K.: Science. dumka, 1987. - 264 p.
8. Kalyaev I.A., Levin I.I., Semernikov E.A. Reconfigured multi-convolutional computing structures M.: JRC RAS, 2008. - 395 p.
9. Karlashchuk V.I. Electronic laboratory on IBM PC. T. 2. Modeling of elements of telecommunication and digital systems. 6 th ed., Revised. and add. - M.: SOLON-PRESS, 2006. -640 p.
10. Computer schematics (short course) / RO. Protsyuk, V.N. Korneichuk, P.V. Kuzmenko, V.P. Tarasenko. - K.: PP "Korniychuk", 2006. - 433 p.
11. Palagin A.V. Reconfigurable Computing Systems: Fundamentals and Applications / A.V. Palagin, V.N. Opanasenko. - K.: Просвіта, 2006. - 280 p.
12. Stepanov A.N. Architecture of computer systems and computer networks. - St. Petersburg: Peter, 2007. -509 p.
13. IBM is working on the creation of a "computer brain" <http://www.cybersecurity.ru/it/82336.html>
14. Marakhovsky L.F., Voevodin S.V., Mikhno N.L., Sharapov O.D. Comp'yutera circuit: a workshop for bachelor's specialties. "Інтелектуальні системи прийняття рішень". - Київ: КНЕУ, 2009. -245 p.

15. Avizhenis A. Fault tolerance - a property that ensures the permanent availability of digital systems // Proc. Institute of Electrical and Electronics Engineers. - 1978. -T. 66.-No. 10.-C. 5-15.

To Chapter 6

1. Glushkov V.M. Synthesis of digital automata. - Moscow: Fizmatgiz, 1962. - 476 p.

2. Povoroznyuk A.I. Architecture of computers: Training. Help. / National. tech. University of Kharkiv Polytechnic Institute. - H.: Tornado, 2004. - 355 p.

3. V.Tutur, 15th anniversary of the SKIF project: history and results // Academy of Trinitarianism, M., El No. 77-6567, pub. 18063, 09.06.2013.- <http://www.trinitas.ru/rus/doc/0023/001a/00231048.htm>

4. Problems of the construction of cybernetic systems: Sat. sci. Tr. / Academy of Sciences of Ukraine Institute of Cybernetics. V.M. Glushkova, Science. Council of the Ukrainian Academy of Sciences on the problem of "Cybernetics"; Editorial Board: VVPavlov Ed. and others - K., 1993. -70 p.

5. The Earl of RF. Encyclopedia of electronic circuits / R.F. The earl, V. Shiits. - Moscow: DMK, 2007. - 249 p.

6. Tatour V.Yu., Microelectronics of Alexander Taran // "Academy of Trinitarianism", M., El No. 77-6567, pub. 18107, 07.20.2013. - <http://www.trinitas.ru/eng/doc/0234/001a/02341044.htm>

7. Nikitin A.V., A little about the memristor ... // "Academy of Trinitarianism", M., El No. 77-6567, publ.19539, September 12, 2014

8. Marakhovsky L.F., Chechik A.L. and others. The ways of cognition of the regularities of the evolution of complex systems (Search and evaluation of the choice of effective solutions and automata of the third kind): a collective monograph. - Odessa: LLC "Institute of Creative Technologies", 2012. -282 p.

9. Evdokimov V.F. Stasyuk AI, Shcherbakov V.I. Matrix computing devices: Algorithms and structures. - K.: Science. dumka. 1993. -151 p.

10. Evdokimov V.F. Stasyuk AI, Shcherbakov V.I. Matrix computing devices: Algorithms and structures. - K.: Science. dumka. 1993. -151 p.

11. Evreinov E.V., Prangishvili I.V. Digital automata with a tunable structure (homogeneous media). - Moscow: Energia, 1974. -240 p.

12. Kalyaev I.A., I.I. Levin II, Semernikov EA Reconfigured multi-convolutional computing structures M.: JRC RAS, 2008. - 395 p.

13. Mishchenko V.A., Kozyuminsky V.D., Semashko A.N. Multifunctional automata and element base of digital computers / Pod. Ed. V.A. Mishchenko. - Moscow: Radio and Communication, 1981. - 249 p.

14. Multifunctional regular computing structures. Balashov, V.B. Smolov, G.A. Petrov, D.V. Puzankov - Moscow: Sov. radio, 1978.-288 p.

15. Overview of machine design of LSI, logical synthesis and parallel computing. - Moscow: EI VT, 1989. - P. 20-22.

16. Palagin A.V. Reconfigurable Computing Systems: Fundamentals and Applications / A.V. Palagin, V.N. Opanasenko. - K.: Просвіта, 2006. - 280 p.

17. Palagin AV, Opanasenko VN, Sakharin V.G. Digital devices on FPGA type FPGA // modeling. - 2000. - №2. - P. 21-33.
18. Marakhovsky L. F. Discrete devices with a multifunctional organization of memory // Kiev Institute of National Economy. - K. : 1987.-244 p. Dep. in Ukr NIINTI. 30.12.87. №3346-Ук87.
19. Marakhovsky L.F. The concept of constructing parallel computer systems: from circuits of automata memory to polygraph devices // Pricy of the international symposium on the development of the first EOM and the introduction of european technologies into the development of computer technologies - K. : "Phoenix" UAINP, 1998. - P. 274-281.
20. Marakhovsky L.F. Multilevel devices of automatic memory. I ч. - Kiev: USIM. - No. 1.-1998.- P. 66-72
21. Marakhovsky L.F. Multilevel devices of automatic memory. II ч. - Kiev: USIM. - №2. - 1998. - P. 63-69
22. Marakhovsky L.F. Multifunctional memory circuits. - Kiev: USIM - No. 6.-1996.- P. 59-69
23. Marakhovsky L.F. Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.
24. Marakhovsky L.F. Devices of computers with a multifunctional system of memory organization: Proc. allowance - K. : UMK VO, 1996. - 56 p.
25. Marakhovsky L.F., Voevodin S.V., Mikhno N.L., Sharapov O.D. Comp'yutera circuit: a workshop for bachelor's specialties. "Intelligent decision-making systems". - Kiev: KNEU,2009. -245 p.
26. Marakhovsky L.F., Mikhno N.L. The electronic machine is calculated. - The patent. It is registered in the State Register of the Patent of Ukraine on mercenary models - No. 34167 of the 25th lip of the year 2008 p. - (51) IPC (2006) G06F 17/00 - Bul. 14. - 10 seconds.
27. Marakhovsky L.F., Mikhno N.L. Elementarny bahatofunktsionalni schemes of automatized memory. // Collection of scientific papers GETUT, Series "Transport systems and technologies", 2008, Issue.13. - P. 229-241
28. Marakhovsky L.F., Mikhno N.L. Microprograms of attachments of cherry. - The patent. Registered in the State Register of Patents of Ukraine on the No. 87871 of the date 28. 08.2009 p. - (51) IPC (2009) G06F 9/00 - Bul. 16. - 6 p.
29. Marakhovsky L.F., Mikhno N.L. Symbolic language and method of microstructural synthesis of multifunctional memory circuits. // 3b. sciences. NRATI DETHY. Series "Transport systems and technologies".- K. : DETUT, 2010. - Bul. 16. - P. 178-185.
30. Marakhovsky L.F., Mikhno N.L. Structural automaton. - Patent-approved in the Derzhavnoe reestri of the patent of Ukraine on the korisni model № 25816 dated 27 August 2007 - (51) IPC (2006) G06F 1/00 - Bul. 13.- 12 p.
31. Marakhovsky L.F., Mikhno N.L. Scheme of memory. - The patent. - Restated in the Derzhavnoe reestri of the patent of Ukraine on the corridor model No. 29581 dated 25 December 2008. - (51) IPC (2006) G05B 11/42 -Bul. 2. - 14 p.

32. Marakhovsky L.F., Mikhno N.L. Scheme of memory. - The patent. - Established in the Derzhavnoe reestri of the patent of Ukraine on the corridor model No. 29582 dated 25 December 2008. . - (51) IPC (2006) G05B 11/42 -Bul. 2. -10 p.

33. Marakhovsky L.F., Mikhno N.L. The theory of constructing potential elementary circuits of automata memory. - M .: "Academia of Trinitarianism", EINII77-6567, pool.14508. 16.07.07. - 19 p.

34. Marakhovsky L.F., Gavrilenko V.V., Mikhno N.L. Mathematical foundations of digital automata in the third genus. // News of the National Transport University. - Part 2. - K .: NTU.- Issue 17, 2008. – P. 329-335.

35. Marakhovsky L.F., Gavrilenko V.V., Zaitsev O.V., Mikhno N.L. Structural synthesis of automata for a one-hour section of the sun and an environment / Information Science // News of the National University of Biology and Nature Conservation of Ukraine. - K., 2009 p. - release 139. - P. 114-120

36. Marakhovsky L.F., Mikhno N.L., Pogrebnyak V.D. Scheme of memory. - The patent. It is registered in the State Register of the Patent of Ukraine on mercenary models - No. 34166 dated 25 lipnya 2008 p. - (51) IPC (2006) H03K 29/00 - Bul. 14. -12 p.

37. Mikhno N.L. Ways to build a reconfigurable processor at the "elemental" level / Zbirnik naukovykh prac DETUT, Seriya "Transportni sistemi i tehnologii", 2011, Vip. 18. - P. 84-94

38. Mikhno N.L. Ways to build reconfigurable computer systems on automaton memory elements / Zbirnik naukovykh prac DETUT, Seriya "Transportni sistemi i tehnologii", 2011, Vip. 19. - P. 146-152.

39. Reference book on digital computers: (processors and memory) / B.N. Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others / Ed. B.N. Malinovsky. - K .: "Equipment ", 1979. - 366 p.

To Chapter 7

1. Glushkov V,M, Synthesis of digital automata. - Moscow: Fizmatgiz, 1962. - 476 p.

2. Marakhovsky L.F. Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.

3. Reference book on digital computers: (processors and memory) / B.N. Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others / Ed. BN Malinovsky. - K .: "Equipment ", 1979. - 366 p.

To Chapter 8

1. Hasson C. Microprogram control. – Translation from English. Ed. VC. Levin - Moscow: ed. "WORLD", 1973. - 240 p. (issue 1) and 477 p. (issue 2).

2. Reference book on digital computers: (processors and memory) / B.N. Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others / Ed. BN Malinovsky. - K .: "Equipment ", 1979. - 366 p.

3. Marakhovsky L.F., Mikhno N.L. Microprograms of attachments keryvannya. - Patent. Registered in the State Register of Patents of Ukraine on the № 87871 of the date 28. 08.2009 p. - (51) IPC (2009) G06F 9/00 - Bul. 16.- 6 p.

To chapter 9

1. Handbook of digital computers: (processors and memory) / B.N. Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others / Ed. B.N. Malinovsky. - K.: "Equipment", 1979. - 366 p.

2. Marakhovsky L.F. Multilevel devices of automatic memory. I ч. - Kiev: USIM. - No. 1.- 1998.- P. 66-72

3. Marakhovsky L.F. Multilevel devices of automatic memory. II ч. - Kiev: USIM. - No. 2. - 1998. - P. 63-69

To Chapter 10

1. Glushkov V.M. Theory of algorithms. -K.: KVIRTU, 1961. - 167 p.

2. Handbook of digital computers: (processors and memory) / B.N. Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others / Ed. B.N. Malinovsky. - K.: "Equipment", 1979. - 366 p.

3. Marakhovsky L.F. Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.

4. Marakhovsky L.F., Mikhno N.L. The electronic machine is calculated. - The patent. Registered in the State Patent Register of Ukraine for self-serving models No. 34167 of the 25th lip of the year 2008 p. - (51) IPC (2006) G06F 17/00 - Bul. 14. - 10 p.

5. Marakhovsky L.F., Mikhno N.L. Microprograms of attachments of cherry. - The patent. Registered in the State Patent Register of Ukraine for self-serving models on the No. 87871 of the date 28. 08.2009 p. - (51) IPC (2009) G06F 9/00 - Bul. 16.- 6 p.

6. Marakhovsky L.F., Mikhno N.L. Structural automaton. - Patent-approved in the Derzhavnoe reestri of the patent of Ukraine on the korisni model No 25816 dated 27 August 2007 p. - (51) IPC (2006) G06F 1/00 - Bul. 13.- 12 p.

7. Marakhovsky L.F., Mikhno N.L. Scheme of memory. - The patent. - Restated in the Derzhavnoe reestri of the patent of Ukraine on the corridor model No. 29581 dated 25 December 2008. - (51) IPC (2006) G05B 11/42 -Bul. 2. - 14 p.

8. Marakhovsky L.F., Mikhno N.L. Scheme of memory. - The patent. - Established in the Derzhavnoe reestri of patents of Ukraine on the corridor model No. 29582 dated 25 December 2008. - (51) IPC (2006) G05B 11/42 -Bul. 2. - 10 p.

9. Marakhovsky L.F., Mikhno N.L., Pogrebnyak V.D. Scheme of memory. - The patent. Зареєстровано в Держаному реєстрі Patentів України на корсні моделі No. 34166 dated 25 lipnya 2008 p. - (51) IPC (2006) H03K 29/00 - Bul. 14. -12 p.

10. Murakhovsky V.I. Iron P.C. New opportunities. - St. Petersburg: Peter 2005. 592 p.

11. Glushkov VM, Kapitonova Yu.V., Mishchenko AT Logical design of discrete devices. - K.: Science. dumka, 1987. - 264 p.

12. Mikhno N.L. Ways to build a reconfigurable processor at the "elemental" level / Zbirnik naukovych prac DETUT, Seriya "Tran-sportni sistemi i tehnologii", 2011, Vip. 18. - P. 84-94.

13. Hasson C. Microprogram control. - Moscow: Mir, 1973. - Issue. 1 - 240 s. ; 1974. - Issue. 2. - 477 p.

14. Mikhno N.L. Ways to build reconfigurable computer systems on automaton memory elements / Zbirnik naukovykh prac DETUT, Seriya "Transportni sistemi i tehnologii", 2011, Vip. 19. - P. 146-152.

15. Marakhovsky L.F. The concept of constructing parallel computer systems: from circuits of automata memory to polygraph devices // Prits of the international symposium on the development of the first EOM and the introduction of European technologies into the development of computer technologies - K. : "Phoenix" UAINP, 1998. - P. 274-281.

16. Marakhovsky LF Multilevel devices of automatic memory. I ч. - Kiev: USIM. - № 1.- 1998.- P. 66-72

17. Marakhovsky L.F. Multilevel devices of automatic memory. II ч. - Kiev: Usim. - №2. - 1998. - P. 63-69 18. Marakhovsky LF. Multifunctional memory circuits. - Kiev: USIM - No. 6.-1996.- P. 59-69.

To chapter 11

1. Lenat D.B. Artificial Intelligence. Sat. scientific. Pop. articles. Trans. with English. Under. ed VM Kurochkina. - Moscow: Mir, 1986. - P. 174-186.

2. G.S. Pospelov Artificial intelligence is the basis of new information technology. - M., "Science", 1988. -279 p.

3. Pospelov D.A., Stefanyuk V.L. Artificial intelligence in foreign studies. // Information materials 3 of the Scientific Council on the Complex Problem "Cybernetics", 1986.-P. 3-33.

4. Pospelov D.A. Fantasy or science: on the way to artificial intelligence. - Moscow: Science. Ch. Ed. fizmatlit, 1982. -224 p.

5. Principles of self-organization. Trans. with English. Ed. and with a preface by Dr.Sc. AND I. Lerner. - M., Izd. "The World", 1966. -621 p.

6. Rabinovich. Z.L. Fundamentals of the theory of elemental computer structures. - 2 nd ed., Pererab. and additional. - Moscow: Radio and Communication, 1982. - 279 p.

7. Rabinovich. Z.L. On the problem of intellectualization of computers: a little history and some perspectives. // Pratsi міжнародного симпозиуму з історії securing the first EOM that introduces europeans into the development of computer technology - K. : "Phoenix" УАИИП, 1998. - P. 169-173.

8. Rabinovich. Z.L. On the concept of machine intelligence and its development. // Cybernetics and system analysis. – 1995. - №2. - P. 163-173.

9. Realities and predictions of artificial intelligence Per.s English. Ed. V.L. Stefanyuka. - Moscow: Mir, 1987. - 254 p.

10. IBM is trying to imitate the human brain [http://www.pcwork.ru/ibm\\_pyitayutsya\\_imitirovat\\_chelovecheskiy\\_mozg.htm](http://www.pcwork.ru/ibm_pyitayutsya_imitirovat_chelovecheskiy_mozg.htm)

11. IBM is working on creating a "computer brain" <http://www.cybersecurity.ru/it/82336.html>.

12. Khursin L.A. The beginning of the theory of systems of a social type. - K. : Svit, 2001. - 279 p.

13. Marakhovsky LF Multilevel devices of automatic memory. I ч. - Kiev: USIM. - No. 1.- 1998.- P. 66-72.
14. Marakhovsky LF. Multilevel devices of automatic memory. II ч. - Kiev: Usim. - №2. - 1998. - P. 63-69
15. Marakhovsky LF. Multifunctional memory circuits. - Kiev: USIM - No. 6.- 1996.- P. 59-69.
16. Zade L.A. Fundamentals of a new approach to the analysis of complex systems and decision-making processes// Mathematics today. - M., 1974. - P. 5-49.
17. Zade L.A. The concept of a linguistic variable and its application to the adoption of approximate solutions: Per. with the English .. - M.: Mir, 1976. - 165 p.
18. Kofman A. Introduction to the theory of fuzzy sets: Per. with fr. - Moscow: Radio and Communication, 1982. - 432 p.
19. R. L. Graham and V. Rödl. Numbers in Ramsey Theory, in Surveys and in Combinatorics. London Mathematical Society Lecture Notes Series, 1987, No. 2 123, P.111-153.
20. Nikitin AV, Logic of control of the cell // "Academy of Trinitarianism", M., El No. 77-6567, pub. 1707, 29.11.2011.
21. Towards a cognitive computer [http://itc.ua/articles/na\\_puti\\_k\\_sozdaniyu\\_kognitivnogo\\_kompyutera\\_43475?Page=122](http://itc.ua/articles/na_puti_k_sozdaniyu_kognitivnogo_kompyutera_43475?Page=122). IBM is trying to imitate the human brain [http://www.pcwork.ru/ibm\\_pyitayutsya\\_imitirovat\\_chelovecheskiy\\_mozg.htm](http://www.pcwork.ru/ibm_pyitayutsya_imitirovat_chelovecheskiy_mozg.htm) 23. IBM is working on a the creation of a "computer brain" <http://www.cybersecurity.ru/it/82336.html>

To chapter 12

1. Paul Merolla<sup>1</sup>, John Arthur<sup>1</sup>, Filipp Akopyan<sup>1,2</sup>, Nabil Imam<sup>2</sup>, Rajit Manohar<sup>2</sup>, Dharmendra S. Modha<sup>1</sup>. A Digital Neurosynaptic Core Using Embedded Crossbar Memory with 45pJ per Spike in 45nm. –IBM Research - Almaden, <sup>2</sup>Cornell University, 2014
2. Thieves H. IBM created a chip with artificial neurons. - 11 August [http://json.tv/tech\\_trend\\_find/ibm-sozdala-chip-s-iskusstvennymi-neyronami](http://json.tv/tech_trend_find/ibm-sozdala-chip-s-iskusstvennymi-neyronami) On materials [arstechnica.com](http://arstechnica.com)
3. Nikitin AV, Somewhere on the way to understanding ... // "Academy of Trinitarianism", Moscow, El № 77-6567, pub. 18092, 07.07.2013
4. Nikitin AV, Information in DNA, RNA and Proteins // "Academy of Trinitarianism", Moscow, El No. 77-6567, Moscow, Publ.16495, April 23, 2011
5. Marakhovsky LF Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.
6. Marachovsky L.F. Basic Concepts to Build the Next Generation of Reconfigurable Computing Systems. - International Journal of Applied And Fundamental Research. - 2013. - No. 2 - URL: [www.science-sd.com/455-24170](http://www.science-sd.com/455-24170) (20.11.2013).
7. Marakhovsky L.F. Fundamentals of the theory of the synthesis of digital devices on automatic memory circuits: monograph.- K.: GETUT, 2014. - 278 p. //

Interdisciplinary research in science and education. - 2014. - Book; URL: [www.es.rae.ru/mino/168-1400](http://www.es.rae.ru/mino/168-1400) (reference date: 02/02/2014).

To chapter 13

1. Wilkes Mauris. Early digital computer developments in England || Proceedings of the international symposium on the history of the creation of the first computers and the contribution of Europeans to the development of computer technologies. Computers in Europe, the past, the present and the future. - Kiev: "Phoenix" UANP, 1988. – c. 13-16.
2. Malinovsky B.N. Computer pioneers of CIS computerizes. || Proceedings of the international symposium on the history of the creation of the first computers and the contribution of Europeans to the development of computer technologies. Computers in Europe, the past, the present and the future. - Kiev: "Phoenix" UANP, 1988. – c. 27–30.
3. McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115—133.
4. Wiener, N. (1965). Perspectives in Neurocybernetics. *Progress in Brain Research*, 17, 399—404.
5. Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley and Sons.
6. Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, New Series, 59 (236), 433—460.
7. Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungs problem. *Proceedings of the London Mathematical Society*, 42, 230—265.
8. Willshaw, D. J., Buneman, O. P., & Longuet-Higgins, H. C. (1969). Non-holographic associative memory. *Nature*, 222, 960—962.
9. von Neumann, J. (1958). *The Computer and the Brain*. Yale University Press, 1958.
10. Holland, O. E. (1997). Grey Walter: The Pioneer of Real Artificial Life. *Proceedings of the 5th International Workshop on Artificial Life*, MIT Press, Cambridge, 34—44.
11. White, H. (1992). *Artificial Neural Networks: Approximation and Learning Theory*. Cambridge, MA: Blackwell, 224—258.
12. Rosenblatt, F. (1962). *Principles of Neurodynamics*. Washington, DC: Spartan Books.
13. Minsky, M. L., & Papert, S. A. (1969). *Perceptrons*. Cambridge, MA: MIT Press.
14. Galushkin, A. I. (1974). Sintez mnogoslounnikh sistem raspoznavaniya obrazov. Moskva: Energiya [in Russian].
15. Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Ph.D. thesis, Cambridge, MA: Harvard University.
16. Ivakhnenko, A. H. (1968). Metod hrupovogo ucheta arhumentov —

konkurent metoda stokhasticheskoy approksymaciyi. *Avtomatika (Automatics)*, 3, 58—72 [in Russian].

17. Chua, L. O. (1971). Memristor — The Missing Circuit Element. *IEEE Transactions on Circuits Theory (IEEE)*, 18 (5), 507—519.
18. Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, 8, 338—353.
19. Mamdani, E. H. (1976). Advances in the Linguistic Synthesis of Fuzzy Controller. *International Journal Man-Machine Studies*, 8, 669—678.
20. Mamdani, E. H., & Assilian, S. (1975). An Experiment in Linguistic Synthesis with Fuzzy Logic Controller. *International Journal Man-Machine Studies*, 7(1), 1—13.
21. Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15, 116—132.
22. Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biol. Cybernetics*, 20, 121—136.
23. Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36 (4), 93—202.
24. Grossberg, S. (1976). Adaptive pattern classification and universal recoding, part I. Parallel development and coding of neural feature detectors. *Biol. Cybernet.*, 23, 121—134.
25. Willshaw, D.J., & Von der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization. *Proc. R. Soc. London B.*, 194, 431—445.
26. Grossberg, S. (1980). How does the Brain Build a Cognitive Code? *Psychological Review*, 87, 1—51.
27. Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59—69.
28. Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 79, 2554—2558.
29. Coppin, B. (2004). *Artificial Intelligence Illuminated*. Jones & Bartlett Learning.
30. Khokins, J., & Bleksli, S. (2007). *Ob intielliektie: Pier. s angl. Moskva: OOO "I.D. Viliams"* [In Russian].
31. Rummelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Internal Representation by Back-Propagation Errors. *Nature*, 23, 533—536.
32. Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A Learning Algorithm for Boltzmann Machines. *Cognitive Science*, 9 (1), 147—169.
33. Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2, 321—355.
34. Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20, 273-297.
35. Maass, W. (1997). Networks of spiking neurons: the third generation

- of neural network models. *Neural Networks*, 10 (9), 1659—1671.
36. Izhikevich, E. M. (2004). Which Model to Use for Cortical Spiking Neurons? *IEEE transactions on neural networks*, 15 (5).
  37. Furber, S., & Temple, S. (2007). Neural systems engineering. *J. R. Soc. Interface*, 4, 193—206.
  38. Strukov, D. B., Snider, G. S., Stewart, D. R., & Williams, R. S. (2008). The missing memristor found. *Nature*, 453, 80—83.
  39. Beckett, J. (2008). Demystifying the memristor: Proof of fourth basic circuit element could transform computing. *HPL.HP.COM*. Retrieved May 10, 2014, from <http://www.hpl.hp.com/news/2008/apr-jun/memristor.html>.
  40. SyNAPSE: IBM Cognitive Computing Project. *Watson.ibm.com*. Retrieved May 9, 2014, from <http://researchweb.watson.ibm.com/cognitive-computing>.
  41. HRL Laboratories, LLC. Center for Neural and Emergent Systems. *Hrl.com*. Retrieved April 19, 2014, from [http://www.hrl.com/laboratories/cnes/cnes\\_main.html](http://www.hrl.com/laboratories/cnes/cnes_main.html).
  42. Modha, D. S. (2011). Cognitive Computing: Neuroscience, Supercomputing, Nanotechnology. DAC 2011. Lecture conducted from IBM Research, Almaden, San Jose, CA. Retrieved May 3, 2014, from <http://www2.dac.com/events/videoarchive.aspx?confid=122&filter=keynote&id=122-120--0&#video>.
  43. IBM Research: Neurosynaptic chips. *Watson.ibm.com*. Retrieved May 11, 2014, from <http://researchweb.watson.ibm.com/cognitive-computing/neurosynaptic-chips.shtml>.
  44. Marachovsky L.F. Basic Concepts to Build the Next Generation of Reconfigurable Computing Systems. // *International Journal Of Applied And Fundamental Research*. – 2013. – № 2 – URL: [www.science-sd.com/455-24170](http://www.science-sd.com/455-24170) (20.11.2013) – 12 p.
  45. Levine R.D. Supercomputers. *Sat. na.u.-popul. articles. Trans. From the English .. Ed. V.M. Kurochkina. - Moscow: Mir, 1986. - P. 3-30.*
  46. Evreinov EV, Parangishvili I.V. Digital automatic machines with an adjustable structure (homogeneous media). -M .: Energy, 1977. - 240 p.
  47. Kalyaev A.V. Homogeneous commutated register structures. - Moscow: Moscow: Sov. Radio, 1978. -236 p.
  48. Mishchenko VA, Kozyuminsky V.D., Semashko A.N. Multifunctional automata and component base of a computer. Ed. V.M. Mishchenko. - M .: Radio and Communication, 1981. - 249 p.
  49. Balashov EP, Smolov VB, Petrov GA, Puzankov DV Multifunctional regular computational structures. - Moscow: Sov. radio, 1978 - 288 p.
  50. Yakubaitis E.A. Multifunctional Logical Models // *AVT*, 1976.-№2. P.1-15.
  51. Nikitin AV, Artificial neuron // "Academy of Trinitarianism", M., El No. 77-6567, public.20230, February 20, 2015. - 44 c.
  52. McCulloch US, Pitts V. Logical calculus of ideas related to nervous activity // *Neurocomputer* .- 1992.- No. 3, 4.- P. 40-53.

53. Rozenblatt, F. Principles of Neurodynamics. Perceptron and the theory of brain mechanisms .- Moscow: Mir, 1965.- 480 p.
54. Minsk M., Piperte S. Perceptrons. - Moscow: Mir, 1971.- 208 p.
55. Hinton J.E. Training in parallel networks // Reality and predictions of artificial intelligence .- Moscow: Mir, 1987.- P. 124-136.
56. Glushkov V.M. Synthesis of digital automata. - Moscow: Fizmatgiz, 1962.-248 p.
57. Promonenkov VK, The Information Paradigm // "Academy of Trinitarianism", M., El No. 77-6567, pub. 20589, 16.05.2015 -17 p.
58. Palagin A.V. Reconfigurable Computing Systems: Fundamentals and Applications / A.V. Palagin, V.N. Opanasenko. - К .: Просвіта, 2006. - 280 p.
59. Marakhovsky LF Multilevel devices of automatic memory. I ч. - Kiev: USIM. - No. 1.- 1998.- P. 66-72.
60. Marakhovsky LF Multilevel devices of automatic memory. II ч. - Kiev: USIM. - №2. - 1998. - P. 63-69.
61. Marakhovsky LF Multifunctional memory circuits. - Kiev: USIM - No. 6.- 1996.- P. 59-69.
62. Marakhovsky LF Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996.-128 p.
63. Marakhovsky LF Fundamentals of the new information technology. Fundamental basics of designing reconfigurable devices of computer systems and artificial neurons: monograph. LF Marakhovsky, NL Mikhno - Germany: Saarbrcken, LAP LAMBERT, 2012. - 347 p.
64. Marakhovsky LF Expansion of the fundamental foundations of the modern elemental base of computer systems // "Academy of Trinitarianism", M., El No. 77-6567, pub. 2006, 26.01.2015. - 12 p.
65. Lyapunov AA, Problems of theoretical and applied cybernetics, M: Nauka, 1980. - P. 320-323.
66. Kalashnikov Yu.Ya. "Triad of life (substance, energy, information)", SciTecLibrary.ru; 14.08.2007. Website: [http // new-idea.kulichki.com](http://new-idea.kulichki.com).
67. Popov VP, Krainyuchenko IV, Global evolutionism and synergetics of the noosphere, Essentuki, IEBP, 2003 (holism.narod.ru).
68. Popov VP, Krainyuchenko IV, Information and energy // "Academy of Trinitarianism", M., El No. 77-6567, pub. 18083, June 27, 2013.
69. Tatour V.Yu. 15 years of the SKIF project: history and results. // "Academy of Trinitari-zmah" - M .: - El. No. 77-6567. Publ. 18063, 06/09/2013.
70. Molchanov IN, Perevozchikova O.L. Khimich OM, Mova VI, Nikolaychuk AA Intellectual personal computer of hybrid architecture // Iskus-tvenny intellect. - 2012. - №3. - P. 73-78.
71. Palagin A.V. Reconfigurable Computing Systems: Fundamentals and Applications/ A.V. Palagin, V.N. Opananenko. - К .: Просвіта, 2006. - 280 p.
72. Kalyaev IA, Levin II, Semernikov EA, Reconfigurable computing systems. <http://fpga.parallel.ru/papers/kaljaev4.pdf>

73. Gudilov V. V. Reconfigurable Vector Evolutionary Processors // Izvestiya SFU. Technical science . 2013. № 7 (144). URL: <http://cyberleninka.ru/article/n/rekonfiguriruemye-vektornye-evolyutsionnye-protssory>
74. Marakhovsky LF, Gavrilenko VV, Mikhno NL Mathematical foundations of digital automata in the third genus. // Journal of National transport University.// News of the National Transport University. - Part 2. - K.: NTU.- Issue 17, 2008. - From 329-335.
75. Marakhovsky LF, Gavrilenko VV, Zaitsev OV, Mikhno NL Structural synthesis of automata for a one-hour section of the sun and an environment / Information Science // News of the National University of Biology and Nature Conservation of Ukraine. - K., 2009 p. - Vip. 139. - P. 114-120.
76. Marakhovsky LF Fundamentals of the new information technology. Fundamentals of designing reconfigurable devices of computer systems and artificial neurons: monograph. LF Marakhovsky, NL Mikhno - Germany: Saarbrcken, LAP LAMBERT, 2012. - 347 p.
77. Marakhovsky LF Artificial intelligence: a monograph. - K.: DETUT, 2014. - 241 p.
78. Khursin L.A. The beginning of the theory of systems of a social type. - K.: Svit, 2001. - 279 p.
79. Gagin V. System analysis "The blade of life." - Odessa: El. ed., 2001. - 315 p.
80. Marakhovsky LF, Moskvina MM, Moskvina VM та ін. Pristriy for the knowledge of catastrophes in basic diagrams of memory on elements I-NI. - The patent. Зареєстровано в Держаному реєстрі патентів України на корсні моделі № 88895 dated 10.04.2014 p. - (51) IPC F15C 1 | 10 (2006.01) - bul. № 7. - 9 p.
81. Marakhovsky LF, Moskvina MM, Moskvina VM та ін. Attachments for the knowledge of catastrophes in basic diagrams of memory on elements ABO-NI. - The patent. Зареєстровано в Держаному реєстрі Патентів України на корсні моделі № 84889 dated 11.11.2013 p. - (81) IPC (2013.01) G06N 7/00 - bul. № 21.-11p.
82. Reference book on digital computers: (processors and memory) / BN Malinovsky, EI Bryukhovych, EL Denisenko and others / Ed. BN Malinovsky. - K.: "Техніка", 1979. - 366 p.
83. Marakhovsky LF, Moskvina MM, Moskvina VM Analis pratsezdatnosti catastrophic vidmovy at base schemes pam'yati // IX mizhnarodnaya naukovotekhnichnaya conference "Girotehnologii, navigatsiya, keryvannya ruhom i konstruyuvannya aviaciino-kosmichnoï tehniki: Zbirnik dopovidei. - K.: NTUU "KPI", 2013.- 7 p.
- To the conclusion
1. Glushkov V.M. Synthesis of digital automata. - Moscow: Fizmatgiz, 1962. - 476 p.
  2. Glushkov V.M. Theory of algorithms. -K.: KVIRTU, 1961. - 167 p.
  3. Glushkov VM, Kapitonova Yu.V., Mishchenko AT Logical design of discrete devices. - K.: Science. dumka, 1987. - 264 p.

4. Bir Stefford. Cybernetics and production management. -M.: Izd. "Science", 1956, - 392 p.
5. Handbook of digital computers: (processors and memory) / B.N. Malinovsky, E.I. Bryukhovich, E.L. Denisenko and others. Ed. B.N. Malinovsky. - K.: "Техніка", 1979. - 366 with.
6. Bazhanov VA, Volgin LI V. I. Shestakov and K. Shannon: the fate of one remarkable idea // Scientific and Technical Kaleidoscope, 2001, № 2, P. 43-48.
7. B.N. Sendov Bl. H., Rozov N.Kh. John Atanasov is the creator of the world's first project for the Electronic Computing Machine. The primes of the international symposium on the development of the first EOM are the introduction of European languages into the development of computer technologies. - K.: Phoenix, UAINP, 1998. - P. 31-32.
8. Sergienko IV, Kapitonova Yu.V., V.M. Glushkov is a pioneer in the mathematical theory of computing systems and the founder of the Institute of Cybernetics of the National Academy of Sciences of Ukraine // Prats of the International Symposium on the Development of the First Countries of the European Economic Area, which introduces European technology into the development of computer technology. - K.: "Phoenix" UAINP, 1998. - P. 17-24.
9. Marakhovsky LF, Voevodin SV, Mikhno NL, Sharapov O.D. Comp'yutera circuit: a workshop for bachelor's specialties. "Intelligent decision-making systems". - Kyiv: KNEU, 2009. -245 c.
10. Glushkov VM, Letichevsky AA Theory of Discrete Transformers // Publishing House of the Computing Center of the USSR Academy of Sciences. In kN. Selected works on general information on AI. Malcev. Novosibirsk, 1970.
11. Glushkov V.M. The theory of automata and formal transformations of microprograms. // Cybernetics, 1965. - №5. - P. 1-10.
12. Glushkov VM, Tseitlin GE, Yushchenko E.L. Algebra, Languages, Programming. - K.: Science. dumka, 1974. - 328 p.
13. Glushkov VM, Kapitonova Yu.V., Letichevsky AA Computer-aided design of computers. - K.: Science. Dumka, 1975. - 232 p.
14. Glushkov VM, Rabinovich Z.L. and others. Computing machines with developed interpretation systems. - K.: Science. Dumka, 1970. - 256 p.
15. Sergienko I.V. About new directions of computer science development // Cybernetics and system analysis. - 1997. - №6. - P. 3-93.
16. Glushkov VM, Ignatiev MV, Myasnikov VA, Torgashev VA Recursive machines and computers. - 1974. - Kiev. - 26 p. (Pre / IK of the Ukrainian Academy of Sciences)
17. Glushkov VM, Introduction to the theory of self-improving systems. - K.: КВИПТУ, 1962. - 109 p.
18. Glushkov, VM, Introduction to Cybernetics. - K.: Publishing House of the Academy of Sciences of the Ukrainian SSR, 1964. - 324 p.
19. Tatour V.Yu. 15 years of the SKIF project: history and results. // "Academy of Trinitarianism" - M.: - El. No. 77-6567. Publ. 18063, 06/09/2013.

20. Molchanov IN, Perevozchikova OL, Khimich OM, Mova VI, Nikolaychuk AA Intellectual Personal Computer of Hybrid Architecture // Artificial Intelligence. - 2012. - №3. - P. 73-78.
21. Palagin A.V. Reconfigurable Computing Systems: Fundamentals and Applications / A.V. Palagin, V.N. Opananenko. - K.: Enlightenment, 2006. - 280 c.
22. Kalyaev IA, Levin II, Semernikov EA, Reconfigurable computing systems. <http://fpga.parallel.ru/papers/kaljaev4.pdf>
23. Gudilov V. V. Reconfigurable Vector Evolutionary Processors // Izvestiya SFU. Technical science. 2013. № 7 (144). URL: <http://cyberleninka.ru/article/n/rekonfiguriruemye-vektornye-evolyutsionnye-protsessory>
24. Johnson Colin / IBM Cognitive Computer imitates brain work / - [http://www.informationweek.com/news/hardware/processors/231500276?Cid=nl\\_IW\\_weekend\\_2011-08-20\\_html](http://www.informationweek.com/news/hardware/processors/231500276?Cid=nl_IW_weekend_2011-08-20_html)
25. IBM is trying to imitate the human brain [http://www.pcwork.ru/ibm\\_pyitayutsya\\_imitirovat\\_chelovecheskiy\\_mozg.htm](http://www.pcwork.ru/ibm_pyitayutsya_imitirovat_chelovecheskiy_mozg.htm)
26. IBM is working on creating a "computer brain" <http://www.cybersecurity.ru/it/82336.html>
27. Marakhovsky LF, Expansion of the Fundamental Foundations of the Modern Elementary Database of Computer Systems // "Academy of Trinitarianism", M., El No. 77-6567, publisher.20068, 01/26/2015 <http://www.trinitas.ru/eng/doc/0023/001a/00231055.htm>
28. Problems of the construction of cybernetic systems: Sat. sci. Tr. / Academy of Sciences of Ukraine Institute of Cybernetics. V.M. Glushkova, Science. Council of the Ukrainian Academy of Sciences on the problem of "Cybernetics"; Editorial Board: VVPavlov Ed. and others - K., 1993. -70 p.
29. Gavrilov MA, Devyatkov VV, Pupyrev EI Logical design of discrete automata. - Moscow: Nauka, 1977. - 352 p.
30. Zakrevsky A.D. Logical synthesis of cascade circuits. - Moscow: Nauka, 1981. - 416 p.
31. Evreinov EV, Prangishvili I.V. Digital automatic machines with an adjustable structure (homogeneous media). - Moscow: Energia, 1974. -240 p.
32. Mishchenko VA, Kozyuminsky V.D., Semashko A.N. Multifunctional automata and element base of digital computers / Pod. Ed. V.A. Mishchenko. - Moscow: Radio and Communication, 1981. - 249 p.
33. Multifunctional regular computing structures. Balashov, V.B. Smolov, G.A. Petrov, D.V. Puzankov - Moscow: Sov. radio, 1978.-288 p.
34. Promonenkov VK, Information Paradigm // "Academy of Trinitarianism", M., El No. 77-6567, pub.20589, May 16, 2015, 17 c.
35. Glushkov V.M. About cybernetics as a science. Cybernetics, thinking, life. M.: Science. 1964.
36. Lyapunov, AA, Problems of Theoretical and Applied Cybernetics. M.: Science. 1980.-C. 320-323.

37. Popov VP, Krainyuchenko IV, Invariants of Evolution in the Human Psyche// "Academy of Trinitarianism", M., El No. 77-6567, publ.17828, January 10, 2013
38. Marakhovsky LF, Chechik AL, et al. Ways of cognition of regularities in the evolution of complex systems (Searches and evaluation of the choice of effective solutions and automata of the third kind) Collective monograph. - Odessa: LLC "Institute of Creative Technologies", 2012. - 278 p.
39. Marakhovsky L.F. Discrete devices with a multifunctional organization of memory (monograph) // Kiev. Institute of Nar. hoz-va .- Kiev, 1987, Dep .. in UkrNIINTI, on 30.12.87. No. 3346- Uk. 87.- 244 p.
40. Marakhovsky LF Finite Automata with a Multifunctional Memory Management System: Proc. allowance. - Kiev .: UMK VO, 1991. - 67 p.
41. Marakhovsky LF Devices of computers with a multifunctional system of memory organization: Proc. allowance. - Kiev .: UMK VO, 1992. -56 p.
42. Marakhovsky LF Fundamentals of the theory of designing discrete devices. Logical design of discrete devices on automatic memory circuits: monograph. - Kiev: KGU, 1996. -128 p.
43. Marakhovsky LF. Fundamentals of the new information technology. Fundamentals of designing reconfigurable devices of computer systems and artificial neurons: monograph. LF Marakhovsky, NL Mikhno - Germany: Saarbrcken, LAP LAMBERT, 2013. - 347 p.
44. Marachovsky L.F. Basic Concepts to Build the Next Generation of Reconfigurable Computing Systems.// International Journal of Applied And Fundamental Research. - 2013. - No. 2 - URL: [www.science-sd.com/455-24170](http://www.science-sd.com/455-24170) (20.11.2013) .- 6 p.
45. Marakhovsky LF. Computer-schematics: posibnik. - K .: KHEY, 2008. - 360 p.
46. Marakhovsky LF. Multifunctional memory circuits. - Kiev: USIM - No. 6.- 1996.- P. 59-69.
47. Marakhovsky LF. Multilevel devices of automatic memory. I ч. - Kiev: USIM. - No. 1.- 1998.- P. 66-72.
48. Marakhovsky LF Multilevel devices of automatic memory. II ч. - Kiev: Usim. - №2. - 1998. - P. 63-69.
49. Biological encyclopedic dictionary. - Moscow: Soviet Encyclopedia, 1989. - 864 p.
50. Timofeev-Resovsky NV, Vorontsov NN and others. A short sketch of the theory of evolution. - Moscow: Nauka, 1977. - 301 p.
51. Bryukhovich E.I. The future of computing, as it appears in natural laws and scientific foresight. // Pratsi International Symposium on History securing the first EOM that introduces europeans into the development of computer technology - K .: "Phoenix" УАИИП, 1998. - С. 344-349.
52. Marakhovsky LF, Mikhno NL, Moskvina MV Automata of the third kind - a new step to modeling the work of the human brain. - M .: "Academy of Trinitarianism", M., El No. 77-6567, pub. 17223, 17.01.2012. -9 with.
53. Marakhovsky LF, Moskvina MV Three scientific directions in the field of information processing// "INTERNET-Osvita-nauka-2012", the eighth international sci-

entific practical conference IOH-2012, 1-5 October, 2012: Collection of works. - Vinnytsa: VNTU, 2012.-P. 215-219.

54. Marakhovsky LF, Kozubtsov I.N. Interdisciplinary prism based on the expert system // Interdisciplinary research in science and education. - 2012. - №1Sp. [Electronic resource]. - Access mode URL: [www.es.rae.ru/mino/157-654](http://www.es.rae.ru/mino/157-654) (reference date: 06.07.2012) -10 p.

55. Marakhovsky LF, Kozubtsov I.N. Hypothesis of the wave theory of the development of cognition in the fractal dynamic scientific picture of the world of knowledge. // VIVernadsky and the noospheric paradigm of the development of society, science, culture, education and economy in the 21st century: a collective monograph / Under scientific. Ed. AISubetto and VA Shamakhov. In 3 volumes. Volume 1. - St. Petersburg.: Asterion, 2013. - P. 161-172.

56. Marakhovsky LF New interdisciplinary scientific direction. // "Academy of Trinitarianism", M., El No. 77-6567, pub. 18074, 18.06.2013 - 14 p.

57. Marakhovsky LF Fundamentals of the theory of synthesis of digital devices on automatic memory circuits: monograph. - K.: ГЭТУТ, 2014. - 278 p.

58. Marakhovsky LF, Moskvina VV, Moskvina MV Analysis of the reliability of catastrophic events at base schemes pam'yati. // IX International Scientific and Technical Conference "Gyrotechnology, Navigation, Motion Control and Aerospace Engineering": A Collection of Reports. - K.: NTUU "KPI", 2013. - 7 p.

59. Marakhovsky LF, Moskvina MV Basics of designing reconfigurable computer control systems. // IX International Scientific and Technical Conference "Gyrotechnology, Navigation, Motion Control and Aerospace Engineering": A Collection of Reports. - K.: NTUU "KPI", 2013.-8 p.

60. Marakhovsky LF, Romanok Yu. O. The question of constructing reconstructed microprocessors // Collection of scientific works DETUT Ministry of Education and Science of Ukraine. Series "Transport Systems and Technologies". - Vip.24. - K.: DETUT, 2014 - P. 187-195.

61. Marakhovsky LF The methods of the project are the information on the information and security systems: the Naval-methodical compiler for special purposes 8.05020203 "Automation and automation for transport (vehicle traffic)". Specialization "Computer-information security systems". - K.: DETUT, 2015. - 267 p.

62. Marakhovsky LF, Moskvina VV Комп'ютерні інформаційно-керуючі системи на залізничному транспорті: Navychalno-metodicheskij posibnik for magistriv special'nosti 8.05020203 "Automatics and automation on transport (rail transport)". - K.: DETUT, 2015. - 127 p.

63. Marakhovsky LF, Kovalev VV, Romanok Yu.O. Educational and methodical manual on discipline "Algorithmization and Programming" Laboratory workshop. - K.: DETUT, 2015. - 275 p.

64. R. L. Graham and V. Rödl. Numbers in Ramsey Theory, in Surveys and in Combinatorics. London Mathematical Society Lecture Notes Series, 1987, No. 2 123, pp. 111-153.

65. Nikitin AV, Logic of control of a cell // "Academy of Trinitarianism", M., El No. 77-6567, pub. 17037, 29.11.2011

66. Marakhovsky LF. Development of computer devices with a multifunctional memory organization system (theory and practice) (Specialty: 05.13.08 - computers, systems and networks, elements and devices of computer technology and control systems) - Thesis for a scientific degree of Doctor of Technical Sciences. - Kiev: Ministry of Defense of Ukraine, KSEU, 1995. - 355 p.

67. Marakhovsky LF. Development of computer devices with a multifunctional memory organization system (theory and practice) (Specialty: 05.13.08 - computers, systems and networks, elements and devices of computer technology and control systems) Volume II. Application. - Thesis for the degree of Doctor of Technical Sciences. - Kiev: Ministry of Defense of Ukraine, KSEU, 1995. - 131 p.