Министерство науки и высшего образования Российской Федерации
Сибирский федеральный университет

METHODS OF DESIGN AUTOMATION OF HETEROGENEOUS COMPUTING SYSTEMS AND INFORMATION MODELS OF OBJECTS

Монография

Электронное издание

Красноярск
СФУ
2020

УДК 004.41'22
ББК 32.97-02
Б820

Борде Бернгард Исаакович

Б820 **Methods of design automation of heterogeneous computing systems and information models of objects** : Монография / Б. И. Борде. – Электрон. дан. (2,4 Мб). – Красноярск : Сиб. федер. ун-т, 2020. – Систем. требования: PC не ниже класса Pentium I ; 128 Mb RAM ; Windows 98/XP/7 ; Adobe Reader V8.0 и выше. – Загл. с экрана.

ISBN 978-5-7638-4367-5

Creating information models of objects is characterized by increased laboriousness, and multivariate design of objects to select the optimal solution according to the efficiency criterion increases the time of creating objects. Technological progress and competition are forcing to reduce the time for designing and creating new facilities. The creation of a research software package that complements the design system can reduce the total time and eliminate human error. From the conceptual description of the project in the form of a formalized task (FT), a project in CAD or a batch file for automatic execution can be automatically obtained. The execution time of the command file is reduced by an order of magnitude compared to the command set in CAD. The possibility of error is excluded in the usual and proposed design option. A technology has been created that generalizes design at different levels of abstraction.

УДК 004.41'22
ББК 32.97-02

ISBN 978-5-7638-4367-5

Электронное научное издание

Подготовлено к публикации издательством
Библиотечно-издательского комплекса

Подписано в свет 30.10.2020. Заказ №11409
Тиражируется на машиночитаемых носителях

Библиотечно-издательский комплекс
Сибирского федерального университета
660041, г. Красноярск, пр. Свободный, 82а
Тел. (391)206-26-16; http://rio.sfu-kras.ru
E-mail: publishing_house@sfu-kras.ru

CONTENTS

INTRODUCTION

Development computer technology is accompanied by an increase in the complexity and variety of machines [1, 4, 6, 91] with a decrease in switching energy and the cost of computing operations. The implementation of these requirements is possible only in conditions of complex automation of production of computer equipment and components. The complexity of the components is constantly increasing, therefore, the automation of the design of computer equipment and components is an urgent task. Various computer-aided design systems (CAD) are known: industrial, educational, research and experimental advanced systems for checking the principles of action.

In a short period of 60 years (1956-2016), the path has gone from the first transistors of NPO Svetlana for a prototype of computers and desktop analog machines MN-7 on lamps to mobile devices and large data centers of Sberbank in Skolkovo [I6, I8, I24], designed in CAD REVIT [I1]. In all machines with an index, the MH model is nonlinear; diodes on a typesetting field for continuous logic operations and in nonlinearity blocks were used [140, 141].

Industrial CAD systems belong to the well formalized technical stages of design and are divided into specialized and integrated. Integrated CAD systems are distinguished by an expanded set of components and a variety of connectors and ports with different operating principles. The first specialized CAD of computers was CAD CASPI [9], created at ITMiVT RAS under the supervision of acad. S. A. Lebedev. As the specialized industrial CAD systems for computing systems, the domestic system Prim5.3, GRIF4 [78] and foreign CAD systems for personal computers are used: PCAD, ALTIUM, ORCAD, CADDY, PADS, EAGLE, CADSTAR. As integrated industrial CAD systems for stationary objects, we consider CAD systems from Autodesk Revit and Infra Works, ASCON Renga, working with the BIM object information model, in accordance with GOST and international standards ISO [I28-I30]. As integrated industrial CAD systems for mobile objects, we consider CATIA CAD systems.

Important is the adoption of professional standards for the training of specialists in the creation and modification of integration solutions, code 06.041 of the Ministry of Labor of the Russian Federation in 2017 [I31]. The purpose of the preparation is the integration of information systems and cloud services. Work should be carried out using automated systems. Professional standards must be consistent with industry needs for specialists.

Educational research CAD (UI CAD) correspond to the early poorly formalized design stages. In the early stages of design, synthesis and analysis of many variants of computing devices and systems is desirable. The goal of creating the COD (Conceptual Object Design) software and methodological complex is to increase the level of complexity of the tasks of designing heterogeneous computing devices and systems [12-15] and to improve the engineer’s labor productivity.

For descriptions of designed objects in a low-level language used in industrial CAD, the scope of description is proportional to the number of options.

The description of objects in a high-level language allows us to approximate the volume of the description of many objects of one class to the volume of the description of one object. In order to reduce the volume of descriptions, multilevel models of computing systems are used.

The levels of models of computing systems correspond to various tools, the complex of which is implemented in computer-aided design systems. Distinguish CAD of high and low levels. Low-level CAD systems correspond to the lower levels of model abstraction and are used for functional-logical and design. High-level CAD systems provide a description of many technical solutions for system and structural-algorithmic design.

The design process of complex systems and computers in particular is an iterative process of downward and partially upward design at the lower levels. High-level CAD systems are effective for automating the conversion of design decisions during the transition to various low-level CAD systems. The complex of tools for automating the conversion of design decisions between CAD systems of various levels is called the interface in the work.

A design decision perceived by a formal system is called a formalized task, which consists of various sections of a description of design decisions. A single entry of formalized tasks is desirable regardless of the number of applications. The conversion of a single formalized task into the format of various CAD systems is provided by the interface between high-level CAD and a specific low-level CAD. The standards for the description of electronic equipment EDIF (Electronic Data Interchange Format) and STEP (Standard for Exchange of Product data) ISO 10303 only partially provide conversion, as they are used only for the external representation of formalized tasks.

The monograph complements the textbooks [101-103, 110], and it considers a multi-level model of a computer system, and features of synthesis and analysis. The main attention is paid to the automated analysis of heterogeneous computing systems with digital and analog signals at various levels of abstraction with a procedure-oriented component-wise description of many technical solutions based on standard universal high-level programming languages and non-procedural descriptions of design solutions at the lower levels.

For computer-aided design, starting from the early stages, under the guidance of the author, a high-level CAD and research CAD system was developed to automate the analysis of the operation of computing devices and systems and automatically convert formalized tasks to a low-level CAD format. A distinctive feature of UI CAD is the multiple use of a single formalized task for various applications and multifunctional component models with a common interface. The main application is an automated analysis of the operation of a device or system described in a formalized task. The formalized task describes the structure of the device or system and external influences. The main results of the analysis are timing charts and resource estimates, and with additional applications, the conversion of a formalized task into the format of one of the industrial CAD systems or a virtual reality language. Automation of the conversion of a formalized task to the format of a specific CAD is called an

interface with this CAD. The use of a single formalized task for various applications is provided by multifunctional component models with a common interface. An interface is defined by syntax and semantics tables for each type of component. Models of all types of components for each application are combined into static or dynamic libraries. A dialog box ensures that the application and component model libraries match. Automation of analysis and assessment of resources allows an engineer or student to focus on creative design procedures for the synthesis of descriptions, and automatic resource assessment makes it easier to choose the best solution. CAD has an interface with industrial CAD systems for importing and exporting descriptions.

The software is implemented for personal computers and servers, which allows you to use the software and methodological complex for local and distance learning. On personal computers, the complex is implemented in Windows, LINUX, and OS / 2 environments. The server part of the complex is implemented in Windows, LINUX, OS / 2 environments, in the dialog processing subsystem (PDO or PTS) of the S390 virtual machine system (VM / ESA, zVM). Formalized tasks can be performed on a personal computer or transferred for execution to application servers via the Internet. Currently, all servers are located on virtual machines of the data center of the Siberian Federal University and form the basis of the system of distance educational technologies (DOT) of the Siberian Federal University. Network services for training are hosted on e.sfu-kras.ru, and formalized project tasks are performed on the CAD UI on application servers. Many sample projects are hosted on application servers. Application servers can interface with real hardware. Application servers and methodical server are located on the virtual machines of the SibFU data center and are accessible on the Internet. This provides an environment for interactive online learning. Results are sent to users by e-mail or accepted by the user. In cloud structures, CAD is hosted on a data center and the results are stored there. The user is allowed to download the results or make them publicly available. Cloud structures are more efficient due to the lack of the need to purchase and install CAD systems at workplaces, access from a browser or network device, but users have security issues. Application servers can interface with real hardware. Application servers and methodical server are located on the virtual machines of the SibFU data center and are accessible on the Internet. This provides an environment for interactive online learning. Results are sent to users by e-mail or accepted by the user. In cloud structures, CAD is hosted on a data center and the results are stored there. The user is allowed to download the results or make them publicly available. Cloud structures are more efficient due to the lack of the need to purchase and install CAD systems at workplaces, access from a browser or network device, but users have security issues. Application servers can interface with real hardware. Application servers and methodical server are located on the virtual machines of the SibFU data center and are accessible on the Internet. This provides an environment for interactive online learning. Results are sent to users by e-mail or accepted by the user. In cloud structures, CAD is hosted on a data center and the results are stored there. The user is allowed to download the results or make them

publicly available. Cloud structures are more efficient due to the lack of the need to purchase and install CAD systems at workplaces, access from a browser or network device, but users have security issues. Application servers and methodical server are located on the virtual machines of the SibFU data center and are accessible on the Internet. This provides an environment for interactive online learning. Results are sent to users by e-mail or accepted by the user. In cloud structures, CAD is hosted on a data center and the results are stored there. The user is allowed to download the results or make them publicly available. Cloud structures are more efficient due to the lack of the need to purchase and install CAD systems at workplaces, access from a browser or network device, but users have security issues. Application servers and methodical server are located on the virtual machines of the SibFU data center and are accessible on the Internet. This provides an environment for interactive online learning. The results are sent to users by e-mail or accepted by the user. In cloud structures, CAD is hosted on a data center and the results are stored there. The user is allowed to download the results or make them publicly available. Cloud structures are more efficient due to the lack of the need to purchase and install CAD systems at workplaces, access from a browser or network device, but users have security issues. In cloud structures, CAD is hosted on a data center and the results are stored there. The user is allowed to download the results or make them publicly available. Cloud structures are more efficient due to the lack of the need to purchase and install CAD systems at workplaces, access from a browser or network device, but users have security issues. In cloud structures, CAD is hosted on a data center and the results are stored there. The user is allowed to download the results or make them publicly available. Cloud structures are more efficient due to the lack of the need to purchase and install CAD systems at workplaces, access from a browser or network device, but users have security issues.

The program-methodical complex allows you to describe many technical solutions in the form of formalized tasks at various levels of abstraction. Formalized tasks in high-level universal languages are unified in various syntax environments C ++, ADA, PLI, JAVA. Dialogue debugging of formalized tasks, automated analysis with comparison of estimated and actual results, as well as automatic resource assessment and documentation of results are provided. With satisfactory analysis results, many technical solutions or optimal structures can be transferred to industrial CAD systems PCAD, ALTIUM, (PADS Mentor Graphics), ORCAD, CADDY, (EAGLE, REVIT Autodesk), (CADSTAR Zuken), CATIA and PRAM 5.3. Examples of specialized cloud-based CAD systems are EasyEDA; Integrated Autodesk InfraWorks, Autocad 360, Fusion 360, Onshape [115].

Industry norms and rules are used in specialized CAD systems by industry, despite their changes over time due to the development of materials and technologies. In complex CAD, everything is much more complicated, and more time is required for the real use of rule systems (set of rules-SP). The development of Internet-based devices (IOTs) based on systems on a chip should facilitate the assessment of the status of all components and the use of rule systems in CAD.

Examples of an energy-efficient data transmission system for IOT are LPWAN and NB (teleofis.ru).

The software-methodological complex UI CAD <COD> is presented on an optical disk (DVD ROM), on university servers with network access (e.sfu-kras.ru), contains client and server versions for Windows, Linux, OS/2 and posted in the public domain in the library of the Siberian Federal University (bik.sfu-kras.ru).

The work was carried out in the laboratory of heterogeneous computing systems of the Department of Computing Engineering of IKIT SFU, used in classes and in the computer center of the Institute of Space and Information Technologies (IKIT) of SFU, and in the computer data center of the Siberian Federal University. The data center virtual machines host teaching materials (e.sfu-kras.ru) and personal dashboards for employees and students, as well as application servers for performing formalized tasks of projects in various environments with a large number of examples: CODWIN for Windows, CODLIN for Linux, CODOS for OS/2-EcomStation [50-57].

This work was supported by the Ministry of Education of Russia by order of 195 dated 16.3.1987 “Creation and development of educational and research software and methodological CAD complexes (UI PMK CAD) in universities”. Clause 3.2.17 of Order 195 approved the author’s work on PMK UI CAD CAD heterogeneous computing systems (PMK UI CAD CAD NVS). The complex is continuously developed by the author and is supplemented by the best projects of IKIT SFU students, carried out under the guidance of the author.

Requests for the contents of the book and the complex can be sent Professor B. I. Borde by email: bborde@sfu-kras.ru.

1. DEVELOPMENT OF COMPUTER SYSTEMS AND MEANS FOR THEIR MODELING AND DESIGN

1.1. The development of heterogeneous computing systems

Consider the main types of computers and the stages of development of information technology. According to the methods of presenting information, computers are subdivided [1, 2, 6, 140] into analog (AVM) [140], digital (digital computers) [91] and analog-digital (ADC) [53, 111, 129, 141]. In the analogue representation, the result corresponds to the signal value, the error is determined by the components and the level of interference and increases during the calculation process, as well as during storage and transmission of information in analog form over communication lines. Therefore, with the analog representation, all operations are performed simultaneously and are limited in space, the time for solving the problem does not depend on its complexity. However, the amount of equipment and cost are proportional to the complexity of the task.

In digital representation, the result corresponds to the state of many elements; each element can be in different, resistant to interference states. Elements with two stable states provide the highest noise immunity and low switching energy [35, 36, 125, 142, 144, 148]. The value of the threshold signal and the voltage of the energy source are determined by the level of interference. In digital computers, elements of logical operations and information storage are implemented. However, the error in performing calculations and data storage decreases with increasing number of bit elements. The transmission of information in digital form does not increase the error. Therefore, the digital computer sequentially performs operations on stored data. However, the simultaneous execution of operations on the data stream is carried out only in conveyor computers. The digital form of presentation of information made it possible to create an evolving global network of computers and network service centers (DSS) or data processing centers (DPC) [4, 5, 157, 158, 159].

At the first stage, due to the lack of general communication the user group systems ran on the same machine. The virtual machine system [5, 28–33, 86, 92] made it possible to provide an individual environment at the workplace due to a specialized communication subsystem with the ability to exchange messages. Specialized network services became available with the advent of data processing and transmission systems based on a virtual machine system. The complexes connected by the data transmission network allowed employees to receive specialized computing and educational services [86].

Personal computers with local communication subsystems made it possible to install software at the workplace and solve the required tasks. The effectiveness of personal computing systems was limited by the qualifications of users, high energy consumption and high cost of ownership.

The combination of personal computers with global communication systems [91, 100, 107, 157] and service centers made it possible to raise the quality of information systems. Software is installed at the workplace to solve

basic problems, and additional tasks and information retrieval are performed in the form of network services. The cost of ownership has decreased with increasing quality, but a qualified specialist is required for management. The increase in the number and complexity of computing and communication systems in all areas of activity has become limited by the number and qualifications of personnel.

Personal computing facilities are divided into stationary and mobile [4]. Stationary are located at workplaces and have a wired connection to the network, their speed and power consumption are constantly increasing, they are necessary for developers and professional users. Despite the effectiveness of stationary terminal machines (thin clients), they are not widespread.

Mobile personal equipment is represented by portable computers – MPC, tablet – TPC, telephones and communicators with a wireless connection to a data network. Increasing the degree of integration of the main components of data processing and transmission has allowed the creation of mobile Internet devices (MID) and ultra-mobile personal computers – UMPC, for example, Computer Card from Intel. Mobile network devices and computers are software compatible with personal computers, and monitor sizes average between monitors of tablet computers and communicators. Mobile tools are optimized not only in terms of cost per unit of productivity, but also in energy consumption. One of the main parameters is the operating time from an autonomous power source. Specialized systems on a chip (SOC) have been developed for mobile devices. The chip contains analog-to-digital and digital components and sensors, the connections of which are programmed and can be changed dynamically (PSOC). Examples of PSOC are INTEL D1000, D2000 crystals. Mobile users and affordable servers form a dynamic computing and communication system. Limited mobile computing and energy resources stimulate the use of network services that meet standards and are not dependent on a hardware or software platform [4]. Network services are performed at data centers combining computing resources, storage facilities, uninterrupted power supply and cooling. Virtualization can improve the efficiency of data centers [4, I6, I8, I24]. The main ones are the quality parameters and the reliability of the services provided.

Personal and mobile means of data processing and transmission allow today to receive online educational and computing services at any time at the workplace and at home [4].

1.2. Abstraction levels of computing systems

Design decisions are models of objects at various levels of abstraction. The initial stages of design [100–105] are distinguished by a high level of abstraction, which decreases with the specification of the structure and parameters of objects.

The following stages of design and maintenance of the life cycle of objects are known: the goal of creating and evaluating the effectiveness, conceptual, functional, design, technological, testing, maintenance and disposal of objects [28–33, 103, 104].

Different stages of design correspond to various descriptions of objects and criteria for their effectiveness. Distinguish the content and syntax of descriptions. The content of the descriptions is determined by the subject area and the design stage. The variety of descriptions is explained by their syntax, not content.

At the upper level of abstraction (fig. 1.1), functional models of systems as a whole are considered, which are called macro-models and are described by the functions of outputs and transitions. In various CAD systems, functional models of systems are presented in various forms and in different languages.

At the next level of abstraction are structural models that reflect the internal structure of components. Such models will be called micromodels. At this level, one can roughly estimate resources: mass and dimensions, static power and switching energy, thermal characteristics, external environment and cost [1, 2, 6, 9, 11, 27–33, 87, 101].

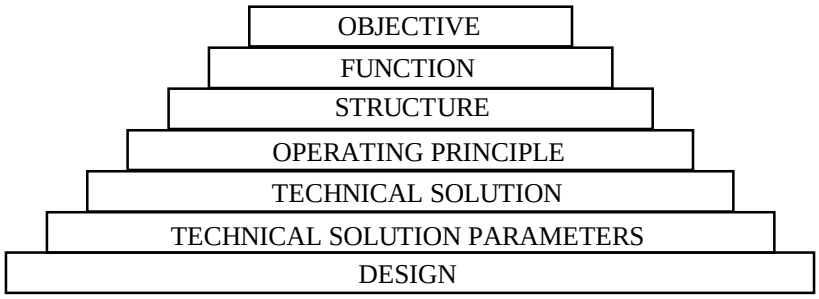

Fig. 1.1. The hierarchy of technical solutions at various levels of abstraction

The same structure can be implemented using components with different operating principles. It is possible to compare options from components with different operating principles. The processing subsystem can be implemented on some physical principles, and the communication one on others. The physical principles of the action of computer elements are described in detail in [1, 6, 8, 9, 16, 19, 20, 21] and will be considered only when assessing resources or optimization [95, 96, 124, 125].

Knowing the principle of action, taking into account the level of technology and technology, you can go to the level of technical solution. The technical solution should be feasible. After receiving the technical solution, it is possible to change the parameters of the solution in order to improve the quality of the product. Such examples are considered in the author's works on parametric optimization [12–15, 34–38].

Having determined the parameters of the technical solution, they proceed to the system design. For example, depending on the power dissipated by the crystal, an appropriate package will be assigned to it. The design is significantly affected by the external environment (operating conditions). Thus, at the upper level of abstraction are functional models of the components, and at the lower level are the structures of the components and their geometry. Geometric models of components are used in the construction of [101].

Design solutions for computing devices and systems at different stages of design represent many components and relationships. Relations can be of the type of affiliation (corresponding to the list of elements included in the device) or the transmission of information through channels. Relationships vary in content and presentation. The main thing is the content of the relationship, and the presentation form allows for many implementations that are limited by industry standards and specific CAD systems. International standards ISO 10303 and EDIF, PDIF define the protocol and data exchange structures between various CAD systems [128, 130, 133, 134].

There are problems of parametric [95, 96, 112, 124, 149] and structural optimization [3, 17, 27, 28, 60–63, 71, 83, 88, 102–105, 110]. In parametric optimization, controlled parameters corresponding to the extreme range of the output parameter are estimated [15–19, 34–38]. During structural optimization, many structures are synthesized, from which the best according to the quality criterion are selected [103]. Structures differ in components or component relationships. For computing systems, a relationship can be represented by a common data channel in the form of a connection for stationary components and the ability to exchange messages for mobile components.

In the table 1.1 shows the main levels of abstraction of heterogeneous computing systems and criteria for their effectiveness. The models for each level are independent, however, in the transition to nanotechnology [59], the logical, physical, and spatial levels of computing systems are dependent and affect the result; therefore, a number of iterations are necessary taking into account the dependent levels.

Table 1.1

NVS abstraction levels

№ level	Abstraction	Level	Name performance criteria
7	Processing method of information	System complex	Perfection of an energy or substance conversion system in an unsteady environment
6	Information Processing System Structures	End result	Efficiency of energy or substance conversion in information processing
5	Internal structure	System Information	Perfection Information Transformation System
4	Logical implementation of information converters	Structural	Perfection of the structure – the number of ticks for processing an application for an element
3	Physical feasibility	Logical	Perfection of logical information converters
2	Spatial placement	Physical	Switching operation complete

Continuation of the table 1.1

1	Processes in a heterogeneous environment	Spatial	Bulk (surface) density
0	No	Inhomogeneous physical environment	Permissible working conditions

At the systemic integrated level (7), the efficiency of energy or substance conversion is evaluated with or without a structure that processes information. At the level of the final result (6), the perfection of the information processing structure is evaluated. At the system information level (5), the perfection of the information processing transformation system is assessed with the abstraction of its internal structure. The structural level (4) is estimated by the number of ticks for processing an application for an element.

The logical level (3) is characterized by abstractions of the spatial distribution of components and their physical implementation. However, when coordinating a computing system (BC) with external channels, it is necessary to specify the physical implementation of the components. At the physical level (2), the state change energy for the binary component is estimated – the switching energy, which determines the power dissipated by the object for a specific frequency and limits the degree of integration of the components [69, 84, 99, 125, 142]. At the spatial level (1), the arrangement of elements tends to the maximum volumetric or surface density of the components, limited by technological standards and allowable power dissipation. Spatial placement is taken into account only in the design of objects with stationary components [6, 120–123].

The appearance of components with medium and high degree of integration has led to the need to create BC models that are heterogeneous in terms of abstraction. For example, in the BC model based on a microcomputer for signal processing, external digital or analog signals must be represented in a multi-valued alphabet. The level of representation of internal processing components is similar, up to interface nodes that act as converters of levels of abstraction of signal representation [33, 118, 138]. If the protocols of signal transformations inside the microcomputer are standard, then there is no sense in their detailed analysis and you can go to a higher level of abstraction. Therefore, BC models that are heterogeneous in levels of abstraction are needed.

The process of designing and testing a new facility is iterative (fig. 1.1). In each iterative cycle, design procedures for the synthesis, analysis, and decision-making are performed. The decision analysis procedure for known components is carried out automatically, and the synthesis and decision-making are implemented by an engineer or student.

The result of the synthesis is a description of the object, the result of the analysis x evaluation of characteristics and a diagram of the behavior of the object under certain external influences.

For simple objects, the analysis of behavior is simpler than its description, however, with an increase in the complexity of objects, the analysis of behavior becomes a very difficult task compared to its description. Therefore, the automation of analysis should be ahead of the automation of the synthesis of computer systems.

The iterative process is controlled to obtain a description of the object, the characteristics and behavior of which satisfy the task. You can control the description of the object obtained as a result of synthesis, and external influences on the object.

The purpose of creating a system or device is the final results obtained after the implementation of objects, which should fulfill the functions at given intervals of change in destabilizing factors. The formation of the technical task completes the stage of external design [48, 60–63].

1.3. The development of heterogeneous computing systems

When performing work on test automation and technological processes, it was necessary to find the optimal distribution in the space of programmable devices for interfacing with objects [42, 43, 44]. The analytical cost model allowed us to find a solution.

Microcomputers make it possible to implement multifunctional active devices for interfacing with objects (IWO). The accumulation of knowledge and models of processes and signals involves entering the deviations of the signals from the expected ones and setting the boundaries of the operation intervals according to a certain algorithm [17, 29, 72, 84, 85, 155, 156].

The proximity of active IWO to objects can reduce the level of interference at the input of devices and increase the information performance of the entire system. However, the distribution of IWO in space leads to an increase in hardware costs. The optimal distribution of IWO in space is determined by the level of development of the element base, from microcomputers with analog-to-digital devices to systems on a chip (SOC) [42–44, 58].

The optimal distribution problem in the IWO space on the basis of a microcomputer was solved [42–44] using the relative cost model that provides acceptable reliability of the results. For the time of physical and moral aging of TAM equipment and the time of existence of the structure of the TAS subsystem (estimation of the average time interval between tunings), an estimate is obtained of the optimal number of active IWOs based on NPUCO microcomputers and the optimal number of signals for IWOs – NCO. An informative parameter is the total relative cost of IWO for the entire life of the system:

$$KCUS = 2 * ((TAS / TAM) * CKUCO + TAS * CTUCO) / (CPLA * (LA + LB)), \quad (1.1)$$

where CKUCO, CTUCO – capital and operating costs for IWO on the basis of microcomputers; CPLA – linear cost of analog highways; LA, LB – geometric dimensions of the object.

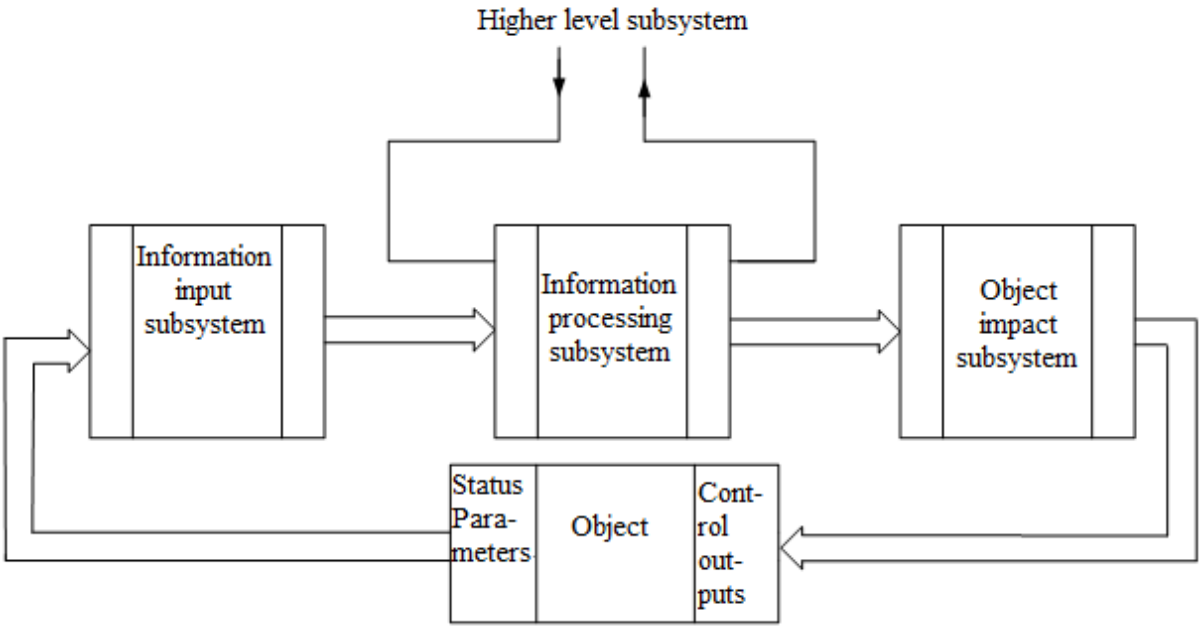

Fig. 1.2. The structure of the active device pairing with the object

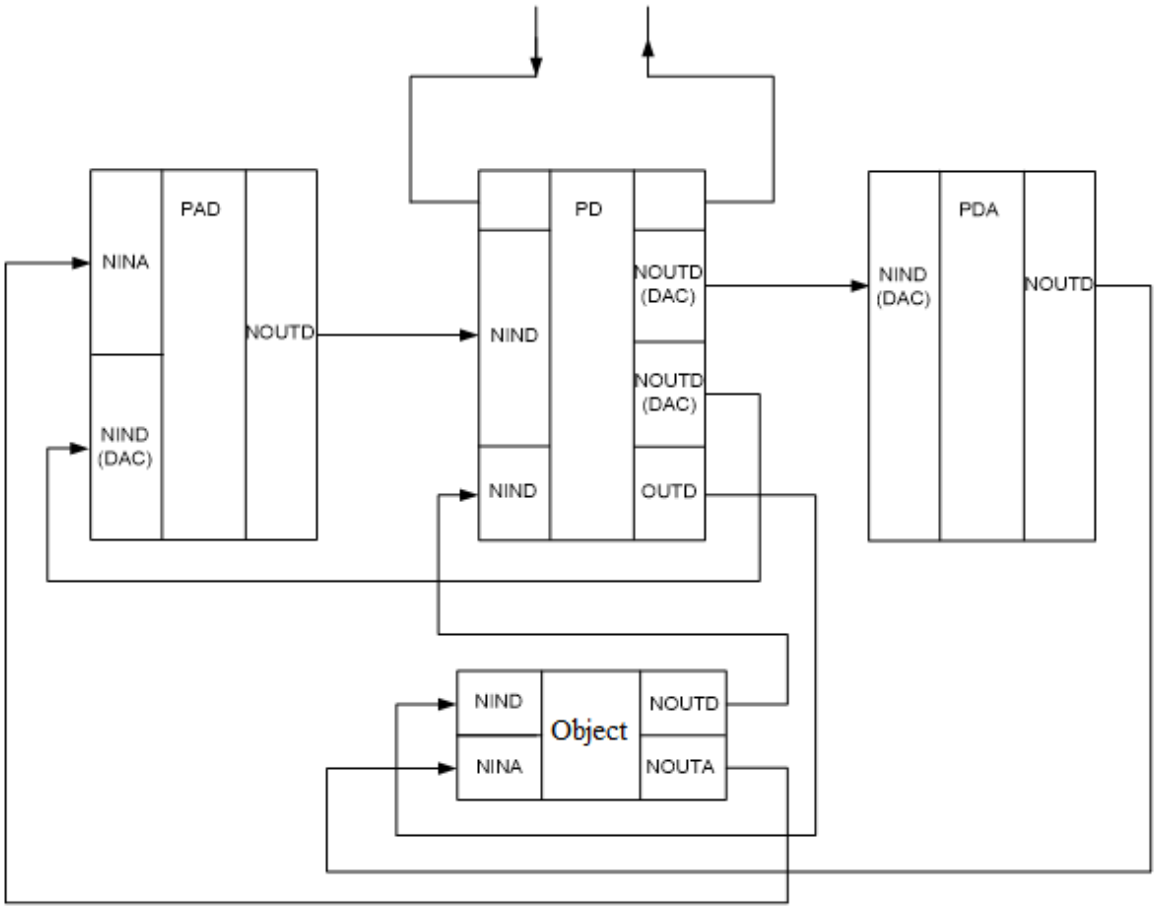

Fig. 1.3. The structure of the active device for interfacing with an object with digital-analog and analog-to-digital subsystems: PD – digital processor,
PAD – analog-to-digital processor, PDA digital-to-analog processor

Dependencies of NPUCO and NCO on the number of NCS signals, the KCUS parameter, and the relative cost of digital highways, taking into account the nature of the IWO with the next hierarchy level (coefficient KS) is determined by the expressions

$$NPUCO = \sqrt{\frac{NCS}{\frac{CPLD}{CPLA \cdot KC} + KCUS}} \tag{1.2}$$

and

$$NCO = \sqrt{NCS\left(\frac{CPLD}{CPLA \cdot KC} + KCUS\right)}, \tag{1.3}$$

where CPLD is the linear cost of the digital highways.

The results of calculations by formulas (2) and (3) are presented in an interactive graph in [28, 29]. Thus, with a decrease in the cost of IWO on the basis of a microcomputer, the degree of their distribution in space should be increased. The limiting factor is the relative cost of digital highways. Reducing the average interval between system adjustments increases the degree of distribution of IWO in space. The distribution of the signal conversion and processing subsystem based on the microcomputer is cost-effective, despite the increase in the cost of equipment, which is confirmed by the results of the development and implementation of such subsystems. In the near future they will be replaced by wired or wireless systems on a chip (SOC) [58, I3, I8].

In fig. 1.2 shows the structure of an active device for interfacing with an object [2, 3, 28, 29], many of which are distributed in space and work simultaneously. Figure 1.3 shows the generalized structure of an active device for interfacing with an object with subsystems of digital and analog signals [2, 3, 28, 29]. The output subsystem (PDA) converts groups of digital signals into analog ones and consists of two parts. The first is fed to the object and depends little on the algorithm of the digital subsystem. The second part is fed to the input subsystem for converting analog signals to digital (PAD) and, as knowledge about the object accumulates, it corresponds to the expected analog signals. A digital processor (PD) may contain the required number of cores, depending on the complexity of the object model.

1.4. Complexes of modeling and design heterogeneous computing systems

One of the first computer-aided design systems was CASPI [9], created under the leadership of S. A. Lebedev, B. A. Babayan and G. G. Ryabov. KASPI implemented a subsystem for modeling logical circuits, levels of physical and

spatial placement of components, tracing and preparation of data for technological equipment.

In 1970-1990 years, The Ministry of Education coordinated and supported the development of research software and methodological complexes for modeling and designing computational and test systems [29, 101, 102, 135, 137, 138, 139]. Most of the work related to the logical and circuit levels with specialized project description languages, similar to the well-known foreign modeling system SPICE [103]. The main program and methodological complexes were created under the guidance of I. P. Norenkov [102, 135], E. I. Artamonov [2, 3], V. I. Anisimov [139], V. M. Dmitriev in Tomsk [7], V. V. Toporkova [137, 138] and A. K. Polyakov [114, 115]. The Ministry of Radio Industry developed the PRAM (RAPIRA) complexes under the leadership of Yu. Kh. Vermishev [86, 128], of which we used PRAM53 together with PRAM2, PRAM53M / PC97, GRIF-4 [779], (development of PCAD-200X in NPO DIAMOND) Yelshin Yu. M.

In the process of developing computer systems and increasing the degree of integration of microcircuits, modeling and design complexes became specialized for microchip developers [1, 132] and for system integrators [8, 103, 114, 115]. Complexes for system integrators, to which we pay the main attention, along with the logical level, over time had to solve problems of the structural and system level. If in existing complexes the main input of information is graphic and does not require knowledge of programming languages, then in the future the main input of information will be textual or interactive, in universal languages for describing systems with additional libraries.

Modern complexes of modeling and design are represented by commercial and free products [58, 116]. For systems on a chip, the main task is to create an “executable system model” corresponding to a formalized task put into practice by V. M. Glushkov. Consider the complex “Visual Elite” by Mentor Graphics and “Riviera Pro” by Aldec. The main project description language is the IEEE std standard adopted in 2005. 1666-2005 SystemC [I23]. It is proposed to use the standard C language (C ++) with restrictions and additional libraries. Along with the standard, the use of VHDL fragments (a subset of the ADA language for describing digital systems) and the input of graphic fragments are allowed. For macro programming in CAD, NETScriptCAD is adopted.

The use of graphical fragments does not allow us to describe in a single formalized task many design options and move on to the second level of complexity of design problems [103]. The complexity of the complexes with the description in one main language on one platform does not exclude system errors at a high cost of the product.

Free CAD can be attributed to two groups. The first group allows you to abandon the commercial products of circuit design and engineering design (Kicad, gEDA, FreePCB, Electric) and is suitable for the technical design of one or two options for the object. The second group of free products is being developed by an international consortium based on standards, thanks to Internet technology, as an extension of the international tool project Eclipse [I4].

Thus, the existing systems of modeling and design do not allow to proceed to the solution of design problems of the second level of complexity while increasing or maintaining labor productivity. Complexes on the same platform with descriptions in one main language do not exclude system errors. There is no automatic conversion of formalized task options into a schematic diagram and a three-dimensional model with the display of signals at the early stages of design. There is no abstract type of components.

The necessity of solving the urgent problem of creating an open modular multilevel software and methodological complex for multivariate modeling and designing heterogeneous computing systems with the introduction of a formalized set of many options and the automatic formation of diagrams, schemes, three-dimensional appearance of the object and the assessment of its basic parameters in various syntactic environments and on various platforms is shown.

It is necessary to increase labor productivity in multivariate design and in the learning process by automating the formation of models and documents [233].

The author has proved the tendency of the optimal distribution of analog and digital data processing in space by approximating the means of analog-digital processing to the sources and consumers of signals in the process of development of computer systems. In the limit, when switching to systems on a chip (SOC), analog signals are processed directly at the sensor, and data is transmitted over a serial digital network. In the production of large batches, specialized SOCs are used, containing even sensors such as DS18B20 from Dallas, and later from Maxim. For small batches, PSOCs containing digital and analog subsystems, input and output communication subsystems produced by Cypress Semiconductor in conjunction with the programming environment are effective PSoC Development Tools.

Since 2018, we have been using a system on an Intel D2000 chip as a debugging module in conjunction with Intel System studio.

2. PRINCIPLES OF CONSTRUCTION OF MULTI-LEVEL MODELING SYSTEMS AND PROJECTIONS COMPUTER SYSTEMS

2.1. Description of design solutions for computing systems

Design decisions are models of objects at various levels of abstraction. The initial stages of design [1, 9–12, 89, 93, 95, 103] are distinguished by a high level of abstraction, which decreases with the specification of the structure and parameters of objects.

At first, the object is represented by a functional macromodel, then the object is represented by a micromodel of abstract functional components. Abstract functional components are represented by micromodels of abstract or specific functional components. The process continues to the level of specific functional components for which it is possible to evaluate the resources of creating objects.

I. P. Norenkov distinguishes design tasks by difficulty level [102, 103]. The first level of complexity includes tasks in which technical solutions are given and there is no synthesis. The second level of complexity is distinguished by many possible technical solutions in the presence of resources for analysis, evaluation and selection of the best option according to the efficiency criterion. The third level of complexity [103] includes tasks with many possible technical solutions with a lack of resources for analysis and evaluation of all possible options, so some of the options have to be excluded without a detailed analysis. Thus, tasks of the third difficulty level are transformed to the second.

Computer-aided design occupies an intermediate position relative to traditional (manual) and automatic design.

When designing aircraft, state standard R ISO 10303, introduced on July 1, 2000, provides for the generation of product information to supplement it during production, operation, maintenance and disposal [29, 33, 79, 104, 133]. Information can be used in computer systems in various organizations. Product descriptions must be complete and compatible.

The description of computing systems consists of models of components and connections between them. It can be represented in various ways. When describing the system in the form of a formalized task, the “Components” and “Connections” sections are sufficient.

A computer system as a design object is represented by a set of interacting components, each of which can be considered as a composite object. The model of an abstract object is represented by a data structure whose value reflects its state by methods or operations that transform the states of objects in accordance with perceived input messages. The object model generates output messages or procedure parameter values for other objects. Important for human perception is the form of representing objects and taking into account exceptional situations. Known symbolic, tabular and graphical forms of presentation. The behavior and interaction of objects is perceived better in graphical form (timing diagrams and circuit diagrams). These are the concepts for describing and representing the

subject area of heterogeneous computing systems. A formal representation of a subject domain (PRO) is possible using ontologies [10, 70]. The ontology model is represented as

$$O = (T, TREL, FVIEW),$$

where T is a finite set of concepts (concepts, terms); TREL – a finite set of relationships between concepts; FVIEW is a finite set of interpretation functions defined on concepts or relationships.

The basic concepts in the field of information technology and automated systems are given in GOST 34.003–90. The concept of functional components and the corresponding symbols on the circuit diagrams are defined in the ESKD standard GOST 2.743–91, and the concept of "product model" in GOST 2.052–2006. The relations between functional components (TREL) constitute a computational module, which can be considered as part of a formalized task (FT), a module description section (UNIT) and presented (FVIEW) as a circuit diagram or volumetric model [29, 32, 33].

The use of ontologies in the design and maintenance of objects is recommended by the Russian standard R ISO 15926, adopted in 2008. Due to the large number of specific components and terms, a multi-level model with a small number of basic concepts at the top level is used. As the working conditions of the object are clarified, the components are specified and their diversity increases. Multilevel presentation allows you to even out the complexity of decision-making at various levels.

International standards ISO 10303 define the protocol and data exchange structures between various CADs [103–105, 115, 133] and object tracking systems. An example of a standard for describing digital systems (VHDL) [8] is the document of the Institute of Electrical and Electronics Engineers (IEEE) No. 1076. The VHDL language is based on the universal algorithmic language ADA. The new edition of VHDL STD 1076.1–1999 (VHDL – AMS) introduced the ability to describe analog signals [132]. In 2004–2008, standards were adopted for the IEEE STD 1666–2005 system-level design language and its development. The lack of a standard leads to incompatibility of representations of design decisions in different CADs with identical contents [1, 31, 84, 104, 128].

The transition to a lower level of abstraction should be accompanied by the concretization of existing descriptions of objects and the introduction of new information in the form of relations. A single entry of information from those sections of the descriptions that are not changed in the subsequent steps is desirable. The description section usually consists of several relationships.

The uncertainty of the structure or parameters of the system [55–58, 65, 67, 90, 97, 155], which is large at the initial stages of design, is subject to specification. Pre-design studies are carried out when only the purpose of creating the system is known, and their result is the technical task (TT). The main part of the TT is a lot of relationships in the form of restrictions, for example: system

performance on a set of tasks no less than given, system cost no more than established, or limit value of a complex criterion cost-performance ratio.

The presence of TT allows you to compile and analyze technical solutions, many of which can be represented in the form of a tree [87, 104, 112] from vertices of type I and OR. Vertices of the AND type determine the necessary composition of the object, and OR vertices determine the indefinite components to be specified by analyzing the consistency of the properties of the component and the object or analyzing the operation of the device.

The result of structural design is the composition of the components and the relationship between them. At the stage of technical design, the composition of components and connections should be fully specified. On fig. 2.1 shows the flowchart of the design algorithm for the structure of objects.

The variety of object description rules in various CADs generates many design languages. There are various ways to enter descriptions: from typing to interactive systems with a graphical interpretation of descriptions and analysis results.

External design of an object includes the construction of a macromodel and resource limitation. Internal design includes the creation of micromodel variants with varying degrees of specification of components. At the lower level for specific components, it is possible to evaluate resources and compare various options for an object. Efficiency assessment is carried out only for models of objects for which the output signals correspond to the expected ones.

A feature of computing systems is the choice of the principle of operation for detailed functional diagrams. The principle of operation is selected depending on the working conditions and the external environment.

The process of obtaining a detailed micromodel of an object is called a <design route>. Unlike the algorithm, the design route may not be completed in a finite number of steps. The design of the object is carried out by the iterative method with training at each iteration. The brief synthesis and interpretation operations are performed by the engineer, and separate simple operations and labor-intensive analyzes are subject to non-availability. To automate the following stages of the design of the computing devices and the system of the results of the structural synthesis must be presented in a view from the window.

Spatial placement of structural components and coefficients is carried out at the design stage of the design. Component geometry models are stored in the general or local data base.

Multilevel CAD COD (Conceptual Object Design) is used to synthesize and analyze many variants of structures and automatically convert them to many design solutions for industrial CAD [12, 27, 32, 33].

To make a decision and synthesize objects in the COD software and methodological complex from a multivariate object-component formalized description of objects (FT), time diagrams are generated with automatic comparison of estimated and actual signals, tables of parameters and optimality criteria, circuit diagrams and images of objects for all options. Component designs

may display digital signals and temperature. To move to technical design, the Federal Law is converted to the format of a specific design system.

Analysis of the discrepancy between the expected and actual results allows us to make a decision on the modification of the rules for the synthesis of a class of objects. Synthesis rules are represented by a decision table and can be described in a formalized task using conditional operators or selection conditions.

Performance is determined using tests: on useful work performed per unit of time or on changing a digital or analog signal over a given time interval [32, 33, 132, 148]. When evaluating performance, the number of the changing digital (nppd) or analog (nppa) signal is set.

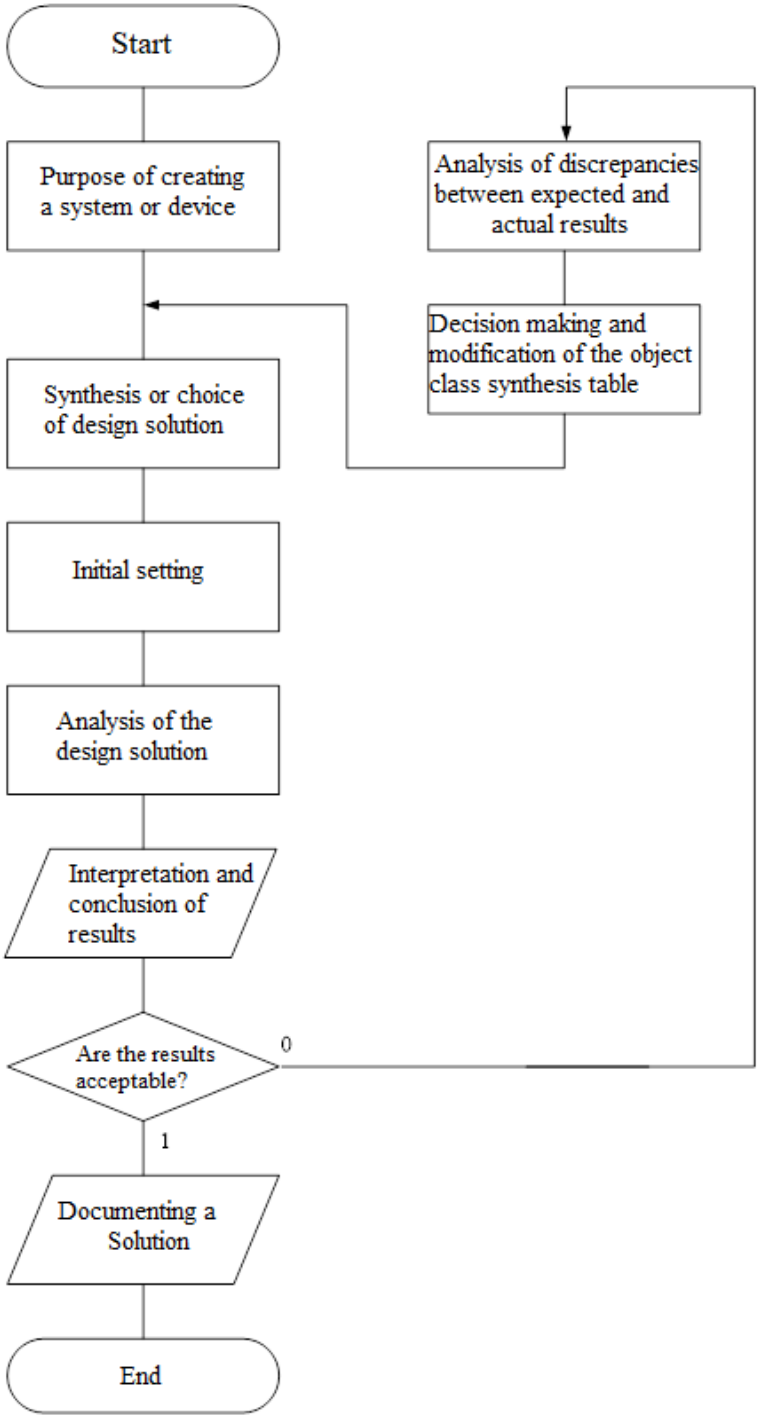

Fig. 2.1. Flowchart of an object design automated design algorithm

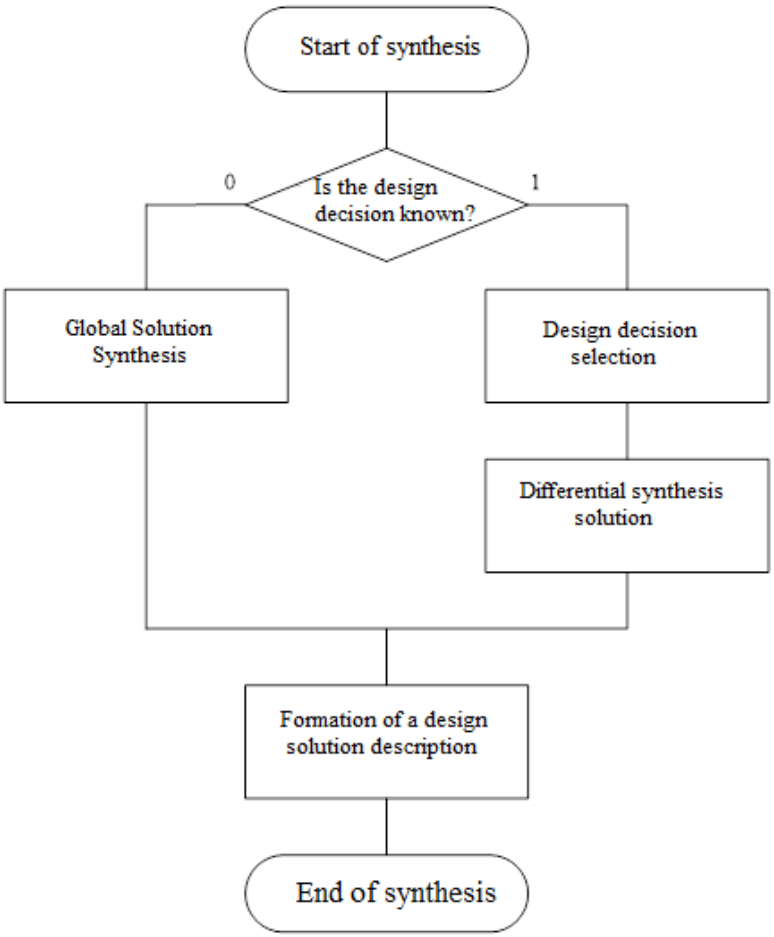

Fig. 2.2. Design decision synthesis flowchart

The minimum volume of the description of the object is presented in symbolic form, and the maximum – in graphic form. Therefore, the input of symbolic descriptions and the automatic conversion of descriptions into forms convenient for human perception are distinguished by the minimum complexity.

The description of the structure of the object is related to the results of internal design, which can be performed by various methods. If the design solution that meets the requirements of the assignment is unknown, then the basic procedure for synthesizing a description of a new object is performed. If there is a design solution required or close to it or a plurality of solutions, it is selected and, by making changes to the structure of the object, they achieve the requirements of the technical task. Methods for the synthesis of solutions are shown in fig. 2.2. Procedures for the examination of technical solutions for novelty require the identification of similar solutions and evidence of the significance of differences, which confirms the need for the chosen direction of differential synthesis and analysis.

When describing many decisions in a low-level language, the number of descriptions is equal to the power of many decisions, and when describing in high-level languages in one description there can be many solutions. Rational is the description in a high-level language of many solutions for one class of computing systems or devices.

For structural optimization, performance criteria are needed [33, 90, 93, 95, 96, 124, 125]. For stationary computing systems, the criterion of efficiency is the cost of a unit of performance. For mobile objects, such a criterion may be the mass of a unit of productivity or the mass of a computing system and an energy source per unit of productivity [29–33]. Such estimates for many solutions represent a complex and time-consuming task, the real solution of which requires tools and information support. Often a limited number of criteria are used, the expressions for which are summarized in table 2.1.

Table 2.1

The main criteria for the optimality of computing systems

Criteria	Dimension	Expression	Application area
Unit Cost	Relative Cost / MIPS	K = C / P	General Purpose Computing Systems
Unit weight	g / MIPS	K = M / P	Onboard Computing Systems
Power and mass unit of productivity	W / MIPS·g / MIPS	K = M / P · PF / P	Portable computing systems

The heterogeneity of computing systems is associated with various carriers and forms of presentation of information from multiple sources, receivers, and various structural organizations of processing subsystems with a wide range of performance [2].

For flexible automated productions and workstations, the functions of computing systems can be controlled by events, which requires dynamic restructuring of the structure. The joint use of processing and data transfer tools allows you to create dynamically controlled structures of information systems.

The operating conditions of computer technology differ significantly at different levels of the hierarchy. At the top level of the hierarchy are data centers [I6, I8, I14, I24]. Data centers are computing and communication resources with an information storage subsystem with autonomous power supply, climate system and security system. Modular data centers can operate in various conditions [I6]. Workstations and personal servers are typically used in the laboratory. Special operating conditions at the upper levels of the hierarchy, necessary for computer systems at the enterprise level, are replaced by laboratory conditions for personal servers and harsh conditions for specialized systems directly at workplaces and as part of technological equipment.

2.2. Structural synthesis of computing devices and systems

Structural synthesis methods [1, 29–33, 102, 103, 132] are presented in more detail in fig. 2.3. If technical solutions (TS) are known and enough resources are available, then a complete enumeration of the options is performed, otherwise the power of many solutions is reduced. For a class of objects, the synthesis algorithm is known and synthesis is possible in the dialogue mode or description

in one of the high-level languages (HLL). An example is the synthesis of a class of roughly accurate analog-to-digital serial count converters [29-33]. A formalized task allows you to get time charts for four options for structures and performance evaluations for fifteen options. The best option was to split the meter into two parts – coarse and accurate.

If the TS of the analogue is known, then differential synthesis is performed by adding and removing a component or links with an assessment of effectiveness. If TS is unknown, then global synthesis is performed.

The structure of the system is determined by many components and the relationships between them. System structures can be static for all variants of the system, static for one variant of the system and dynamically change within any of the variants. Description of the listed structures is possible in high-level languages. In low-level languages, descriptions of only static structures are possible. When changing the structure of the computing system in accordance with the option

$$S(N + 1) = FS(S(N), RS), \quad (2.1)$$

where S(N + 1), S(N) – structures for N + 1 and N-th version of the system; RS – rules for changing structures depending on the values of the estimated and actual signals, processing results, performance criteria and rules for the class of structures, as well as depending on external influences; FS is the function of forming the structure of variant N + 1.

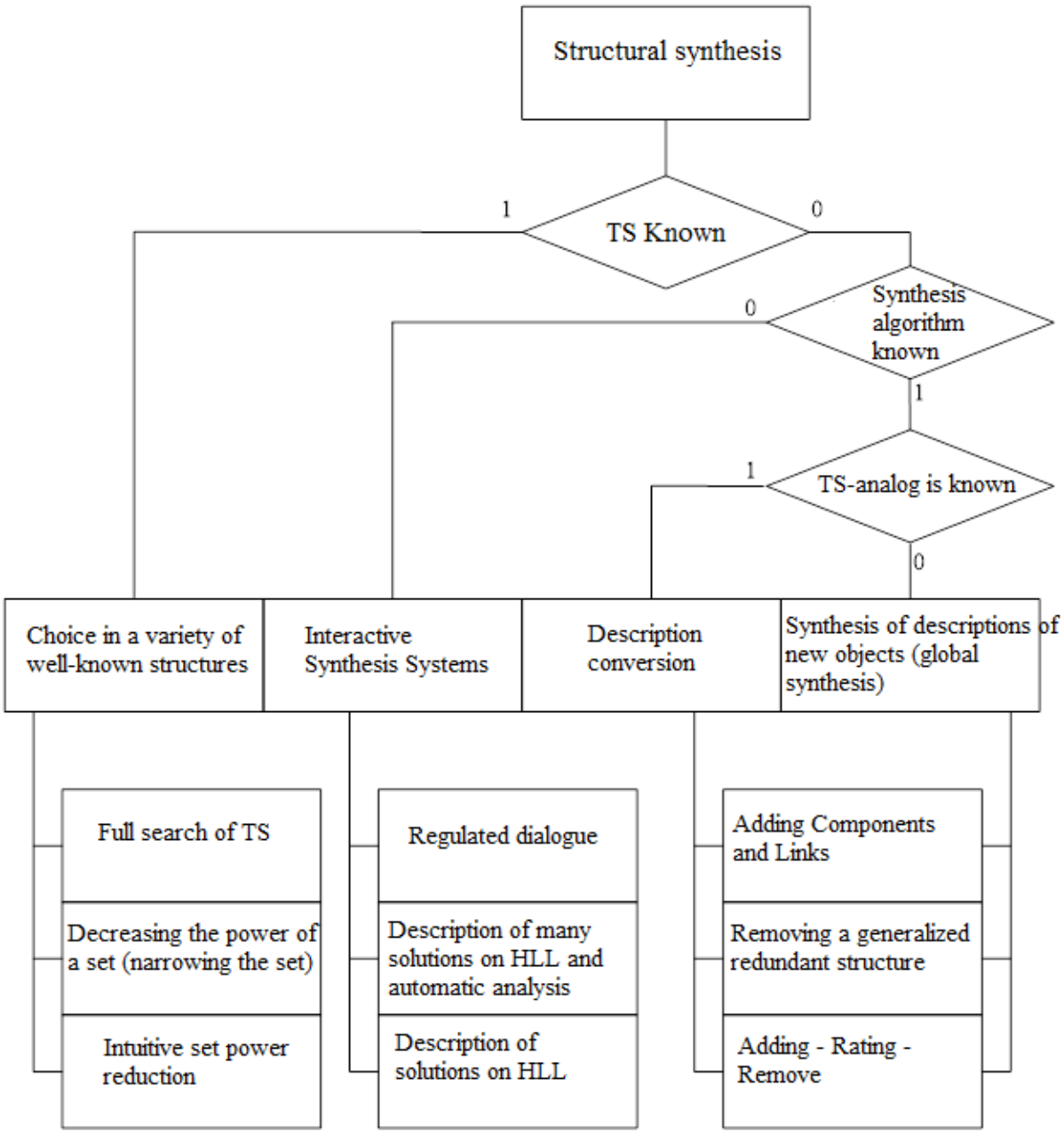

Fig. 2.3. Structural synthesis of computing systems

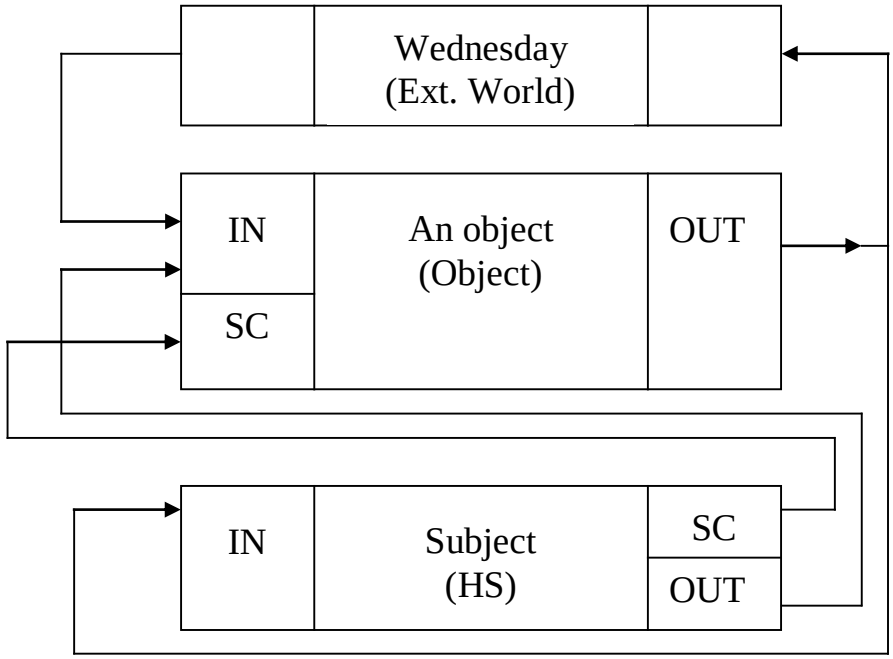

Fig. 2.4. System synthesis procedure taking into account the external environment

The system being created – the object is represented by a finite set of relations between concepts (TREL) [10], and to make a decision about changing a variant, the subject needs to present various properties of the object in visual form, for which interpretation functions (FVIEW) serve. As a result of the design analysis procedure, the object's behavior and resource assessment are obtained. From a formalized task, you can get a diagram in a standard viewer with activity display or a three-dimensional view of a module or system at an early stage of design. Visibility of the interpretation of the results reduces the time of decision making and the formation of synthesis rules [2, 9–11, 55–63].

The external environment is the main factor in choosing the principle of action. Electronic components operate in a limited temperature range and are unstable to many influences. Therefore, it is necessary to consider the principles of operation of analog-to-digital nodes, microcomputer-based systems using the optimal node structure depending on the applications, although in the future you can use a chip system or a signal processor.

2.3. Analog-to-digital devices of computing systems

The main nodes of the subsystems for interfacing with objects are analog-to-digital converters (ADC), analog multiplexers, nodes for processing analog signals – analog processors (AP) and digital-to-analog converters (DAC). Analog and digital multiplexers are typical components of computing systems. Their models are included in the component library, and the syntax and semantics tables are given in [6, 33].

As an example of an analog processor, the library includes functional models of an integrator and a scale converter. Along with the analog input and

output, the models contain digital input control signals and a controlled parameter. For the integrator, the controlled parameter is the time constant, and for the scale converter, the transfer coefficient.

The digital-to-analog converter refers to typical components [6, 31]. Functional DAC models have arbitrary bit depth and output signal range. In the table 3.15 shows the main parameters of the converters produced by various enterprises.

Analog-to-digital converters are among the most complex units, including all of the listed components, and are distinguished by structure, speed, error of operation, and resistance to interference. There are [2, 6, 33] ADCs of sequential counting, bitwise and single counting. The former are characterized by minimal speed and minimum hardware costs. The second occupy an intermediate position, and the converters of one reference are characterized by the highest speed and hardware costs. Comparative characteristics of the three types of ADC are given in table 2.2.

Table 2.2

Comparative characteristics of the ADC

Parameter	ADC Serial Count	ADC Bitwise Balancing	ADC single count
Transformation execution time (in measures)	2^n	n	1
Relative amount of equipment	$3 \cdot n$	$4 \cdot n$	$2 \cdot 2^n$
C / P Efficiency Criterion	$3 \cdot 2^n \cdot$ DELT	4 (n · DELT)	$2 \cdot 2^n \cdot$ DELT

In the table 2.2 n – number of bits, DELT – duration of one clock cycle, C – relative amount of equipment (cost estimate), P – performance.

Using a high-level language to describe analog-to-digital devices allows you to simulate analog-to-digital devices with various types of components. For example, in the ADC in fig. 2.5 in different versions, you can use excellent types of comparators.

The circuit of the tracking ADC with a reversible counter FP06R [31] is shown in fig. 2.5.

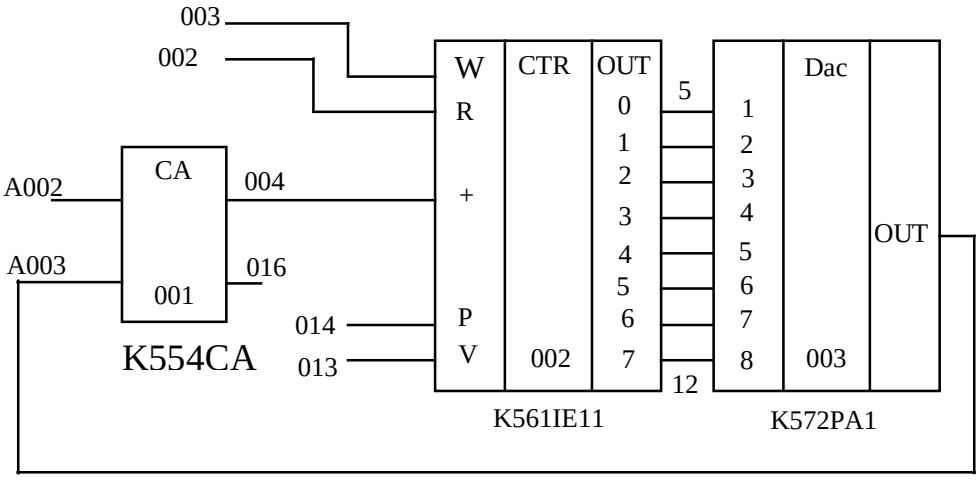

Fig. 2.5. Tracking ADC with reverse counter

Consider an example of optimization of an analog-to-digital serial count converter. The advantage of a serial ADC servo counter with a reversible counter is interference resistance. The presence of even a large noise on the analog input can lead to a longer transition to the equilibrium state, but will not lead to a gross error. The drawback of the tracking ADC is the large time it takes for the feedback signal to change over the entire range. This time can be reduced by dividing the serial counter into sections with the resolution of the section depending on the difference between the input signal (circuit A002 in fig. 2.5) and the feedback signal (circuit A003 in fig. 2.5). The outputs of all counters, as in the main circuit in fig. 2.5, connected to the input of the DAC. Instead of one comparator in the main circuit, a group of comparators is connected, the number of which corresponds to the number of sections of the counter CTR. At the outputs of the comparator, in order to synchronize signal changes and eliminate intermediate states, synchronous triggers are installed in those cases when they are included in the comparator. Comparators must differ in response thresholds, the outputs of the comparators are connected to logic circuits, and the outputs of the latter allow operation of the lower sections of the counter. It is necessary to evaluate the optimal number of CTR sections by the selected criterion. For the criterion of "cost-performance", the optimal structure of the ADC of a sequential count is achieved by dividing the counter into two parts for the senior and lower digits, introducing an additional comparator and logic circuit, and controlling the operation of the counters. in order to synchronize signal changes and exclude intermediate states, synchronous triggers are set in those cases when they are included in the comparator. Comparators must differ in response thresholds, the outputs of the comparators are connected to logic circuits, and the outputs of the latter allow operation of the lower sections of the counter. It is necessary to evaluate the optimal number of CTR sections by the selected criterion. For the criterion of "cost-performance", the optimal structure of the ADC of a sequential count is achieved by dividing the counter into two parts for the senior and lower digits, introducing an additional comparator and logic circuit, and controlling the operation of the counters. in order to synchronize signal changes and exclude intermediate states, synchronous triggers are set in those cases when they are included in the comparator. Comparators must differ in response thresholds, the outputs of the comparators are connected to logic circuits, and the outputs of the latter allow operation of the lower sections of the counter. It is necessary to evaluate the optimal number of CTR sections by the selected criterion. For the criterion of "cost-performance", the optimal structure of the ADC of a sequential count is achieved by dividing the counter into two parts for the senior and lower digits, introducing an additional comparator and logic circuit, and controlling the operation of the counters. the outputs of the comparators are connected to the logic circuits, and the outputs of the latter allow the operation of the lower sections of the counter. It is necessary to evaluate the optimal number of CTR sections by the selected criterion. For the criterion of "cost-performance", the optimal structure of the ADC of a sequential count is achieved by dividing the counter into two parts for the senior and lower digits, introducing an additional comparator and

logic circuit, and controlling the operation of the counters. the outputs of the comparators are connected to the logic circuits, and the outputs of the latter allow the operation of the lower sections of the counter. It is necessary to evaluate the optimal number of CTR sections by the selected criterion. For the criterion of "cost-performance", the optimal structure of the ADC of a sequential count is achieved by dividing the counter into two parts for the senior and lower digits, introducing an additional comparator and logic circuit, and controlling the operation of the counters.

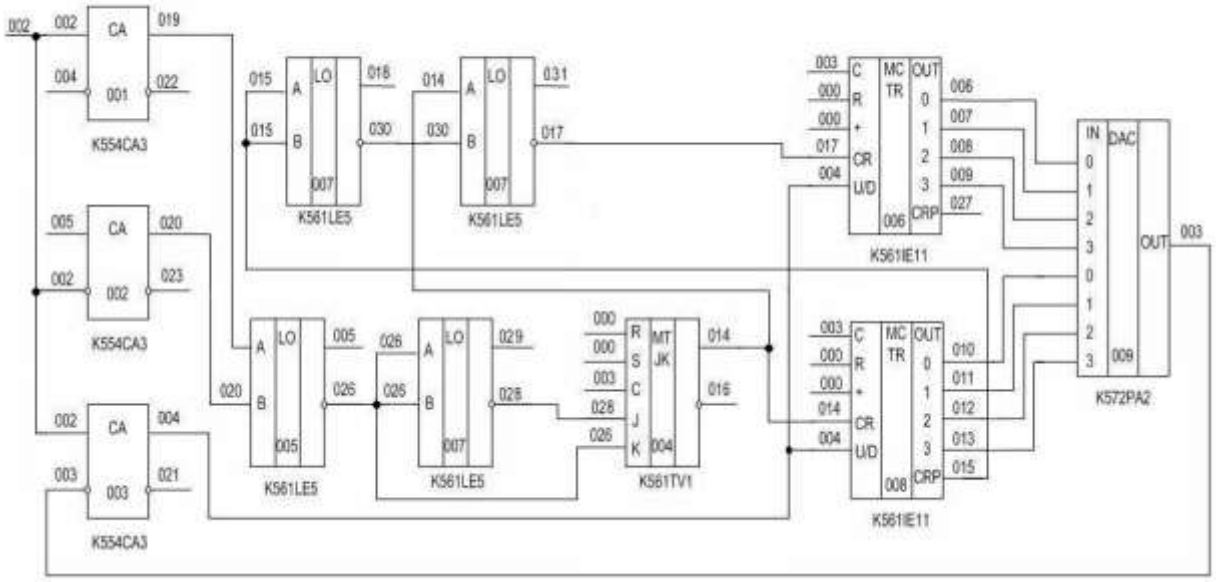

Fig. 2.6. Circuit diagram roughlyprecision ADC fp06rgt

Insignificant hardware costs allow increasing the ADC performance by an order of magnitude [31]. The results are confirmed by an analysis of the values of the effectiveness criterion. The main advantage is roughly accurate ADCs are the ability to set the predicted range of values using the installation inputs of the counter and reducing the input information flow [12]. The output results correspond to the deviation from their predicted values.

The optimization of structures is effective for analog-to-digital devices based on various principles. For example, an ADC of one sample can be built of two ADCs of one sample of a lower bit, a DAC of higher digits, an analog processor that performs the function of subtracting the output voltage of a coarse DAC from the input signal, multiplying the difference by a scale factor, and a digital adder that calculates the sum of the ADC codes of the lower and senior ranks. The input signal is converted by the ADC of higher digits. The high-end ADC output code is converted by the DAC into an analog signal and subtracted by the analog processor (AP) from the input signal. From the AP output, the signal difference multiplied by the scale factor (SF) is converted by the ADC of the lower digits. The output code of the ADC of the upper and lower digits is fed to the digital adder.

To test the effectiveness of the ADC, the entire working part was removed from the circuit – a rough counter, an accurate counter and the logical structure

that ensures their operation. Instead, a micromachine and two parallel interface registers of the MIR type were introduced: one for receiving information from comparators and transmitting it to the micromachine, and the second for receiving information from the micromachine and transmitting it to the DAC.

The introduction of a micromachine allows both operational processing and the presentation of results to the operator. In addition, it became possible to flexibly change the operational processing algorithm. The formalized task is presented in the fp08padc files [27–29].

The micromachine implements three signal prediction modes: zero mode - emulation of the tracking counter; the first mode - the signal change rate is known in advance; full information mode - the input signal or object model is known.

2.4. Inhomogeneous computing systems based on microEBM

The complexity of the task of the design depends on the indefiniteness of the system structure [29–33, 101–103], components and external influences. As an example, consider the option of a computer system based on microEBM, which are shown in the fig. 2.7–2.14. The systems are designed for the rapid assessment of the parameters of the input signals for several channels and are distinguished by their performance.

The structure of the system shown in fig. 2.7, consists of microEBM and interface units for input and output of digital signals. Analog signals must first be converted to code. The processing of information is carried out only programmatically, so the productivity of the system will be minimal.

The microcomputer-based subsystem with interval prediction and estimation of signal deviations from the expected one is shown in fig. 2.8. Knowing the number of teams to process one of the words NK and the average duration of the TK command, you can evaluate the productivity of P (1) processing of 31 input words.

Secondly, the structure includes a general control for all channels-controlled operational automatic pre-processing of the input signals. A subsystem based on a microcomputer with interval prediction and estimation of signal deviations over switched channels is shown in fig. 2.10. In order to reduce the number of devices and the cost, the operating circuit breaker was selected as common for all channels. MicroEBM is used immediately for automatic control, for the final processing of the results and the provision of their processing. The control of the automatic is reduced to the transfer to the relevant interface of the controllers of the initial setting for the resolution of the work and the preservation of the food.

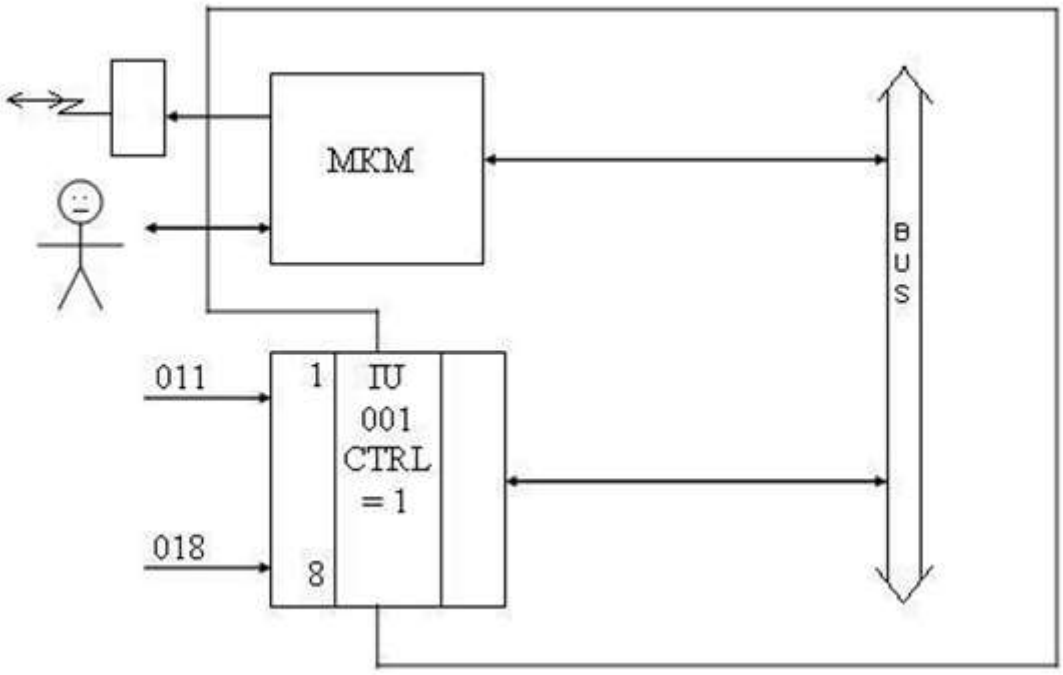

Fig. 2.7. Microcomputer-based subsystem without additional operating nodes

The third structure (fig. 2.11) is distinguished by an increased performance due to the parallel operation of the autos in each channel. MikroEBM transfers automatic machines to the initial state and at the same time resolves the operation. At the end of a predetermined time interval, the permission to operate is set to zero, and the machines keep the result to be received in microEBM. The multi-slot multiplexer MMX is economical to a few interface nodes on the average degree of integration. AFTER RECEIVING DATA AND PROCESSING IN MICROEBM, RESULTS ARE PROCESSED.

The fourth structure (fig. 2.12) is similar to the third, but more on the same account as the identical channels of preliminary processing. The structure is effective if there is a large integrated circuit (BIC) of the host nodes. The regular part of the computing system is conveniently described in high-level languages.

The system structure in fig. 2.13 provides a single-channel conveyor of operating nodes controlled by microEBM. The system differs not only by the highest performance for a separate channel, but also by the higher costs of the equipment. A multi-channel option of the system is available. A subsystem based on a microcomputer with many processors and interface registers with interval prediction and estimation of signal deviations through channels is shown in fig. 2.14.

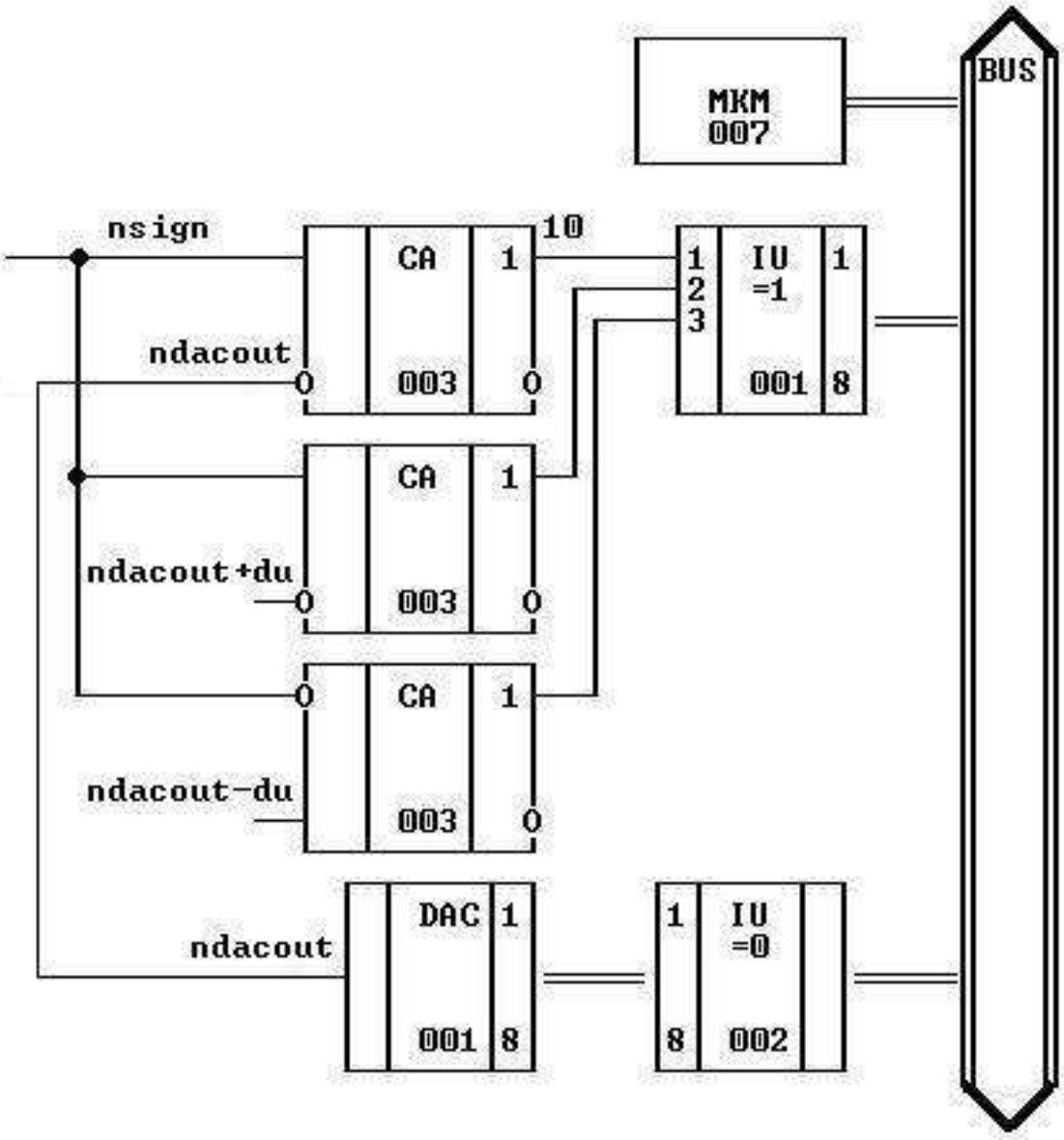

Fig. 2.8. Microcomputer-based subsystem with interval prediction and estimation of signal deviations

For the selection of a particular variant of the system, it is necessary to compare the options for any efficiency criterion. As a criterion [33], they take the ratio of productivity P to cost C or the ratio of cost to performance. In the vicinity of cases, instead of costing the product, the cost of the components is used, which is easy to evaluate, or even the relative weighted amount of the component. The work uses the relative cost of components, the ratio of the cost of components to the base element, which is a two-way valve. The relative cost value is used in the parameter tables of the UIPCAD components.

Table 2.3

Syntax and semantics of the parameters of the evaluation procedure system performance

№	Identifier	Format	Semantics	NOTES
1	CMKM	int	Cost of microEBM	Relative units
2	CIU		Interface node cost	Relative units
3	COE		Cost of operating element	Relative units
4	CMX		Multiplex cost	Relative units
5	NCAN		Number of channels	Number
6	TCAN		Time on the channel	Tacts
7	NK		Number of teams on processing	Number
8	DELT	float	Duration of command execution	c
9	FC		Clock Frequency	Hz
10	LRG	int	Register size	Number
11	FAM	char	User ID	FA22218

If you limit the power, the result will be power-limited, and if you limit the weight – to the maximum product. The cost of the components of the product is evaluated according to its price and implemented programmatically in the form of a sum. It is more difficult to evaluate performance. In simple cases, it is expressed by analysts [29, 33].

For five structures, computing systems are on fig. 2.7–2.14 the program for evaluating the cost of components C and the performance P, as well as calculating the K-ratio of the "performance" cost, is provided. The program is accessible from the main menu and is implemented in graphical and tabular versions. Syntax and semantics of program parameters are given in table 2.3, a list of structures – in table 2.4.

The program in interactive mode allows you to adjust the parameters of the computing system according to the table 2.4. The cost and effectiveness of the five systems considered are displayed in a graph and table. The EFFECTG program can be used to investigate the dependence of the optimal system variant (fig. 2.7-2.14) from changing the cost of the components or the temperature. The cost of the components and the technical characteristics are determined by the technology and the level of development of electronic texnics. Therefore, the BC structure, which is far from optimal now, can give the best result in the future. For research, we select the variable parameter, in accordance with the interruption of the change in the program, the cycle is omitted. It is better to format the results in the form of a graph of the dependence of the optimal option from the variable parameter.

As a result of the analysis, one selects from the structures of computer systems shown in the figures or an additionally proposed one that is optimal

according to the efficiency criterion. The selected system of optimal structure is described and analyzed by both behavior and resource estimates. After refinement, the description of the system can be transferred to the design level of design.

In fig. 2.7 presents a subsystem based on a microcomputer without additional operating nodes. When using a microcomputer in roughly exact interval ADC in fig. 2.6 you can exclude digital components and get the circuit in fig. 2.8. The states of the comparators are entered using the first interface register, and through the second, the estimated values are displayed on the DAC. In fig. 2.9 is a diagram of a computing system with a general operating automaton OA and switching input signals. Multi-channel option with roughly exact interval ADC is presented in fig. 2.10. An additional third interface register is used to control the multiplexer.

The use of a multiprocessor microcomputer with individual registers for interval signal estimation is shown in fig. 2.14. The unit of comparators can be replaced by an ADC of one reference with a small number (4–6) of bits. Similar schemes are shown in fig. 4.21 and 4.22.

The main attention should be paid to systems with a forecast of the signal interval or with an object model: with the accumulation of knowledge about signals and objects during the life cycle of systems, the flow of input information decreases and the flow of predictive estimates of signals increases.

Table 2.4

Computer Systems Structures

Structure Number	Type of structure of the subsystem based on microcomputers
1	No operating nodes
2	Common Operational Node
3	Operating Node Channels and Common Processor
4	Channels for operating and interface nodes
5	Managed Operating Node Pipeline
6	Channels of operating nodes per processor core

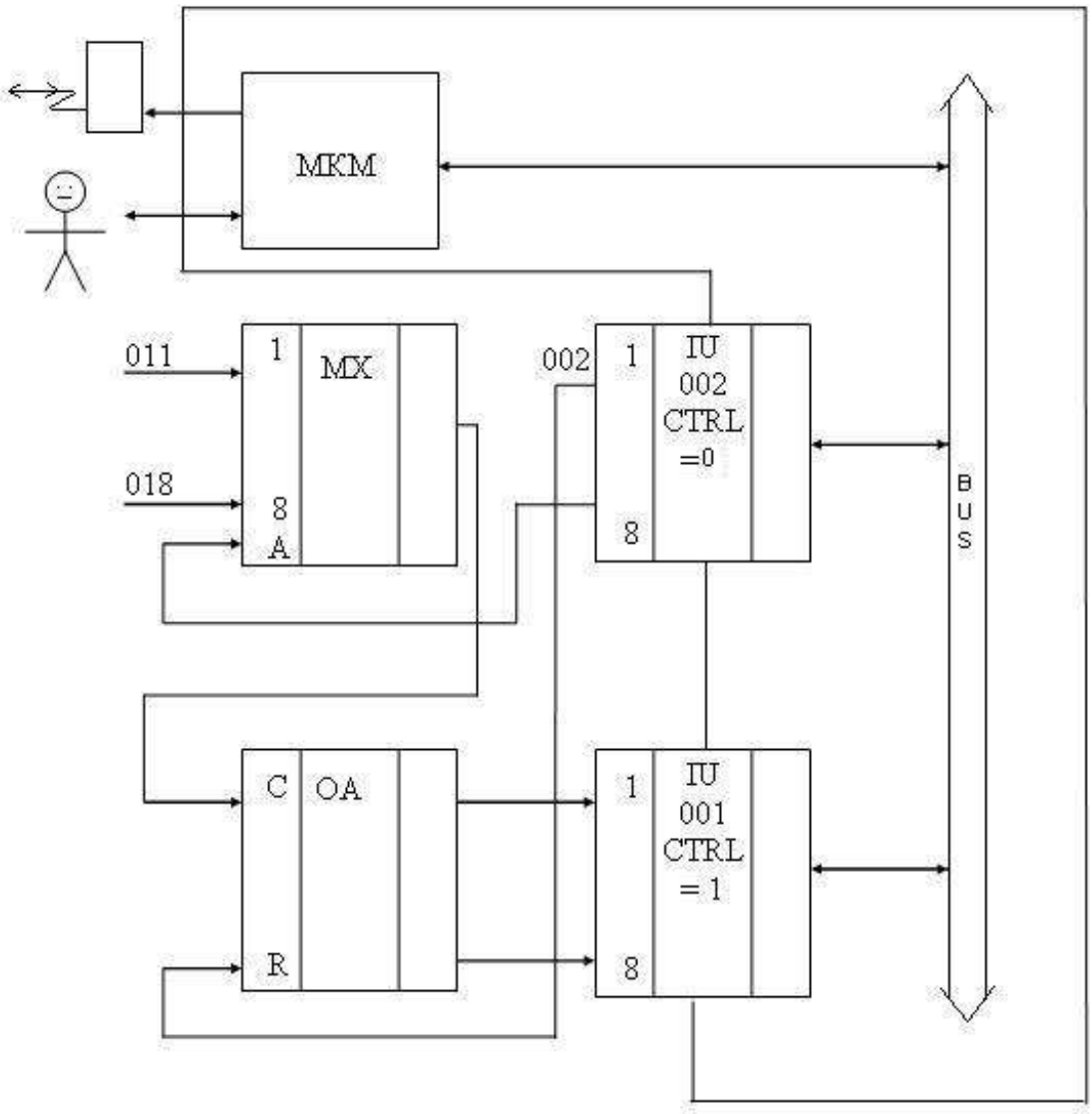

Fig. 2.9. A microcomputer-based subsystem with a common additional operational node

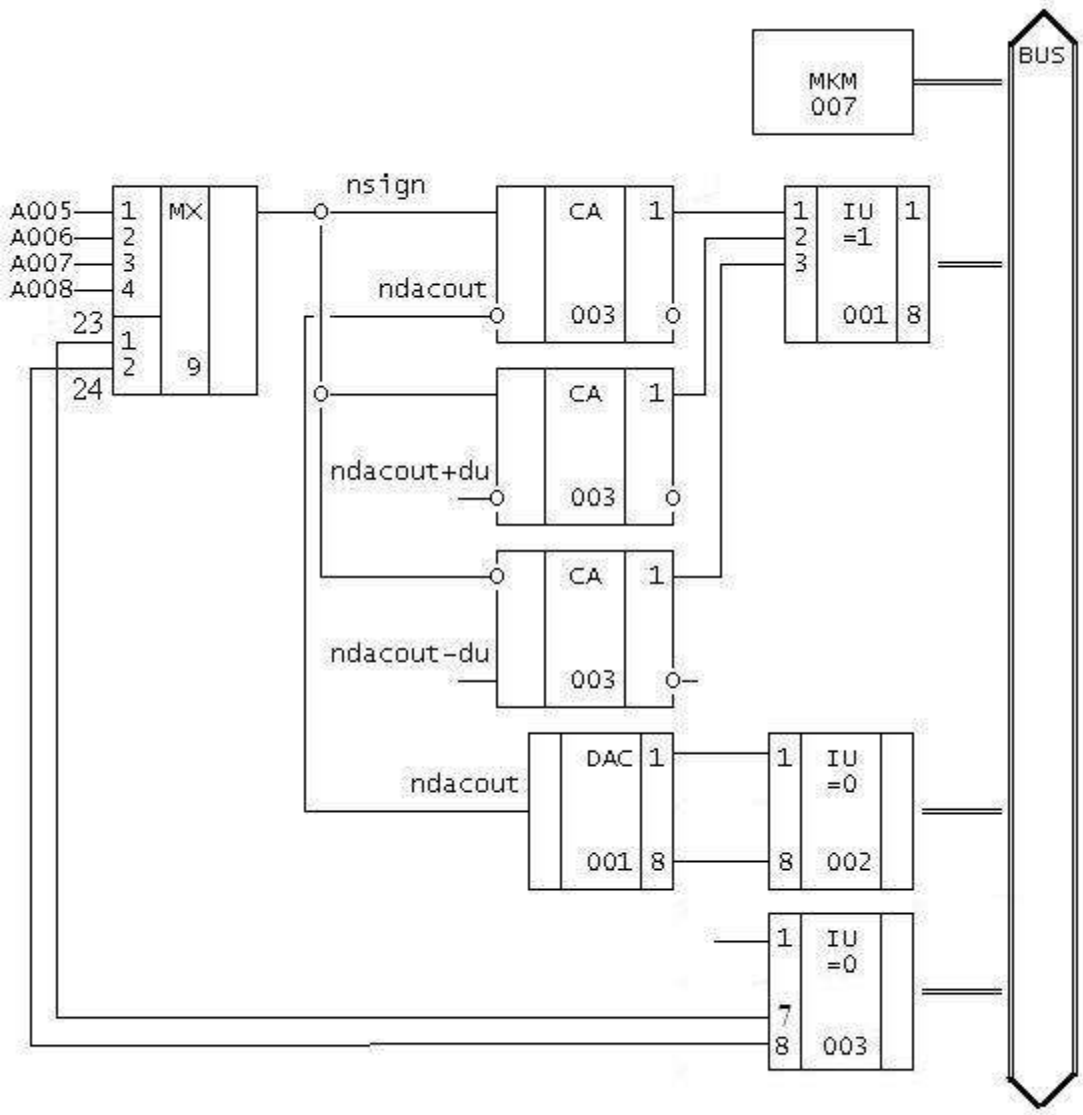

Fig. 2.10. A microcomputer-based subsystem with interval prediction and estimation of signal deviations over switched channels

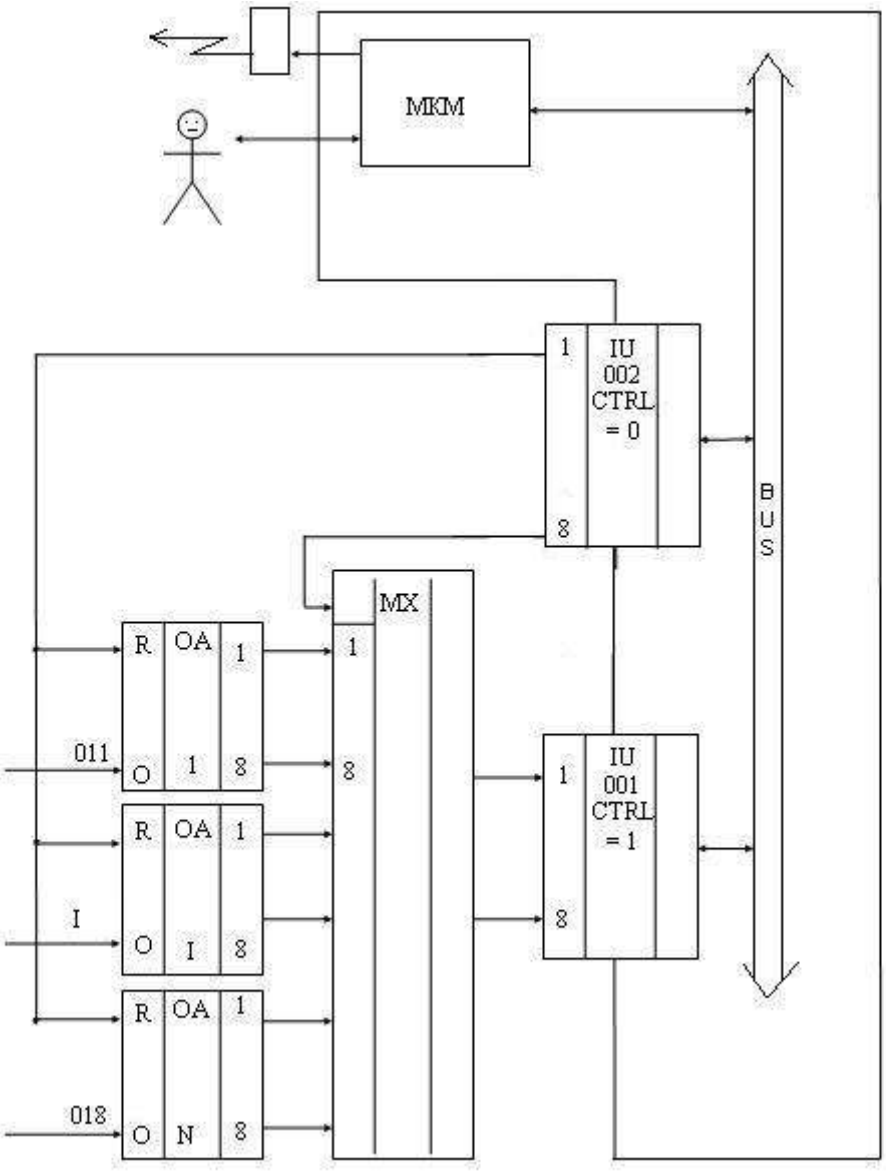

Fig. 2.11. Microcomputer-based subsystem with individual primary processing operating elements signals and common interface nodes

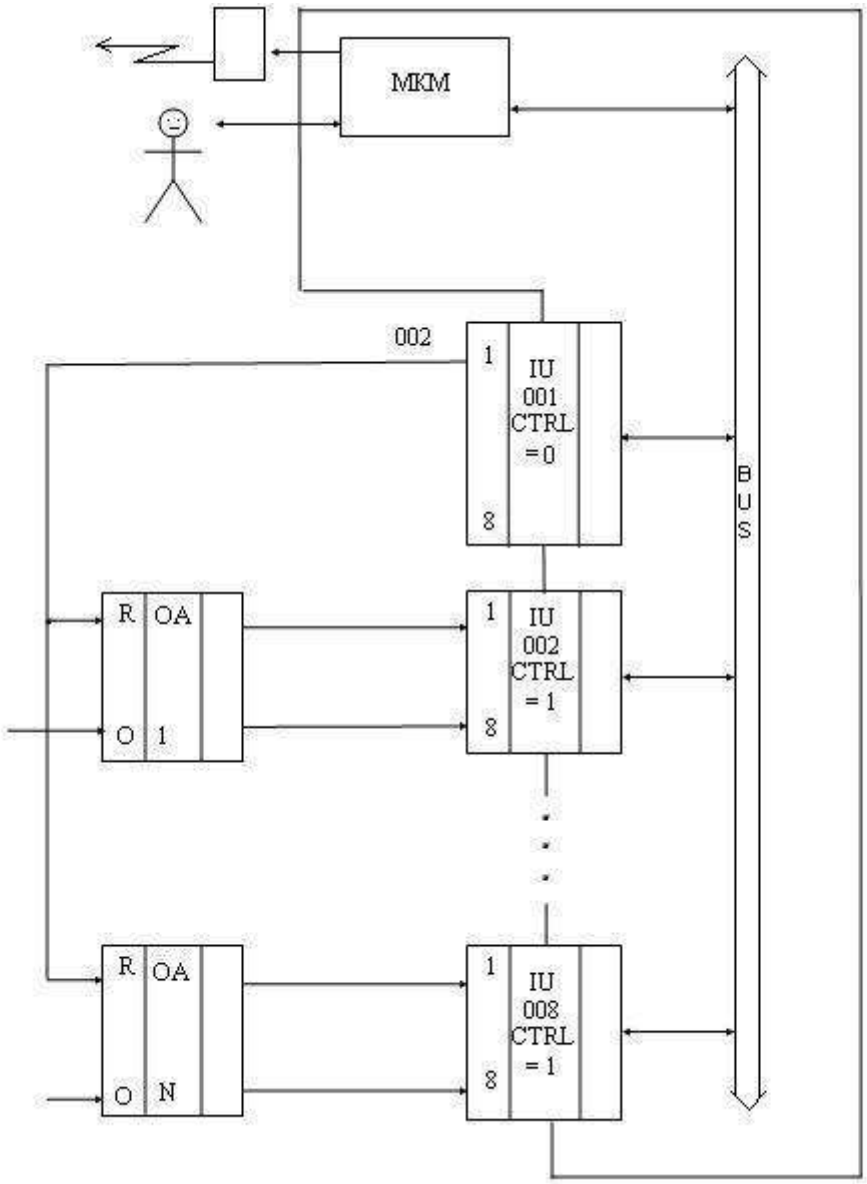

Fig. 2.12. Microcomputer-based subsystem with individual operational elements and interface nodes for primary signal processing

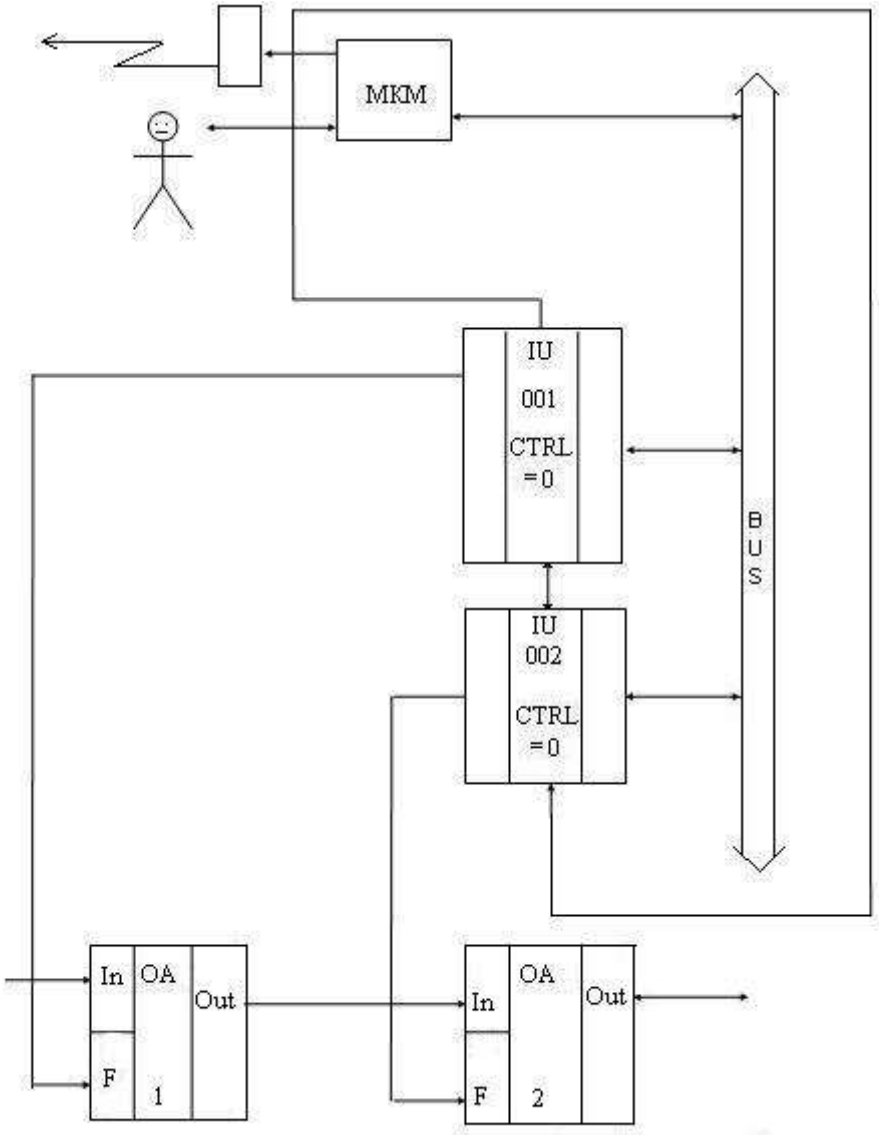

Fig. 2.13. Microcomputer-based system with conveyor managed operating nodes

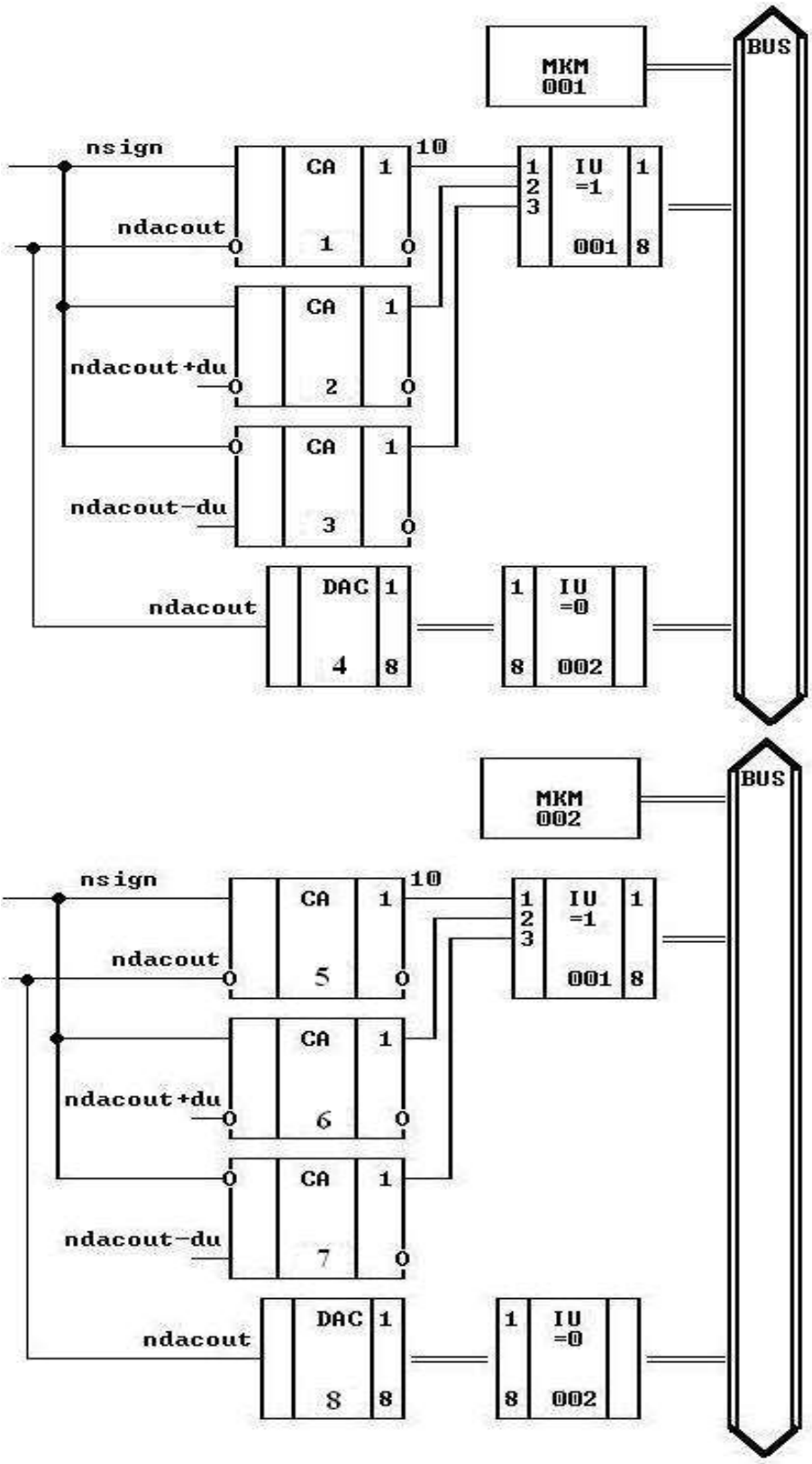

Fig. 2.14. System based on a multiprocessor microcomputer

2.5. Computing systems from network devices and computers

Network programmable devices with sensors form the basis of solutions for the automation of stationary and mobile objects. The processing of analog signals is localized, and data is transmitted digitally. Each device has a network address and access from a network viewer for managing and reading results. The main options for programmable devices based on microcomputers with various levels of hardware support are considered in p. 2.4. The storage and processing subsystems are stationary and are posted on a datacenters (DC), and the collection and preprocessing subsystems may contain stationary and mobile devices and have become known as the Internet devices (IOT). The concentration of information processing and storage continues and will consume a significant share of the generated electricity [64, 107, 110, 143, I31]. Unlike the Internet devices datacenters should be considered as complex objects and standards of information modeling should be applied to them [I28, I31].

The synthesis of a local network, in the form of a formalized task, for solving CAD problems is performed in a CAD COD environment [27, 33]. Network components are user machines, servers, and communications equipment. Users' machines differ in type, workplace characteristics and mobility. Mobility is estimated by the component moving speed (m / s) and is equal to zero for stationary machines. It is not the component that is actually moving, but its carrier. For example, a mobile personal computer (MPC) can be placed at the workplace – the table, while mobility is zero, or in a car, then mobility is equal to the speed of the vehicle. Depending on the position of the machine, its speed of movement, the MPC may or may not be a component of the network. Mobile internet devices (MID) can be used in the workplace, when moving a person, or in a vehicle. Communicator (PPH) is designed for data and voice communications and moves with a person. Servers (SERV), switches (SWITCH), and wireless access points (AP) [4, 7, 10–12, 107, 110], as a rule, are located permanently. Component models are characterized by syntax and semantics tables. Data transmission channels differ in operating principles and comply with international standards [4]. The principle of operation of the data channel must be consistent with the port (output) of the component.

Application servers are associated with technological and testing equipment, and formalized tasks are performed on them [4, 7, 10–12, 107, 110]. Reading content can be hosted on a network server, and when performing a task, it can switch to the selected application server. Stationary facilities are located in accordance with the rules and regulations. Mobile objects form a dynamically changing environment, and they need a special data structure about the state and speed of movement.

2.6. The processes of synthesis and analysis of heterogeneous computing systems

Synthesis procedures allow you to get a description of the object, and analysis procedures – to evaluate the properties of the object and its behavior. For simple objects, the complexity of the analysis is less than the complexity of the description, but with an increase in the complexity of objects, it increases significantly. Therefore, automation of the analysis of complex objects increases the efficiency of design. Structural analysis procedures allow you to evaluate the functions and behavior of the object, composition, compliance of components and relationships, parameters of relationships. Parametric analysis procedures provide information on the statics and dynamics of an object in the time or frequency domain, and sensitivity analysis.

Along with the main analysis procedures, new differential analysis procedures are needed to assess differences in the structure or parameters of objects. An example is the procedure for comparing a group of signals. A computer system as a design object is represented by a set of interacting components, each of which can be considered as a composite object. The model of an abstract object is represented by a data structure whose value reflects its state by methods or operations that transform the states of objects in accordance with perceived input messages. The object model generates output messages or procedure parameter values for other objects.

A description of the design solution sufficient for automated input and interpretation by the formal system is called a formalized task, which is widely used in CAD. Formalized tasks (FT) consist of many sections, and sections (Pi) – of sentences of the corresponding language:

$$FT = \{P1, P2, ..., PN\}. \quad (2.2)$$

Each section is syntactically and semantically homogeneous and corresponds to a specific relation. As an example, consider the research CAD software COD, in which the formalized task consists of the sections presented in table 2.5.

The INPUT section describes the relationship between circuit numbers, signal type, and start and end cycles of a signal. Section information can be presented in text form.

The UNIT section describes the relationship between types and numbers of components, numbers of circuits connected to outputs and inputs, and the description is made component-wise. Section information can also be presented in text form.

Table 2.5

Formalized job sections

Section name	Section Description
INIT	Initial installation
INPUT	Description of external influences
UNIT	Device description
CTRL	Control
MOD	Models of new components

Functional models of MFComp components consist of PMF sections:

MFComp = <PMF1, PMF2, ..., PMF5>.

Sections of the description of functional models are given in table 2.6.

Table 2.6

Component functional model sections

Section name	Section Name (label)	Section Description
1	FINIT	Validation of parameters, placement of data structures and their filling
2	FTYP	Control of component types and selection of possible implementations
3	FS	Component state recovery
4	FOUT	Formation and control of output signals
5	Fend	Release of resources

Specific functional models of the components are given in the second, third section.

Consider the features of textual and tabular descriptions. The tabular form allows you to describe a specific relationship and is used in low-level CAD, for example in PCAD CAD [22], and as an intermediate level of one technical solution in CAD COD. The text form is used in the COD research system, the integrated CAD system PRAM5.3, modeling packages [1, 7, 13, 120–122] and descriptions of computing devices [13]. The text form in low-level languages allows one variant to be described for a specific technical solution and is equivalent to a tabular form. A text form using a high-level language allows us to describe both specific technical solutions and many technical solutions. Examples of descriptions in the high-level languages PL / 1, C ++, JAVA, and ADA are given [27, 31, 113, 114], and in the VHDL language, in [8, 115].

However, the full specification of the components and connections does not allow us to present many design solutions in one description. The introduction of variable types of components and compounds allows you to combine many design solutions in one formal task.

The inclusion relation and the basic properties of elements with an indication of a normative document are usually combined in the homonymous section of a formalized task called a list of elements. The relationships between the components are reduced to the presence or absence of a galvanic coupling between the contacts or optical coupling between the ports. These relationships can be written by component or by chain. In the description of the components, each type of component corresponds to a certain set of circuits with which the contact and the component are galvanically connected. This relationship can be described using the rules of a language. Well-documented standard programming languages [1, 9, 10, 29]. For example, you can use non-procedural or tabular forms. Non-procedural form is usually used at the technical design stages, A procedural form is often used for functional or logical analysis of the operation of devices or systems. Universal programming languages are familiar to students from the first year, which facilitates the development of syntax descriptions. For the translation and interpretation of descriptions, only basic computer software is used, supplemented by the rules for constructing and placing sections of the formalized task, the rules of component-wise description of the structure of the device, and software models of components. External influences are described by input circuits and interpreted by special modules. For the translation and interpretation of descriptions, only basic computer software is used, supplemented by the rules for constructing and placing sections of the formalized task, the rules of component-wise description of the structure of the device, and software models of components. External influences are described by input circuits and interpreted by special modules. For the translation and interpretation of descriptions, only basic computer software is used, supplemented by the rules for constructing and placing sections of the formalized task, the rules of component-wise description of the structure of the device, and software models of components. External influences are described by input circuits and interpreted by special modules.

2.7. Principles of building multi-level systems for modeling and designing computing systems

During the design process, the original description is converted to the required description – design solution. To specify design decisions, additional information is required. The initial description in multi-level CAD is a formalized task. A formalized task using the capabilities of a high-level language provides a description of many technical solutions. In industrial CAD [8], the project is one technical solution. Therefore, in a multilevel CAD system, between the upper level, where many technical solutions can be represented, and the lower level, which represents one technical solution, there should be a middle level with full information about the technical solution variant and the parameters of the components.

Multi-level CAD COD is used for the synthesis and analysis of many options for structures and their automatic conversion into many design solutions for industrial CAD. And consists of many subsystems:

$$COD = <HSC, COMM, SAT, AAT>, \quad (2.3)$$

where HSC (Human Control) – the design management subsystem serves to facilitate the user's work when moving to the second level of complexity of design tasks; COMM is a communications design subsystem, provides the ability to design objects on the Internet; SAT (Synthesis Automation Tools) – tools for automated synthesis of objects; AAT (Analysis Automation Tools) – tools for automatic behavior analysis, resource estimation and object comparison.

The HSC design management subsystem consists of many subsystems:

$$HSC = <SETSEL, PRJSEL, RESSEL, RPRJ>, \quad (2.4)$$

where SETSEL is a subsystem for selecting a formalized task, project description languages, CAD selection and description type for import, CAD selection and interface type for export, message language and server on the network; PRJSEL – subsystem for selecting design results; RESSEL – subsystem for selecting the presentation of results; RPRJ – rules for choosing the sequence of design procedures and operations, design routes for obtaining design and analysis results. As the complex develops, the proportion of rules implemented in the synthesis and analysis subsystems increases. For example, the analysis subsystem automatically compares estimated and actual results, as well as object descriptions and required resources.

The CAD COD user selects the desired design or analysis result, rather than the sequence of design procedures and operations to achieve the goal. Thus, the load on the user is reduced and the level of intelligence of the complex is increased [31, 70, 87, 126, 127]. The formalized part of the design management subsystem is presented in the form of a COD CAD shell, which can be implemented by various means. In [29, 33, 11–13], shell implementations for various operating systems based on a multifunctional editor included in the IBM Visual Age tools, network browsers (Mozilla Firefox, Google Chrome) and the ECLIPSE tool environment are presented.

Multivariate analysis, assessment and selection of the required solution are possible when managing external influences, types of components and their performance, the presence and absence of compounds. The means of a universal high-level language make it possible to solve similar problems at the stages of functional and structural design.

During the design process, a formalized task in a high-level language, representing many technical solutions, is converted into a text or table form of the level of one solution option, which can be presented in text, command or table format for a specific CAD or as a formalized task (FT) [29–33]. The number of options is determined by the source (Federal Law).

Building a system of many independent modules, a basic model – a glossary of terms, and an information model that connects well-known terms with the rules for executing modules is called function-oriented design [87].

The software-methodical complex COD in expression 2.3 contains a complete set of components. Server and client versions with a part of the components or client versions with the addition of components are possible as the CAD network services increase. The information part of the complex can be located on a network server (for example: e.sfu-kras.ru), and the execution of tasks is transferred to one of the application servers. Application servers may differ in operating systems (Windows, OS2 or ECS, LINUX, VM) and hardware. Network service standards allow customers to complete tasks on any server offered in the menu.

The client must contain a viewer with which he selects a formalized task on the server and executes it. To view the simulation results, he needs a program for viewing time diagrams, to view PCAD schemes free program VIEWER version of PCAD, for executing batch files of a specific CAD – CAD package (PCAD, ORCAD, CATIA, AUTOCAD EAGLE, CADDy). A schematic diagram can be seen in the viewer if the JAVA package is installed, and a three-dimensional model with changes in signals or activity – if there are virtual reality plug-ins or X3D for viewers. To reduce the amount of data transferred, the result is transmitted and stored in symbolic form and converted into an image at the request of the client. The optical disk contains the COD software and methodological complex, examples for performing FA, demo versions of CAD and viewing programs [33].

To perform tasks on CAD servers, a significant number of operations and energy are required. Limited operational and energy resources of mobile clients stimulate the development of CAD services.

CAD system the design and technological design can be used in the dial-up or package mode. For example, dial-up CAD on the basis of a PC can serve as PCAD [120–122] and its development GRIF-4. In spite of the lack of dialogue in the integrated industrial CAD program, 5.3. [30, 101] in it the agreement of the systems realizing various stages of design. Therefore, the preference should be given to the integrated CAD with the agreement of the separate stages, which will lead to a “no-go” cycle.

The introduction of APM of the first generation did not change the technology of the project, but the transition to the integrated CAD system allows for the development of a room for food and drink. The increase in the efficiency of the use of the produced objects is achieved by the improvement of the design solutions by the synthesis and analysis of a large number of options. Automation even of separate stages of project implementation in the framework of complex CAD allows to reduce time and costs of one solution, which ensures a lot of work.

2.8. Descriptive Conversion Information Basics in multi-level CAD

CAD COD allows you to automatically evaluate resources for specific types of components. It is possible to estimate resources for many specific types of components in a multivariate analysis. Variable component types are arrays of specific values, the dimension of which is determined by the number of options

(NVARMAX). Resource estimates are derived for each option individually and for all options as a whole for each analysis of a formalized task. To calculate the performance criteria, you need to evaluate the performance of the subsystem (Mbps). The performance value is assigned to the PP variable of a real type (float). We give an example of resource estimation of the first variant of the transfer scheme in the formalized task fp02s. The circuit performance is estimated by changing the output signal specified by the nppd digital signal number (vext.nppd for Java).

The signal is digital 11. The number of changes is 2.

Total power consumption, MBt	GPF	20
Total chip area, mm2	GS	65
The total mass of microcircuits, g	GM	3
Total cost, rel. units	GC	3.2
Performance Mbps	PP	2.5
Cost / performance, rel. units / (Mbps)	KEFC	1.28
Mass / Performance, g / (Mbps)	KEFM	1.2
Power / Performance, mJ / Mbps	KEFP	8

Performance is determined using tests [1, 13, 105, 107], by the useful work performed per unit of time or by changing a digital or analog signal for a given time interval. When evaluating performance, the number of the changing digital (nppd) or analog (nppa) signal is set.

It is allowed to simultaneously set the numbers of digital and analog signals to assess the performance, which is calculated and displayed for both signals. The performance criterion is calculated by the performance of the analog signal. For an analog signal, the sum of the changes should be multiplied by the effective bit depth (lra), defined as the binary logarithm of the total relative error (era):

$$lra = \log_2 (era).$$

The initial values of effective bit depth and error have an initial value minus one (–1), which cannot be really assigned and indicates an error, since the performance takes a negative value. The effective bit depth is evaluated in the component model when the circuit number of the output analog signal matches the number of the performance evaluation signal (nppa).

The algorithm for evaluating the effective bit depth and performance for changing the analog signal is shown in fig. 2.15. The algorithm allows us to estimate the effective bit depth of the analog signal both in the case when the cycle duration exceeds the transient completion time and the main error is the static error, and in cases where the cycle duration is less than the transient completion time.

For a digital signal, the number of its changes (KDOUT) is estimated, excluding surges and noise, the duration of the time interval (equal to the

difference between the end ticks (NTEND) and the start (NTBEG) clock interval) and the beat (DELT). Thus, informational signal performance in Mbps

$$PP = 103KDOUT / (NTEND – NTBEG + 1) * DELT.$$

To evaluate the information performance of digital and analog signals by changing them over a given time interval according to algorithms compiled by the user, the functions Sign PPD and Sign PPA are introduced in the languages C ++, PL / 1, JAVA. The syntax and semantics of the parameters of the SignPPD and SignPPA procedures are given in table 2.7.

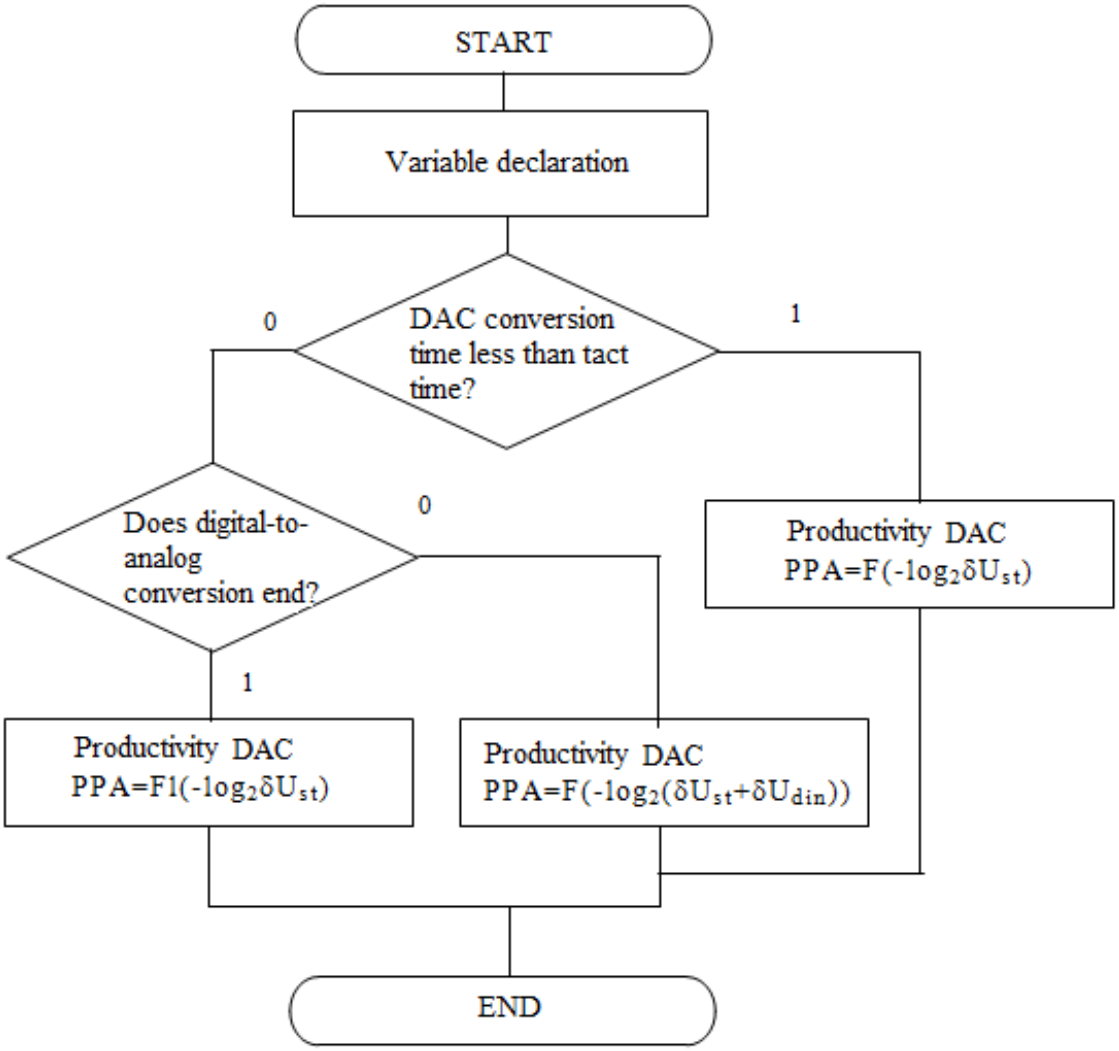

Fig. 2.15. Flowchart of an algorithm for evaluating the performance of an output signal of a digital-to-analog converter (DAC)

Table 2.7

Syntax and semantics of SignPPD and SignPPA function parameters

Room	Identifier	Data Type PL / 1	A type data C, C ++, JAVA	Data type ADA	Semantics	Note
1	NSIGNAL	DEC FIXED (3)	Int	Integer	Signal Number	–
2	TYP	CHAR (*)	char (C ++) String (Java)	Character	Component Type	Type can be abstract and concrete
3	NTBEG	DEC FIXED (6)	Int	Integer	Beginning beat number	–
4	NTEND	DEC FIXED (6)	Int	Integer	End of measure beat number	–

Using the proposed function names and parameters for evaluating performance ensures the implementation of interfaces for the Federal Law with a specific CAD system. Examples of the digital signal performance evaluation function in universal programming languages are given below. The dictionary of performance evaluation procedures is presented in table 2.8.

In PL / 1, the SignPPD procedure is located in the SignPPD.inc file:

```
SIGNPPD: proc (NSIGNAL, TYP, NTBEG, NTEND);
DCL TYP CHAR (*); DCL (NSIGNAL, KDOUT INIT (0) STATIC)
FIXEDEC (3);
DCL (NTBEG, NTEND) FIXED DEC (6);
IF (PPT & (NT> NTBEG) & (NT <= NTEND)) THEN
IF (OUT.NEX (NSIGNAL) ^ = OUT.OLD (NSIGNAL)) THEN
KDOUT = KDOUT + 1; / * ACCUMULATION OF CHANGES * /
IF (NT = NTMAX) & PPT THEN DO;
PP = KDOUT / ((NTEND-NTBEG + 1) * DELT * 1E-3);
PUT SKIP EDIT ('SIGNAL', NSIGNAL) (A, F (6));
PUT SKIP EDIT ('NUMBER OF CHANGES', KDOUT) (A, F (6));
PUT SKIP EDIT ('PP-PERFORMANCE MBIT / C', PP)
(A, E (14.3.6)); END
END SIGNPPD;
```

SignPPD functions in C ++ and Java are in the control.cpp and control.jav files.

An example of a SignPPD function in C ++ is given:

```
void SignPPD (int nsignal, char * typ, int ntbeg, int ntend)
{static int kdout;
if (nt == ntmin) kdout = 0;
if ((nt> = ntbeg) && (nt <= ntend))
if ((ppt == 1) && (out [nsignal] .Nex! = out [nsignal] .Old))
/ * comparison of current and previous signals * /
kdout = kdout + 1; / * accumulation of changes * /
if ((nt == ntmax) && (ppt == 1))
{pp = kdout / ((ntend-ntbeg + 1) * delt * 1e-3);
printf ("\ n signal% d number of changes% d", nsignal, kdout);
printf ("\ n pp-performance Mbit / c% e", pp); }
} // End SignPPD
```

An example Java SignPPD function is provided:

```
public static void SignPPD (int nsignal, String typ, int NtBeg, int NtEnd)
{int kdout, kppd = 0;
if (vext.nt == vext.ntmin) vext.pp = 0;
if ((vext.nt> = NtBeg) && (vext.nt <= NtEnd))
if ((vext.ppt == true) && (vext.out.Nex [nsignal]! = vext.out.Old
[nsignal]))
/ * compare current and previous signals * /
vext.pp = vext.pp + 1; / * change count * /
if ((vext.ppt == true) && (vext.nt == vext.ntmax))
{if (vext.nppd> 0) kppd = vext.tpc [vext.nppd] [7]; kdout = (int) vext.pp;
vext.pp = (float) ((kppd / ((NtEnd-NtBeg + 1) * vext.deltn * 1e-9)) * 1e-6);
System.out.println ("pp-performance Mbit / c" + vext.pp);
System.out.println ("signal" + vext.nppd + "number of changes" + kppd);
}} // End SignPPD
```

The main approach is the one-time input of the objects in the form of formalized tasks for various level analysis and end-to-stage. Accurate to the engineer graphic documents in the form of a schematic and random drawings should be obtained automatically as a result of the interpretation of formalities.

However, the computing system, implemented in accordance with the resulting decrease in health, is subject to the possibility of a significant increase in the incidence of chronic (non-existent) disorders. Differences appear on the external system outputs and may have a random or systematic character.

The reason for the differences is a change in the actual state of the structure of the object, which is somewhat synthesized. According to the accepted form, the differences can be caused by a change in the number of components or by the functions performed by them, as well as by a change.

Table 2.8

Glossary of performance evaluation procedures

Room	Name variable	Appointment	Data type	The size-nost
1	PPA	Analog signal performance	float	Mbps
2	PPDAC	Digital to Analog Converter Performance	float	Mbps
3	PPD	Digital signal performance	float	Mbps
4	KDOUT	Number of digital signal changes during analysis	int	–
5	SPPA	Sum of analog signal changes during analysis	float	–
6	TDAC	Signal Conversion Time	float	Ns
7	DELT	Tact duration	float	Ns
8	F	Main frequency	float	MHz
9	Fl	DAC clock speed	float	MHz
10	δUst	Relative Static Accuracy DAC	float	%
11	δUdin	Relative Dynamic Accuracy DAC	float	%
12	δU	Relative Total DAC Error	float	%

Analysis procedures should allow for the implementation of changes in the environment. The synthesis of variable descriptions of objects, as opposed to the main ones, can be called differential. Differential synthesis procedures allow you to receive a description of the object for a multiple analysis, to make changes to make a new solution from the existing one. A description of the differences can be conveniently used in a multivariate analysis of computing systems.

2.9. Formalized task and algorithms for automated analysis

Researches of heterogeneous computing systems designed for the automatic analysis of the functioning of computing devices and systems.

AFTER AUTOMATED ANALYSIS METHODS, the main nodes of the computing devices are investigated. The cycle ends with the study and optimization of the computing device – a node of a higher-level computing system. The degree of automation of the analysis will increase gradually by improving the tasks with the obligatory improvement of the principles of work.

Algorithm flowchart for an automated analysis is shown in fig. 2.16. For the best satisfaction of the algorithm, it is recommended to manually solve the system of logical equations in the binary and triple bases of the result of coincidence.

We study the generation of commonly used types of signals: with a variable time shift and the number of pulses for control and synchronization devices, with a variable frequency and duty cycle for external devices. Generation of non-periodic and periodic signals is carried out by a special procedure using the bit line of the signal model. Different versions of the signal models are generated by the algorithm, obtained using the model of the device or read from the data set.

The results of the signal generation are used in external input circuits of the circuit used. A simple circuit from logical elements is considered. As a result, the user is familiar with the generation of signals, as well as with the features of an automated analysis.

In the presentation section of a system or device, a component-wise description of all options is performed and an object-component methodology for implementing systems for all applications is developed [30, 31, 33, 39, 40]. A section consists of generalized components specified by an abstract or concrete type of component.

A multivariate analysis of the computing device with the simplest way to control external drives – by reading the bit lines from the readable control unit is used. The algorithm diagram of a multi-variant analysis is shown in fig. 2.17 and implements the IPRINT monitor, the algorithm of multivariate analysis with the output of analog signals with the APRINT monitor, with the output of analog arrays with the APRINTM monitor.

A multivariate analysis can also be performed using a simple monitor with reading data from one or various sections of the library due to complication of the task only. The model is implemented with the help of the internal procedure, in which the exclusion of the forbidden signals of signals must be provided.

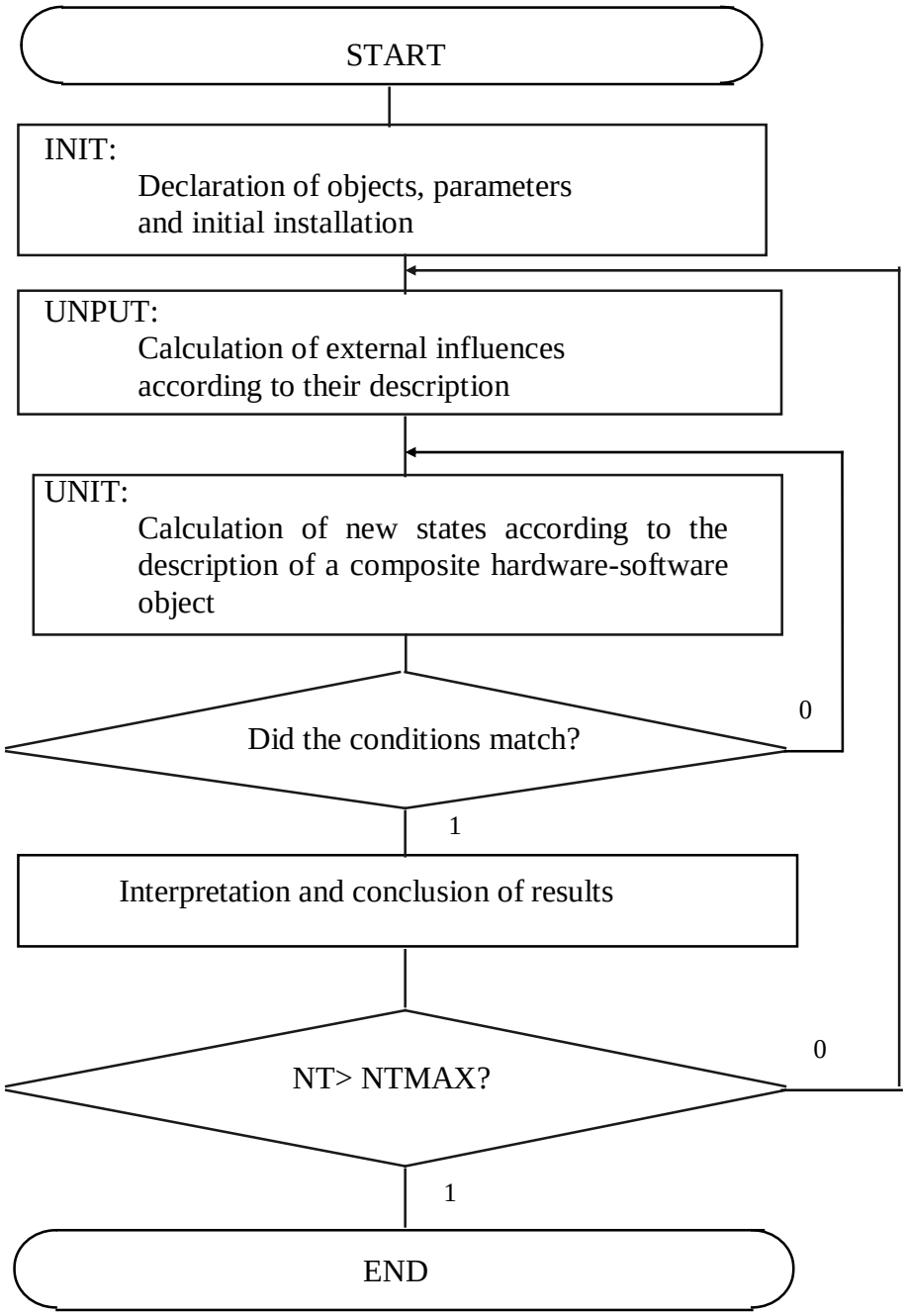

Fig. 2.16. Block diagram of a univariate analysis algorithm

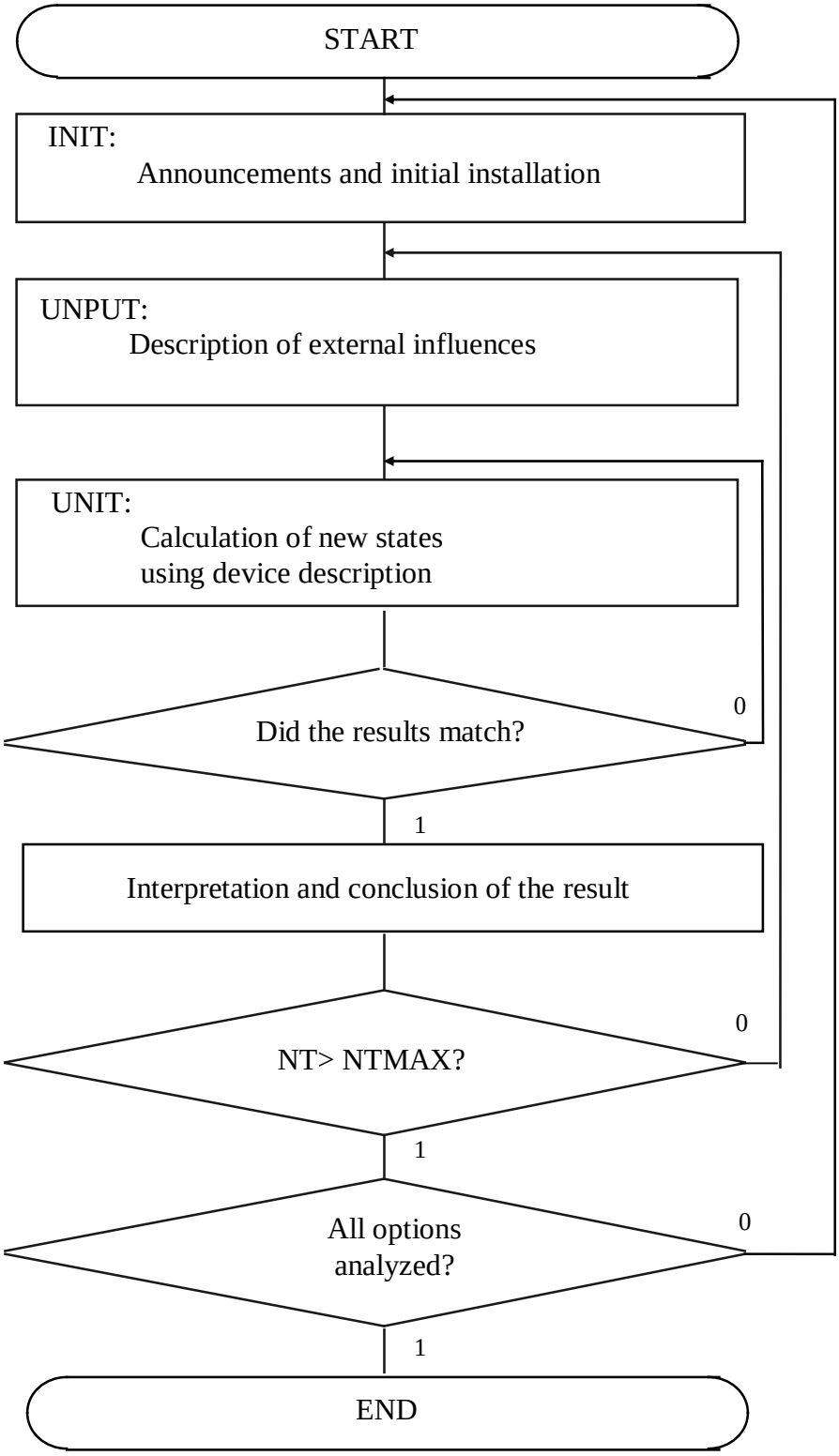

Fig. 2.17. Multivariate Analysis Algorithm Flowchart

Next, macromodels and micromodels of components and nodes are investigated. For example, when analyzing a multiplexer, a macromodel and a micromodel are composed of logic elements, the models are examined under identical conditions, the results are compared and entered into the table of circuits, channels, and results. As examples, one-way and many-way multiplexers, address comparators, code converters are used.

The analysis of devices with a variable structure was performed depending on the uncertainty of the external environment [17, 27, 45, 67, 87, 89, 97, 155]. The structure of the device can be changed statistically (by changing the modules that perform different functions) or dynamics (depending on conditions and situations). Static changes to the structure of the device or system can be entered into the memory and analyze all the options. Dynamic change of the structure is implemented by hardware (with the help of controlled commutated elements) and software (in the section of determination of the coefficients). For example, it is possible to describe a conditional parameter that is suitable for breaking or shorting circuits. The conditions of the time, the combination of the values of the signals or the comparison of the signals can be implemented with the help of the conditions and the choice of universal languages, 119–4,4.

It is possible to use the means of the automatic assessment of the state of all the signals. This allows you to leave the most informative signals on the diagram.

The means of comparing the signals in the predetermined time intervals can be used to evaluate the identity of the macromodel and a micromodel. The results of assessing the status and comparing the signals are displayed in a tabular form in accordance with the level of results given in table 2.9.

The structure based on the microcomputer without operational nodes is analyzed. The processing of information is done programmatically, and the productivity of such a system is less than that of a system with hardware support. The efficiency of this structure will increase with an increase in the degree of integration of components single-crystal microEBMs.

Multichannel digital processing of signals is considered on the basis of microEBM with various hardware support. Minimum hardware support is used in the structure with a common operating unit and an input multi-channel commutator. Operational processing of signals for various channels is carried out by the general operating unit in the time division mode. The results of the operational processing are recorded in the memory of microEBM. The final processing and formatting of the results in a user-friendly form is performed using microEBM.

In the following sections, structures with a gradual increase in hardware support are used. The examples of automated analysis of the structures with the parallel processing of signals along all channels are given. The final is the conveyor structure.

Table 2.9

Results Level Table

Value level result (LRES)	Name of additionally submitted results
–5–0	Character State File for Graphic Display
–1, 1	Timing chart and resource assessment
–2, 2	Comparison Chart for Estimated and Actual Digital Signals
–3, 3	Comparison table of digital and analog signals. Displaying messages for debugging interfaces
–4, 4	Digital signal status table output
–5, 5	Optimization

To facilitate structural optimization of computing devices, automatic resource estimation is provided. The performance assessment is performed by the user, and the values of the main performance criteria are calculated automatically.

The methodological and software of research CAD is based on the principle of a single description of a technical solution and the minimum labor costs of an engineer or student for use in various applications. As applications, we consider the behavior of a computing device or system, the technical design of computer modules in various CAD systems: PRAM5.3, PCAD, ALTIUM DESIGNER, EAGLE, ORCAD, CADDY, and CATIA [11–110].

The interface with design automation systems is mostly automatic. As a CAD system of a design level, the domestic system PRAM5.3 and foreign systems PCAD, ALTIUM DESIGNER, CADDy and ORCAD are used. The interface provides the conversion of formalized tasks with a description of one and many variants of computing devices. Information support of the interface for concretization of formalized tasks is implemented using tables.

Representations of object-component models of many technical solutions of computing systems at various levels of abstraction in high-level languages in various syntactic environments and algorithms for their construction for various applications are created. The study develops an object-component methodology for describing systems for all applications. A new combination in one description of computing systems at various levels of abstraction and algorithms for their conversion in various syntactic environments confirms the advantage of semantics over syntax, reduces the volume of descriptions, allows for conceptual unity and reduces the possibility of environmental error.

Designing is carried out with incomplete information about the object and the external environment. Information is supplemented at each iteration of the design of the object. An analysis of the difference between the estimated and actual results and the performance criteria allows you to create the rules for the synthesis of the object. The main attention should be paid to systems with a forecast of the signal interval or with a model of the object. With the accumulation

of knowledge about signals and objects, the flow of input information decreases and the flow of predictive estimates of signals increases.

At the circuit level, the switching energy and through currents, temperature conditions and error analysis are determined. At a logical level, synthesis and optimization of computing devices is performed. The design of a heterogeneous computing system is carried out using two levels of representation: logical and register transfers. A computing system, including a network fragment, is represented at the levels of services, messages, or packets.

3. MULTIFUNCTIONAL COMPONENT MODELS

3.1. Component Abstraction Levels

At the upper level of abstraction (see fig. 1.1), functional models of components as a whole are considered, which are called macro-models of components and are described by the functions of outputs and transitions. In various CAD systems, functional models of components are presented in various forms and in different languages [14, 87, 132]. For example, in paragraph 3.2, models of components in CAD "COD" are presented, and models of components of heterogeneous systems are described in [14, 132].

At the next level of abstraction are structural models that reflect the internal structure of components. Such models will be called micromodels. At this level, resources can be estimated.

The same structure can be implemented using different principles of action. Therefore, one structure of a computing system can be implemented on various physical principles (electronics, optics, pneumonics) that are resistant to different environments. The physical principles of action of computer elements are described in detail in [1, 6, 11, 16, 29, 59] and will be considered only when assessing resources or optimization.

Knowing the principle of action, taking into account the level of technology and technology, you can go to the level of technical solution. The technical solution should be feasible. After receiving the technical solution, it is possible to change the parameters of the solution in order to improve the quality of the product. Such examples are discussed in the section of parametric optimization.

Having determined the parameters of the technical solution, we can proceed to the design of the component. For example, depending on the power dissipated by the crystal, a housing will correspond to it. The design is significantly affected by the external environment (operating conditions). Thus, at the upper level of abstraction are functional models of components, and at the lower level–component designs and their geometry. Geometric models of components are used in the design [6, 134].

Component models can be used at various levels of abstraction. For example, functional models can be used to automatically analyze the behavior of an object; component parameter models–to evaluate the implementation options of the device. A number of component models are used to interface with design engineering systems that are at a different level of abstraction.

Let's consider functional models of components.

3.2. Component Models

Component models can be functional, geometric. The geometry model of the component can correspond to the symbol in the circuit diagram or the geometry of the construct.

Functional models of MFComp components consist of sections [39, 40, 41, 49] PMF:

MFComp = <PMF1, PMF2, ..., PMF5>.

Sections of the description of functional models are summarized in table 3.1.

Table 3.1

Component functional model sections

Section name	Section Name (label)	Section Description
1	FINIT	Validation of parameters, placement of data structures and their filling
2	FTYP	Control of component types and selection of possible implementations
3	FS	Component state recovery
4	FOUT	Formation and control of output signals
5	Fend	Release of resources

For each class of components, models are developed with a common data structure. In the table 3.2 the main classes of components, the names of procedures and tables of syntax and semantics of parameters are summarized. In the table 3.3–3.8 shows the syntax and semantics of the main classes of components.

Mandatory in the general data structure are the type of component, its number and data corresponding to the names of the output and input circuits. First, the numbers of the output, and then the input circuits are described. The state of the signals whose numbers correspond to the numbers of the circuits are stored in the global data structure. For input circuits, the procedure only reads information about the signal state, and for output circuits, the procedure records new states.

In order to reduce the description volume for each class of components, not one, but two main data structures are used. The first data structure uses arrays of input or output circuit numbers. In this case, there are no restrictions on the order of the circuit numbers. For the second structure, only the high-order circuit number (minimum number value) and the number of circuits are indicated. But in this case, the chain numbers should be arranged in ascending order, and the procedure fills the array of circuit numbers, freeing the student or engineer from this. Names for the first entry point in the form of an array of circuit numbers are formed from the name of the component class and the end of the NIN, and for the second entry point – MIN termination (minimum).

Table 3.2

Component Models

№	Name	Name the procedures	Table name parameters
1	Digital signal generation	SIGNAL	HSIGN
2	Multivariate Analysis Management	IPRINT	HIPRINT
3	Analysis process management	APRINT	HAPRINT
4	Generalized logic element	LO	HLO
5	Trigger	MTJK	HMTJK
6	Multiplexer	MX	HMX
7	Multi-bit Multiplexer Assembly	MMX	HMMX
8	Serial Totalizer	MCT	HMCT
9	Serial reversing adder	MCTR	HMCTR
10	Parallel Interface Node	MIR	HMIR
11	Serial interface node	MIS	HMIS
12	Programmable logic matrix	MLO	HMLO
13	Processor model	MPU	HMPU
14	Analog processor	MPA	HMPA
15	Threshold element	PE	HPE
16	Analog Comparator	CA	HCA
17	Digital to analog converter	DAC	HDAC
18	Analog to digital converter	ADC	HADC
19	Multiplexer universal	MXU	HMXU
20	Majority element	MAJ	HMAJ
21	Sequential Register	RPP	HRPP
22	RAM node	RAM	HRAM
23	Encoder and Decoder	DC	HDC
24	Digital adder	SUMD	HSUMD
25	Timer	TIM	HTIM

Table 3.3

Syntax and semantics of the IPRINT control module

№	Identifier	A type data PL/1	Data Type C, C ++, JAVA	A type data ADA	Semantics
1	NSMIN	DEC FIXED (3)	int	integer	Minimum Signal Number
2	NSMAX				Maximum Signal Number
3	NTMIN	DEC FIXED (6)			Signal start beat number
4	NTMAX				The number of the cycle end of the signal output

Table 3.4

Syntax and semantics of the APRINT control module

№	Identifier	Data type PL/1	Data typeC, C ++, JAVA	Data type ADA	Semantics
1	NSMIN	DEC FIXED (3)	int	integer	Minimum Signal Number
2	NSMAX				Maximum Signal Number
3	NTMIN	DEC FIXED (6)			Signal start beat number
4	NTMAX				Signal End Cycle Number
5	NAMIN				Minimum Analog Signal Number
6	NAMAX				Maximum Analog Signal Number

Table 3.5

Syntax and semantics of input signal generation procedures (SIGNAL, SIGNALD)

№	Identifier	Data type PL/1	Data type C, C ++, JAVA	Data type ADA	Semantics
1	NSIGNAL	DEC FIXED (3)	int	integer	Signal Number
2	STRBIT	BIT (*)	Bitstring	Character string	Bit string
3	NTMIN	DEC FIXED (6)	int	integer	Signal start measure number
4	NTMAX				Signal change ending beat number

Table 3.6

Syntax and semantics of generalized parameters logic gates LO

№	Identifier	A type data PL/1	Data type C, C ++, JAVA	Data type ADA	Semantics	Notes
1	TYP	CHAR (*)	char (C ++) String (Java)	character	Element type: min and max of the analog signal <MIMAA>, for digital signals, the keys are given in the note	Keys: '&', '|', 'LA', 'LE', 'LI', 'LP', 'M2'

Continuation of the table 3.6

№	Identifier	A type data PL/1	Data type C, C ++, JAVA	Data type ADA	Semantics	Notes
2	NEL	Dec FIXED (3)	int	integer	Item number	–
3	NOUTP				Non-inverting output	–
4	NOUTN				Inverting output	–
Difference for LONIN entry point						
5	NIN	(*) DEC FIXED (3)	IntVector (C ++) Int [] (Java)	integer	Array of input numbers	–
Difference for LOMIN entry point						
6	NMIN	Dec FIXED (3)	int	integer	Min entry number	–
7	NSIGN				Number of inputs	–

Table 3.7

MTJK trigger parameter syntax and semantics

№	Identifier	A type data PL/1	Data type C, C ++, JAVA	Data type ADA	Semantics	Notes
1	TYP	CHAR (*)	Char (C ++) String (JAVA)	character	Item type	564TV1 – key Tv
2	NTJK	Dec FIXED (3)	int	integer	Item number	–
3	NOUTP				Trigger output	–
4	NOUTN				Trigger output inverting	–
5	NINR				Zero Installation Input Number	'11'V setting
6	NINS				Unit Installation Input Number	'11'V setting
7	NINC				Clock Number	Work on '01'V
8	NINJ				Input Number J	–
9	NINK				Entrance Number K	–

Table 3.8

Syntax and semantics of multiplexer models digital and analog signals (MX and MXU)

№	Identifier	A type data PL/1	Data type C, C ++, JAVA	Data type ADA	Semantics	Notes
1	TYP	CHAR (*)	Char (C ++) String (JAVA)	character	The type of multiplexer KP is designed for switching digital signals, KN– for analog	564KP1 564KP2 K561KP1 K561KP2 K1561KP1 K1561KP2 K555KP: 2, 7, 12, 16 155KP: 1, 2, 5, 7 K590KN1 K590KN6 K591KN3
2	NEL	DEC FIXED (3)	int	integer	Item number	–
3	NOUTP				Exit number	–
4	NOUTN				Output number inverting	–
5	NC				Permission Entry Number	–
6	NA	(*) DEC FIXED (3)	int	integer	An array of address transfer chain numbers	The dimension of the array must match the type of multiplexer
7	JIN		IntVector (C ++) int [] (Java)		Array of input circuit numbers	–
MXMIN Entry Point Features						
8	NMIN	DEC FIXED (3)	int	integer	Min entry number	–
9	NSIGN				Number of input circuits	–

The data structures for these entry points should be syntactically different, and for their use, entry point description files should be included in the formalized task. For example, for the PL/1 environment [5, 14], files have a name made up of the W character and a common procedure name, while for C ++ a file of the*. H type is included. For C, there is only one entry point.

Here are examples of descriptions for choosing an entry point for modeling a universal logic element: in PL/1:

- DCL (LONIN ENTRY (CHAR (*), DEC FIXED (3), DEC FIXED (3);

- DEC FIXED (3), (*) DEC FIXED (3)), LOMIN ENTRY (CHAR (*);
- DEC FIXED (3), DEC FIXED (3), DEC FIXED (3), DEC FIXED (3);
- DEC FIXED (3)));
- DCL LO GENERIC (LONIN WHEN (....), LOMIN WHEN (....))

in C ++:

- void LO (char * typ, int nel, int noutp, int noutn, IntVector & jin);
- void LO (char * typ, int nel, int noutp, int noutn, int ninmin, int nsign);

on JAVA:

- public static void LO (String typ, int nel, int noutp, int noutn, int ninmin, int nsign);
- public static void LO (String typ, int nel, int noutp, int noutn, int [] mnin);

The semantics of the parameters are given in table 3.6.

The component model consists of a number of sections and, along with the implementation of the basic functions, issues diagnostic messages.

In the first section, in accordance with the class of components, data structures are declared and, regardless of the entry point used, the arrays of numbers of input and output circuits are filled. It is advisable to declare arrays of arbitrary dimension, followed by determining the sizes of the actually transmitted arrays for each dimension.

The second section of the procedure (FTYP) checks the validity of the component type and determines the type and parameter of the function being performed. An error message is displayed for an invalid component type. After checking the component type, the operator with the FPAR tag remembers the type and number of the component by calling the FPAR function. The type and number of the component are then used to automatically evaluate the device parameters using the FPARSUM procedure. Information about the parameters of the components is stored in the table UIPCAD.DBT. This table also contains the parameters of specific types of components for which power consumption, cost, area, mass are known. Abstract types of components allow arbitrary bit depth and an arbitrary number of inputs / outputs, but do not have specific parameters. Abstract types of components are replaced by concrete when defining the element base.

The third section (FS) is used to assess the status of the component, and the fourth section is (FOUT) – to calculate new output values.

In the fifth section (FF), the validity of the received output signals is evaluated and, if necessary, the forbidden values are replaced with valid ones. For example, for a ternary model, signal levels are indicated: C0 – zero level; C1 – level of a single state; CF – front; CZ – designation of a prohibited state. The use of such designations facilitates the transition from one language environment to another. The name of the ending section is FEND.

3.3. Representation of models in CAD COD

An example of a procedure of a generalized logic element for digital and analog signals in PL/1 with the common name LO and the name of the entry point LONIN for combined unordered circuits (an array of circuit numbers) and the name of the entry point LOMIN for the combined ordered in ascending order of circuit numbers with circuit number parameters senior level and the number of input circuits is given below:

```
LONIN:  PROC(TYP,NEL,NOUTP,NOUTN,JIN);
          DCL(NOUTP,NOUTN,NEL,I,J,II)DEC FIXED(3);
          DCL(JIN(*),NSIGN,MNIN,NMIN)DEC FIXED(3);
          DCL FPAR ENTRY;  DCL TYP CHAR(*);
          DCL NEWI BIT(2) INIT('11'B), FB BIT(4) INIT('1111'B);
          DCL NIN(MNIN)CTL DEC FIXED(3); MNIN=DIM(JIN,1);
          ALLOCATE NIN;NIN=JIN;GOTO FTYP;
LOMIN: ENTRY(TYP,NEL,NOUTP,NOUTN,NMIN,NSIGN);
          MNIN=NSIGN;          ALLOCATE NIN;
          DO II=1 TO NSIGN;NIN(II)=II+NMIN-1;END; /*LOMIN*/
          DCL NTYP INIT(13) DEC FIXED(3), MTYP(NTYP) CHAR(10)
VAR INIT ('&','AND','ЛА','ЛИ','|','OR','ЛЕ','ЛЛ','M2','=','ИП','#','MAJ');
FTYP: /* АНАЛОГОВАЯ ЧАСТЬ */ ;
          DCL ( AMAX, AMIN ) DEC FLOAT;DCL IA DEC FIXED(3);
          IF INDEX(TYP,'MIMAA')>0 THEN DO;
          AMIN=AS(NIN(1)).ANEX; AMAX=AS(NIN(1)).ANEX;
          DO IA=1 TO MNIN; AMAX=MAX(AMAX,AS(NIN(IA)).ANEX);
          AMIN=MIN(AMIN,AS(NIN(IA)).ANEX);END;
/* ВЫВОД РЕЗУЛЬТАТА */
          AS(NOUTN).ANEW=AMIN;AS(NOUTP).ANEW=AMAX;
END;/* MIMAA  */
ELSE DO;/* LO  */
          DO J=1 TO NTYP ;   IF INDEX(TYP,MTYP(J))>0 THEN DO;
                SELECT(J);
          WHEN (1,2,3,4) DO NEWI=C1;FB='0001'B;END;
          WHEN (5,6,7,8) DO NEWI=C0;FB='0111'B;END;
          WHEN (9)         DO NEWI=C0;FB='0110'B;END;
          WHEN (10,11)  DO NEWI=C0;FB='1001'B;END;
                    WHEN (12,13)  DO; CALL
             MAJ(TYP,NEL,NOUTP,NOUTN,NIN);
               GOTO FEND;END;
               OTHER  DO;
               PUT SKIP(1) EDIT('ТИП ',TYP,' ОТСУТСТВУЕТ ЦРОЦ LO
               NEL=',NEL)  (A,A,A,F(3)); ERC=MAX(ERC,12);
               GOTO FEND; END;     END;/*SELECT*/;END;END;
FOUT: /* FPAR */  IF NT=0&PPT THEN CALL FPAR(TYP,NEL);
```

```
            DO I=1 TO MNIN;
            IF NEX(NIN(I))=CZ THEN NEX(NIN(I))=CF;
            NEWI=BOOL(NEWI,OUT(NIN(I)).NEX,FB);END;
            IF NEWI=CZ THEN NEWI=CF;NEW(NOUTP)=NEWI;
          IF NEWI=CF THEN NEW(NOUTN)=CF; ELSE
          NEW(NOUTN)=^NEWI;
FEND:FREE NIN;
END LONIN;
```

An example of a generalized logic element procedure in C ++:

```
#include <vextcpp.h>
void LO(char *typ,int nel,int noutp,int noutn,int nmin,int nsign)
{ IntVector jin (nsign);
  for (int i=0; i<nsign; i++)  jin[i] = i + nmin;
LO(typ,nel,noutp,noutn,jin); }
void LO(char *typ,int nel,int noutp,int noutn,IntVector& jin)
{ int nsign,nmin,i,j;
  unsigned int newi;
  int mnin=jin.Dim();
  IntVector nin(mnin);
static char * typs [] = {"&", "LA", "LI", "1", "LE", "LL", "M2", "IP"};
const int NTYP = sizeof(typs)/sizeof(*typs);
static struct logic
        { unsigned ex1 : 2;
          unsigned ex2 : 2;
          unsigned ex3 : 2;
          unsigned ex4 : 2;
        } ft[NTYP]  =
{ {C0,C0,C0,C1}, {C0,C0,C0,C1}, {C0,C0,C0,C1}, {C0,C1,C1,C1},
   {C0,C1,C1,C1}, {C0,C1,C1,C1}, {C0,C1,C1,C0}, {C1,C0,C0,C1} };
          unsigned int newo[NTYP] =
   { C1,C1,C1,C0,C0,C0,C0,C0 };
        nin = jin;
FTYP: // Begin MINMAX
 if ( strstr(typ,"MIMAA")!=NULL )
 {  int ia; float amin,amax;
    amin = as[nin[0]].ANex; amax = as[nin[0]].ANex;
    for (ia=0; ia<mnin; ia++)
    {  if (as[nin[ia]].ANex > amax) amax = as[nin[ia]].ANex;
       if (as[nin[ia]].ANex < amin) amin = as[nin[ia]].ANex;  }
    as[noutp].ANew = as[noutp].ANex = amax;
    as[noutn].ANew = as[noutn].ANex = amin;
 } // end MINMAX
 else {     for ( j=0; j<NTYP ; j++)
            if (strstr( typ,typs[j] ) != NULL )
```

```
                    newi = newo[j];
                    break;   }
        if (j >> = NTYP)
         {fprintf (stderr, "There is no type% s in the LO procedure,
                Item Number:        % d \ n ", typ, nel);
        return;   }
FPAR:  if (nt == 0 && ppt) { Fpar (typ,nel); }
FOUT: for ( i=0 ; i<mnin ; i++ )
            { if ( out[nin[i]].Nex == CZ )
              out[nin[i]].Nex = CF;
              newi = (
               (ft[j].ex1 & ~newi & ~out[nin[i]].Nex)  |
               (ft[j].ex2 & ~newi &  out[nin[i]].Nex)  |
               (ft[j].ex3 &  newi & ~out[nin[i]].Nex)  |
               (ft[j].ex4 &  newi &  out[nin[i]].Nex)   );  }
              newi = newi==CZ ? CF:newi;
             out[noutp].New = newi;
             out[noutn].New = (newi == CF) ? CF : ~newi;
              }  } // end of LO
```

An example of a generalized logic element procedure in Java:

```
public static void LO(String typ,int nel,int noutp,int noutn,int nmin,int nsign)
   {int jin[] = new int[nsign];
    for (int i = 0; i < nsign; i ++)   jin[i] = i + nmin;
    LO(typ, nel, noutp, noutn, jin);   }
public static void LO(String typ, int nel, int noutp, int noutn, int jin[])
    { int nsign, nmin, i, j;    byte newi = 0;
     int mnin = jin.length;   int nin[] = new int[mnin];
 String [] typs = {"&","LA","LI","|","LE","LL","M2","IP"};//char *typs[] =
{.. .};
 logic ft = new logic(8);
     ft.Set(0,vext.C0,vext.C0,vext.C0,vext.C1);
ft.Set(1,vext.C0,vext.C0,vext.C0,vext.C1);
     ft.Set(2,vext.C0,vext.C0,vext.C0,vext.C1);
ft.Set(3,vext.C0,vext.C1,vext.C1,vext.C1);
     ft.Set(4,vext.C0,vext.C1,vext.C1,vext.C1);
ft.Set(5,vext.C0,vext.C1,vext.C1,vext.C1);
     ft.Set(6,vext.C0,vext.C1,vext.C1,vext.C0);
ft.Set(7,vext.C1,vext.C0,vext.C0,vext.C1);
byte newo[] =
{vext.C1,vext.C1,vext.C1,vext.C0,vext.C0,vext.C0,vext.C0,vext.C0}; nin = jin;
     //FTYP:
if (typ.indexOf("MIMAA") == 0)
{  int ia; float amin,amax;
```

```
      amin = vext.as.ANex[nin[0]]; amax = vext.as.ANex[nin[0]];
       for (ia=0; ia<mnin; ia++)
        { amax = Math.max(amax, vext.as.ANex[nin[ia]]);
          amin = Math.min(amin, vext.as.ANex[nin[ia]]);          }
     vext.as.ANew[noutn] = vext.as.ANex[noutn] = amin;
     vext.as.ANew[noutp] = vext.as.ANex[noutp] = amax;   }
else    {    for (j = 0; j < 8; j ++)
         if (typ.lastIndexOf(typs[j]) != -1 )
           { newi = newo[j];     break;    }
  if ( j >= 8 )
{System.err.println ("Error !! Error !! In the LO procedure there is no type" +
typ + ", element N°:" + nel); return;    }
  //FPAR:
  if (vext.nt == 0 && vext.ppt)  control.Fpar(typ, nel);
  //FOUT:
  for (i = 0; i < mnin; i ++)
      {if (vext.out.Nex[nin[i]] == vext.CZ)
       vext.out.Nex[nin[i]] = vext.CF;
      newi = (byte)((ft.ex1[j] & (newi^3) & (vext.out.Nex[nin[i]]^3))|
  (ft.ex2[j] & (newi^3) &  vext.out.Nex[nin[i]])   | // C0^3 == C1;   T1^3 == C0
                 (ft.ex3[j] &  newi & (vext.out.Nex[nin[i]]^3))   |
                 (ft.ex4[j] &  newi &  vext.out.Nex[nin[i]])   ); }
  newi = (newi == vext.CZ) ? vext.CF : newi;
  vext.out.New[noutp] = newi;
  if (newi == vext.CF) vext.out.New[noutn] = newi;
  else vext.out.New[noutn] = (byte)(newi^3);
 } } // end of LO
```

3.4. Models of components for providing CAD interfaces COD with PCAD system

Possible interface structures of the upper-level COD CAD system with industrial CAD systems are shown in fig. 4.1. Using the figures 4.1 levels allows you to automate the conversion of descriptions of many design decisions in high-level languages [1, 6, 13, 14] into many schemes for industrial CAD systems [2, 8, 28, 29, 32, 33, 103, 120, 121, 160]. The results of studies of multiple implementations of COD CAD interfaces with various industrial CAD systems, classification of functions and data made it possible to provide common component models for various applications. As applications, both CAD PCAD, ALTIUM DESIGNER, ORCAD, CADdy, and virtual reality environment (VRML2) [34] or a standard description of computer systems in ISO10303, EDIF and PDIF formats [10, 29, 33, 81, 75, 79, 116, 133].

An interface is provided for a basic set of component models, including digital integrated circuits, analog-to-digital and digital-to-analog components. The need for the synthesis of component models arises for new components

As an example of component models for the tabular interface of UI CAD PCAD, the DAC model with digital input signals and analog output signal is given below. The DAC model has two entry points with an array of input circuits and with a minimum high-order circuit number indicating the number of inputs. The main model is presented with an array of inputs, and the second is converted to its format. Analog and digital circuits can have the same numbers, because they differ in the prefix NA and ND.

An example of a composite functional component, which corresponds to many real components, is the K561IE11 reversible counter. In this case, the numbers of the input and output circuits are created so that the circuit numbers do not coincide. For example, for the counter they are increased by 100. Here is the text of the DAC interface models with different styles of descriptions in the PL/1 language:

```
   DACMIN:PROC (TYP,NEL,NOUTA,NIN1,NINNEL);
       DCL TYP CHAR(*);
       DCL (NEL,NOUTA,NIN1,NINNEL) FIXED DEC(3);
       DCL I FIXED BIN;
       DCL JIN(NINNEL) FIXED DEC(3);
       DO I=0 TO NINNEL-1;
         JIN(I+1)=NIN1+I;
       END;
       CALL DACNIN(TYP,NEL,NOUTA,JIN);
    END DACMIN;
DACNIN:procedure(TYP,NEL,NOutA,Jin);
      declare TYP char(*);
      declare (NEL,NOutA) fixed dec(3);
      declare Jin(*) fixed dec(3);
      declare i fixed dec(3);
      declare CTYP varying char(20);
      declare NElm fixed dec;
      NElm=ElmCopy(TYP,'DAC');
      if NElm>0 then begin;
        call NetAdd(NElm,'NOUTA','NA',NOutA);
        do i=0 to Hbound(Jin)-1;
          CTYP='NIN'||trim(char(i),' ');
          call NetAdd(NElm,CTYP,'ND',Jin(i+1));
          end;
      end;
  END DACNIN;
```

When the component model is called, it is searched in the correspondence table (uipcad.dbm, uipcad.dbn or uipcad.dbt). If the component is successfully searched for, the record is copied to the schema variant table. After adding a component record, you need to connect the circuit to the required pins. This is

done using the Add Circuit (NetAdd) function, to which the parameters are passed: element number, output name in CAD, type and circuit number. The analog circuit corresponds to type NA, and digital – Nd. Component models are located in the COMP file and are used for command, text, and table interfaces. Automatic conversion of descriptions can be performed using static and dynamic interface libraries. To create a dynamic library of a unified interface in a Windows environment, a def type description file must be prepared using the GENDEFW.BAT command and the ipwgi.bat command file:

```
REM Create a DLL with an import LIB file.
if exist ipgi.obj del ipgi.obj
pli lpgi.pli (ms dllinit default (linkage (system)) langlvl (saa2))
if exist ipwgi.lib del ipwgi.lib
ilib / GI: ipwgi lpgi.obj ipwgi.def
if exist ipwgi.dll del ipwgi.dll
ilink /OUT:ipwgi.dll / DLL lpgi.obj ipwgi.exp> ipwgi.err
```

Below is the text of the DAC and counter models for an interface in C ++:

```
void MCTR(char* _FAR typ,int nel,int Lct,IntVector& _FAR  JOut,int
NOutP,IntVector& _FAR  JIn,int NinP,IntVector& _FAR  Kin)
{    int Count,Fin,FOut,i, inp, outp;
  TSch *tmp;
  char CName[20];
  if((Lct%4)==0)
  { Count=Lct/4;
 // if the length of the counter is more than 4, then the case numbers +100
    if(Count>1)
   { if(NCR==0) NCR=ns+1;  nel=nel+100;  inp=NinP; outp=NOutP; }
  }
  else  {
     if(strcmp(CodSrch("LANG"),"eng")==0)
 printf ("The number of categories of the counter is incorrect !!! \ n");
else printf ("The number of bits of the counter is incorrect !!! \ n"); return
  }
 for(i=0;i<Count;i++)
  {
    tmp=ElmCopy(typ,nel);
    if(tmp!=NULL)
    { Fin=i*4;  FOut=i*4;
     NetAdd(tmp,"NOUT0","ND",JOut[FOut]);
     NetAdd(tmp,"NOUT1","ND",JOut[FOut+1]);
     NetAdd(tmp,"NOUT2","ND",JOut[FOut+2]);
     NetAdd(tmp,"NOUT3","ND",JOut[FOut+3]);
     NetAdd(tmp,"NIN0","ND",JIn[Fin]);
```

```
  NetAdd(tmp,"NIN1","ND",JIn[Fin+1]);
  NetAdd(tmp,"NIN2","ND",JIn[Fin+2]);
  NetAdd(tmp,"NIN3","ND",JIn[Fin+3]);
  NetAdd(tmp,"NOUTN0","ND",JOut[FOut]);
  NetAdd(tmp,"NOUTN1","ND",JOut[FOut+1]);
  NetAdd(tmp,"NOUTN2","ND",JOut[FOut+2]);
  NetAdd(tmp,"NOUTN3","ND",JOut[FOut+3]);
  NetAdd(tmp,"NINN0","ND",JIn[Fin]);
  NetAdd(tmp,"NINN1","ND",JIn[Fin+1]);
  NetAdd(tmp,"NINN2","ND",JIn[Fin+2]);
  NetAdd(tmp,"NINN3","ND",JIn[Fin+3]);
  if(Count>1)
  { if(i==Count-1)  outp=NOutP;
     else   { outp=NCR+i; NCR++; }
     NetAdd(tmp,"NOUTP","ND",outp);   //NDC
     NetAdd(tmp,"NOUTNP","ND",outp);  //NDC
  }
  else  {
     NetAdd(tmp,"NOUTP","ND",NOutP);  //NDC
     NetAdd(tmp,"NOUTNP","ND",NOutP); //NDC
          }
  NetAdd(tmp,"NINP","ND",NinP);
  NetAdd(tmp,"NINNP","ND",NinP);
  if(strstr(typ,"IE11")!=NULL || strstr(typ,"ИЕ11")!=NULL)
  {
    NetAdd(tmp,"NINR","ND",Kin[0]);
    NetAdd(tmp,"NINB","ND",1);
    NetAdd(tmp,"NINNR","ND",Kin[0]);
    NetAdd(tmp,"NINNB","ND",1);
  }
  else   {
    NetAdd(tmp,"NINR","ND",0);
    NetAdd(tmp,"NINB","ND",Kin[0]);
    NetAdd(tmp,"NINNR","ND",0);
    NetAdd(tmp,"NINNB","ND",Kin[0]);
          }
   NetAdd(tmp,"NINC","ND",Kin[1]);
   NetAdd(tmp,"NINV","ND",Kin[2]);
   NetAdd(tmp,"NINO","ND",Kin[3]);
   NetAdd(tmp,"NINNC","ND",Kin[1]);
   NetAdd(tmp,"NINNV","ND",Kin[2]);
   NetAdd(tmp,"NINNO","ND",Kin[3]);
 }
 if(Count>1)   { nel=nel+1; NinP=outp; }
}
```

```
}
void DAC(char* _FAR typ,int nel,int NOutA,IntVector& _FAR  JIn)
{ int i;   char CName[20];
  TSch *tmp;
  tmp=ElmCopy(typ,nel);
  if(tmp!=NULL)
   {
    NetAdd(tmp,"NOUTA","NA",NOutA);
    for(i=0;i<JIn.Dim();i++)
    {
      sprintf(CName,"NIN%u",i);
      NetAdd(tmp,CName,"ND",JIn[i]);
      sprintf(CName,"NINN%u",i);
      NetAdd(tmp,CName,"ND",JIn[i]);
    }
  }
}
void DAC(char* _FAR typ,int nel,int NOutA,int Nin1,int NinNum)
{ int i;   IntVector JIn(NinNum);
  for(i=0;i<NinNum;i++) JIn[i]=Nin1+i;
  DAC(typ,nel,NOutA,JIn);
}
```

The generalized CAD interface of COD with a specific CAD system implies the use of uniform models of top-level components in the comp file for various CAD systems and types of interfaces. However, it is possible to use other models of mid-level components for the command interface. They differ in that each of them outputs to the batch file commands for including a component symbol and naming circuits. Analog circuits are prefixed with NA, and digital–Nd. Inside the models, digital circuits have positive numbers, analog–negative numbers. Examples of models of DAC and counter components in C ++ for the command interface are given in [27, 31].

Models of components of the table and command interfaces in C ++ are translated for all operating systems in the LCP4T and LCP4I subdirectories, loaded into the lcdp4t (uipcad4t) and lcdp4i (uipcad4i) libraries for the PCAD4.5 version and the lcdp8t (uipcad8t) and lcdp8i (uipcad8i) libraries for the PCAD8.5 version. For other programming languages and operating systems, the names of libraries and batch files are compiled in accordance with table 3.9. It is recommended that component model libraries be created using batch files, examples of which are given below.

On Windows and OS/2 operating systems, both static libraries of type lib and dynamic libraries of type dll and additional libraries of type lib are created. Since the types of static and additional libraries are the same, in accordance with table 4.4 names of static libraries begin with the character l, and dynamic and additional–with the symbol i. To create dynamic and additional libraries, you

must have a file of type def, which is created using the batch files gendefo.cmd for OS/2 and gendefw.bat for Windows.

Gendefo.cmd file for a federated interface:

```
REM Create a * .DEF file to the integrated LCGI
REM 1. Compilation of source files
icc  /Id:\codos\include; /Ge- /C  mcfz.cpp mcimport.cpp tsch.cpp comp.cpp table.cpp codread.cpp mcoi.cpp mcv2.cpp mcp8t.cpp mcp4t.cpp mcp8i.cpp control.cpp pcad4.cpp sch4.cpp slb4.cpp sym4.cpp pcad8.cpp sch8.cpp slb8.cpp sym8.cpp mcr.cpp mcai.cpp > icogi.err
REM 2. Creating a * .DEF file
cppfilt /b /q /p mcfz.obj mcimport.obj tsch.obj comp.obj table.obj codread.obj mcoi.obj mcv2.obj mcp8t.obj mcp4t.obj mcp8i.obj control.obj pcad4.obj sch4.obj slb4.obj sym4.obj pcad8.obj sch8.obj slb8.obj sym8.obj mcr.obj mcai.obj > icogi.de
REM This snippet adds LIBRARY EXPORTS automatically.
if exist *.obj del *.obj
if not exist icogi.de goto end
echo LIBRARY EXPORTS >temp
copy temp+icogi.de  icogi.def
del temp
del icogi.de
:end
```

The gendefw.bat file for the unified interface:

```
REM creating a * .DEF file to the integrated LCGI
REM 1. Compilation of source files
icc  /Id:\codos\include; /Ge- /C mcfz.cpp mcimport.cpp tsch.cpp comp.cpp table.cpp codread.cpp mcoi.cpp mcv2.cpp mcp8t.cpp mcp4t.cpp mcp8i.cpp control.cpp pcad4.cpp sch4.cpp slb4.cpp sym4.cpp pcad8.cpp sch8.cpp slb8.cpp sym8.cpp mcr.cpp mcai.cpp > icwgi.er
REM 2. Creating a * .DEF file
cppfilt /b /q /p mcfz.obj mcimport.obj tsch.obj comp.obj table.obj codread.obj mcoi.obj mcv2.obj mcp8t.obj mcp4t.obj mcp8i.obj control.obj pcad4.obj sch4.obj slb4.obj sym4.obj pcad8.obj sch8.obj slb8.obj sym8.obj mcr.obj mcai.obj > icwgi.de
REM This snippet adds LIBRARY EXPORTS automatically.
if exist *.obj del *.obj
if not exist icwgi.de goto end
echo LIBRARY EXPORTS >temp
copy temp+icwgi.de  icwgi.def
del temp
del icwgi.de
:end
```

As a result of the analysis of the implemented CAD UI interfaces with industrial CAD systems, the possibility of building generalized interfaces with a CAD group was shown. A generic interface allows you to use many component models to interface with industrial CAD systems.

Examples of models of DAC components (DAC) and a reverse counter (MCTR) for a generic interface in the JAVA language are given below:

```
//= DAC
public static void DAC (String typ,int nel,int nout,int inBeg, int nSize)
{int[] jin = new int[nSize];
for (int i=0; i < nSize; i++)      jin[i] = inBeg + i;
DAC(typ,nel,nout,jin);}
public static void DAC (String typ,int nel,int nout,int[] jin)
{       {     nel=NEL++;
              vexttsch.tsch.ElmCopy(typ, nel);
              vexttsch.tsch.NetAdd(vext.Curr,"NOUTA","A",nout);
              for (int i=0;i<jin.length;i++)
                    vexttsch.tsch.NetAdd(vext.Curr,"NIN"+i,"D",jin[i]);
       };
};//end DAC
//= MCTR
public static void MCTR (String typ, int nel, int lct, int noutmin, int noutp,
int ninmin, int ninp,int ninr, int ninc, int ninv, int nino, int ninb)
{int[] jin = new int[lct]; int[] jout=new int[lct]; int[] kin=new int[4];
for (int i=0;i<lct;i++) jin[i]=ninmin+i;
for (int i=0;i<lct;i++) jout[i]=noutmin+i;

if (typ.regionMatches(true,0,"IE11",0,4)||
   typ.regionMatches(true,0,"++11",0,4)||
   typ.regionMatches(true,0,"ИЕ11",0,4)   )
{       kin[0]=ninr;  }
else {       kin[0]=ninb; };
kin[1]=ninc; kin[2]=ninv; kin[3]=nino;
MCTR(typ,nel,lct,jout,noutp,jin,ninp,kin);
};
public static void MCTR (String typ,int nel,int lct,int[] jout,
                    int noutp,int[] jin,int ninp,int[] kin)
{ int outp;
           { int stnum=lct/4;
          if (Math.IEEEremainder(lct,4)!=0.0)stnum++;
          for (int k=0;k<stnum;k++)
          { nel=NEL++;
vexttsch.tsch.ElmCopy(typ, nel);
              int NUM=1;
              if (typ.regionMatches(true,0,"IE11",0,4)||
```

```
            typ.regionMatches(true,0,"++11",0,4)||
            typ.regionMatches(true,0,"ИЕ11",0,4)    )
        {       vexttsch.tsch.NetAdd(vext.Curr,"NINR","D",kin[0]);
                vexttsch.tsch.NetAdd(vext.Curr,"NINB","D",1);
        }
        else {    vexttsch.tsch.NetAdd(vext.Curr,"NINB","D",kin[0]);
                  vexttsch.tsch.NetAdd(vext.Curr,"NINR","D",0);
        }
        vexttsch.tsch.NetAdd(vext.Curr,"NINR","D",kin[0]);
        vexttsch.tsch.NetAdd(vext.Curr,"NINC","D",kin[1]);
        vexttsch.tsch.NetAdd(vext.Curr,"NINV","D",kin[2]);
        vexttsch.tsch.NetAdd(vext.Curr,"NINO","D",kin[3]);
        if(k==(stnum-1)) outp=noutp; else outp=vext.ns+k+1;
        vexttsch.tsch.NetAdd(vext.Curr,"NINP","D",ninp);
        vexttsch.tsch.NetAdd(vext.Curr,"NOUTP","D",outp);
        for (int i=0;k*4+i<jout.length && i<4;i++)
            vexttsch.tsch.NetAdd(vext.Curr,"NOUT"+i,"D",jout[k*4+i]);
        for (int i=0;k*4+i<jin.length && i<4;i++)
            vexttsch.tsch.NetAdd(vext.Curr,"NIN"+i,"D",jin[k*4+i]);
        ninp=outp;
      };
    };
};//end MCTR
```

When the component model is called up, an entry is added to the list of elements of the table of the scheme variant. After successfully adding the component record, you need to connect the circuits to the required pins. This happens with the help of the Add Circuit (NetAdd) function, to which the parameters are transferred: circuit number, component number, output name in CAD, type of output in specific CAD. Component models are in the COMP file and are used for all interfaces.

The effectiveness of the generalized interface is explained by the decrease in the volume of static and dynamic libraries, archive files of packages in the JAVA language, as well as the use of common component models, functions for working with tables and files.

3.5. Automation of the formation of libraries of models of components for CAD PCAD

Symbol libraries are used in the synthesis of functional and circuit diagrams, and libraries of constructs – in the design of computer modules.

Creating component libraries is labor intensive and responsible, therefore, the automation of creating libraries is an urgent task [29, 33, 45, 46, 120–122].

In [134], a text description of component symbols was converted into a graphic format without taking into account the functions performed by terminal

types and their equivalence. The author's work [33] describes a method for creating component symbols using a special text description file of type def and using the uipcad.dbt correspondence table used to interface research CAD with other systems. A file of def components of type def is created in a text editor and converted to pdf format, from which PDIFIN utility creates a symbolic image of type sym. The basic parameters for describing the component are formed using the installation symbols for the SYM.sym environment and PRT.prt, which corresponds to the PCAD CAD version. Basic parameters define the metric system of units, information layers and parameters. After the symbols of all the components are formed, the pclib library linking program, which creates the library, is automatically called.

As a result of the analysis of programs for the automatic generation of symbol and construct libraries [31, 134], multilevel models for the formation of symbols and constructs of components are shown in fig. 3.1–3.3. In fig. 3.1 presents a model for the direct formation of symbols and constructive components. In this case, the functional modules of the UI CAD system with industrial CAD systems are used. Such modules are methods for searching and reading components in the correspondence table (Table file), methods for finding paths and names of a formalized task (CodRead file). Specialized are the functions of forming a macro file, as well as methods for reading, adding and writing symbol tables and constructs. The method is fast, but requires knowledge of the structure of binary character files (SYM) and constructs (PRT).

The model for generating symbols and component constructs in a text format for exchanging CAD data PCAD – PDIF (PCAD Data Interchange Format) in pdf files is shown in fig. 3.2. Files of the pdf type are converted to internal binary formats using the pdifin.exe utility. Pdifin programs must conform to the PCAD CAD database format. For PCAD 4.5, the database format is 1.04, and for PCAD 8.5–database format 2.09. The PDIF text format is used to convert component circuits and libraries both in CAD PCAD and in other CAD electronic equipment: ALTIUM PCAD, ORCAD.

<table>
<tr><td colspan="2">1 Control module for generating symbols and component constructs</td></tr>
<tr><td>2 Search Models for Paths and Names of the Federal Law (CodRead)</td><td>Methods for reading and searching for a component in the correspondence table (Table)</td></tr>
<tr><td>3 Macro File Generation Functions</td><td>Methods for reading, adding, and writing symbol tables (SYM) and constructs (PRT)</td></tr>
</table>

Fig. 3.1. Multilevel model of direct formation symbols and component constructs

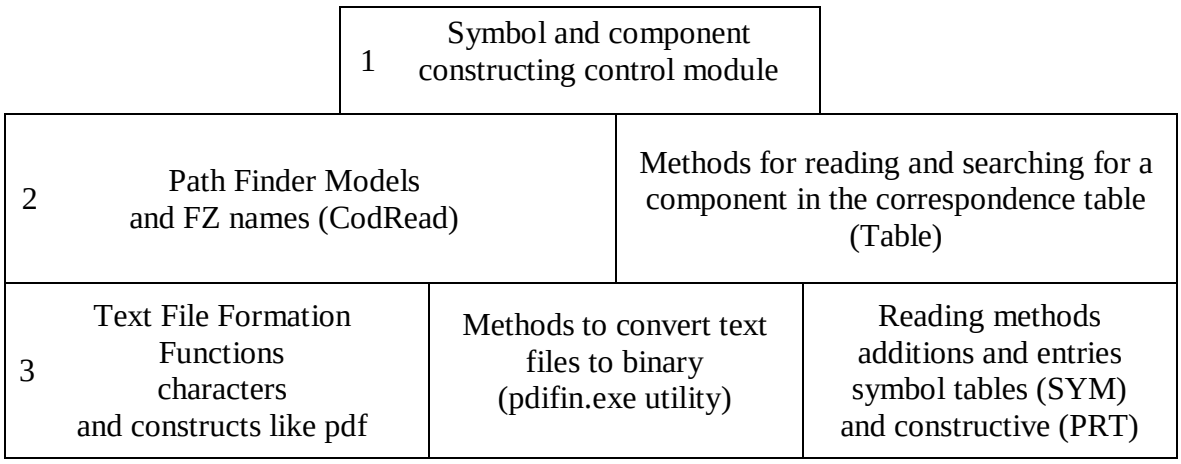

Fig. 3.2. Multilevel character formation model
and component constructs using a pdf type symbol file

The model for the direct formation of symbol libraries and component constructs is shown in fig. 3.3. The method of direct library formation is the fastest and rarely uses access to the hard drive. In the process of creating libraries in the internal CAD format, PCAD uses the functions of the lower levels of the table interface described in Section 3.2. This approach is explained by the generality of schema data structures (file of type SCH) and symbol library (file of type SLB). A circuit is a set of components and connections between them, and a library – many components.

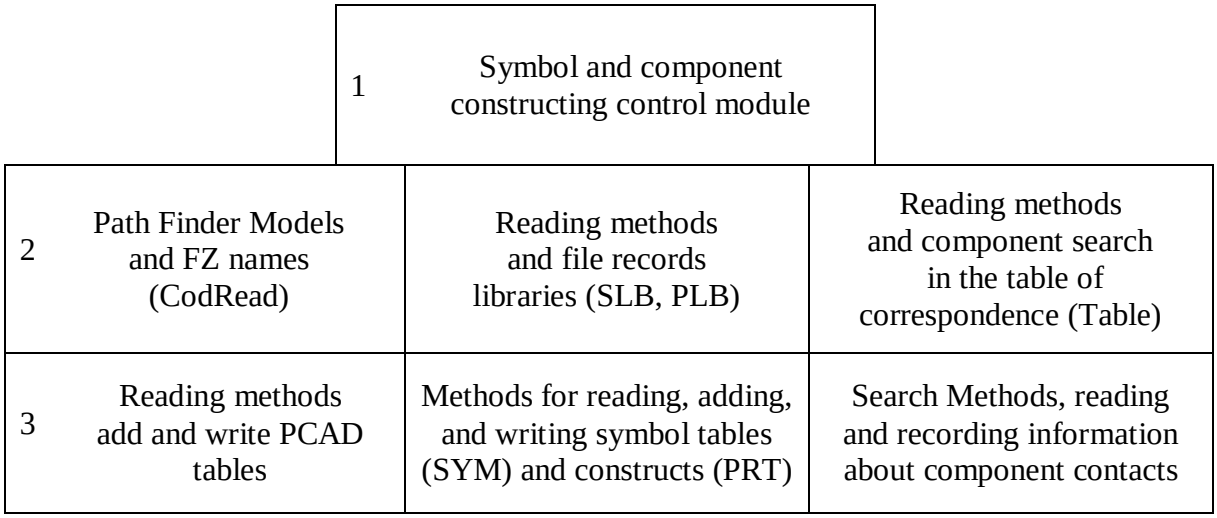

Fig. 3.3. Layered model
direct formation of libraries of symbols and constructs

A number of programs have been developed to form component libraries. Character libraries are formed using the uipcad.dbt table and the symbdbt program. The uipcad.dbt table is a component characterization software. Information support for the formation of symbols and constructs of components is located in the uipcad.dbn table and differs from the uipcad.dbt table in the presence of constructive data in accordance with the pin contacts. The uipcad.dbn table is used by the symbdbn component character program. The name of the library to be created is requested, and the slb type is assigned automatically.

A table of type dbn differs from a table of type dbt only in the last column. Sections separated by a semicolon are introduced into the text of the column description, and the description of the output is supplemented by the equivalence type and contact number.

The record has the following format:

<description>::="<text>"
<text>::=<description of the group of conclusions(DGC)>;[<DGC>; ... < DGC>]
< DGC>::=<description of the output (DO)>,[<DO>, ... <DO>]
<output description>:: =<type of output> <equivalence group>
<contact number> <output name in CAD PCAD>
<output name in CAD COD>.

You should pay attention to the description of components for which the functions of individual sections logical elements – have different or equivalent functions. In this case, the component power leads are not described in each section, but in the last description of the terminal group. The descriptions of the output group of the individual sections are separated by a semicolon, and their number is not limited. After the last section, a semicolon may be missing.

Table 3.9

Component Pin Name Table

Appointment character	Symbol Value	Semantics of meaning
Pin number component	N	First character in the name
Number of pins of one type	K	First character in the name
Array of pins of the same type	M	First character in the name
Attitude directions data transmission	IN	Input
	OUT	Output
	IO	Bidirectional output (input, output)
	ADR	Address input
	0A	Analog Zero Level
	GND	Common signal output
	UP	Positive voltage
	UN	Negative voltage
The ratio of the digits of a number, a string	MIN	The minimum input and circuit number corresponds to the high order
	MAX	The maximum input and circuit number corresponds to the low order
	CP	Senior rank
	MP	Low rank

Continuation of the table 3.9

Signal Function Symbol	R	Initial Installation (Reset)
	S	Setting the state (Set)
	V	Work permit
	C	Synchronization
Output type	P	Direct (Positiv)
	N	Inverse (Negativ)
Signal type	D	Digital
	A	Analog
Type of channel (circuit) and signal	E	Electric
	O	Optic
	P	Pneumatic
	R	Radio channel
	M	Mechanical

In CAD COD at the functional level, power pins are not used, but they are necessary to go to the design level. To distinguish the power leads, the following notation rules are accepted: 0V – digital common conductor; 0VA – analog common conductor; UP, UP1, UP2 – the output of the positive supply voltage in ascending order; UN, UN1, UN2 – output negative supply voltage in ascending order in absolute value. The designation of conclusions is formed in accordance with the data in table 3.9.

For example, the description of the terminals of the K561LA7 chip will be as follows:

"0 1 1 IN1 NIN1, 0 1 2 IN2 NIN2, 1 0 3 Y NOUTP;
0 2 5 IN1 NIN1, 0 2 6 IN2 NIN2, 1 0 4 Y NOUTP;
0 3 8 IN1 NIN1, 0 3 9 IN2 NIN2, 1 0 10 Y NOUTP;
0 4 12 IN1 NIN1, 0 4 13 IN2 NIN2, 1 0 11 Y NOUTP;
0 0 7 0V 0V, 0 0 14 UP1 UP1"

To automate the creation of component construct libraries, the PRTBAT program was developed. Information support for the formation of constructs is in the tables uipcad.dbn and pac.dbt. The uipcad.dbn table is discussed above, the structure of the pac.dbt table is shown below. The table contains seven columns. The first column contains the designation of the housing according to the standard, and its content coincides with the corresponding column of the tables uipcad.dbt and uipcad.dbn. The second column determines the number of contacts, the third and fourth–dimension on the long (GABX) and short (GABY) sides. The fifth and sixth columns define the pitch along the long (STEPX) and short (STEPY) sides in millimeters. The seventh column contains the name of the image (trace – FP) of the construct on the printed circuit board. The program parameter is the name of the library to be created. Messages about detected errors are recorded in the PRTBAT.ERR file. The program works similarly to the previous one. Using the information in the tables, a pdf file is created, which is converted by the PCAD PDIFIN utility into a construct file of the prt type. Upon completion

of the creation of many prt files, a batch file for the PCLIB utility is created, as well as a library, and files of the prt type are erased. The formation of symbols and component constructs using a text description and subsequent conversion to the PCAD CAD database format by the PDIFIN utility is characterized by clarity,

The algorithm for creating symbols and component constructs in the format of a specific CAD system with subsequent integration into a library structure is shown in fig. 3.4, the algorithm for automatically creating libraries of symbols and component constructs in the specific CAD format is presented in fig. 3.5.

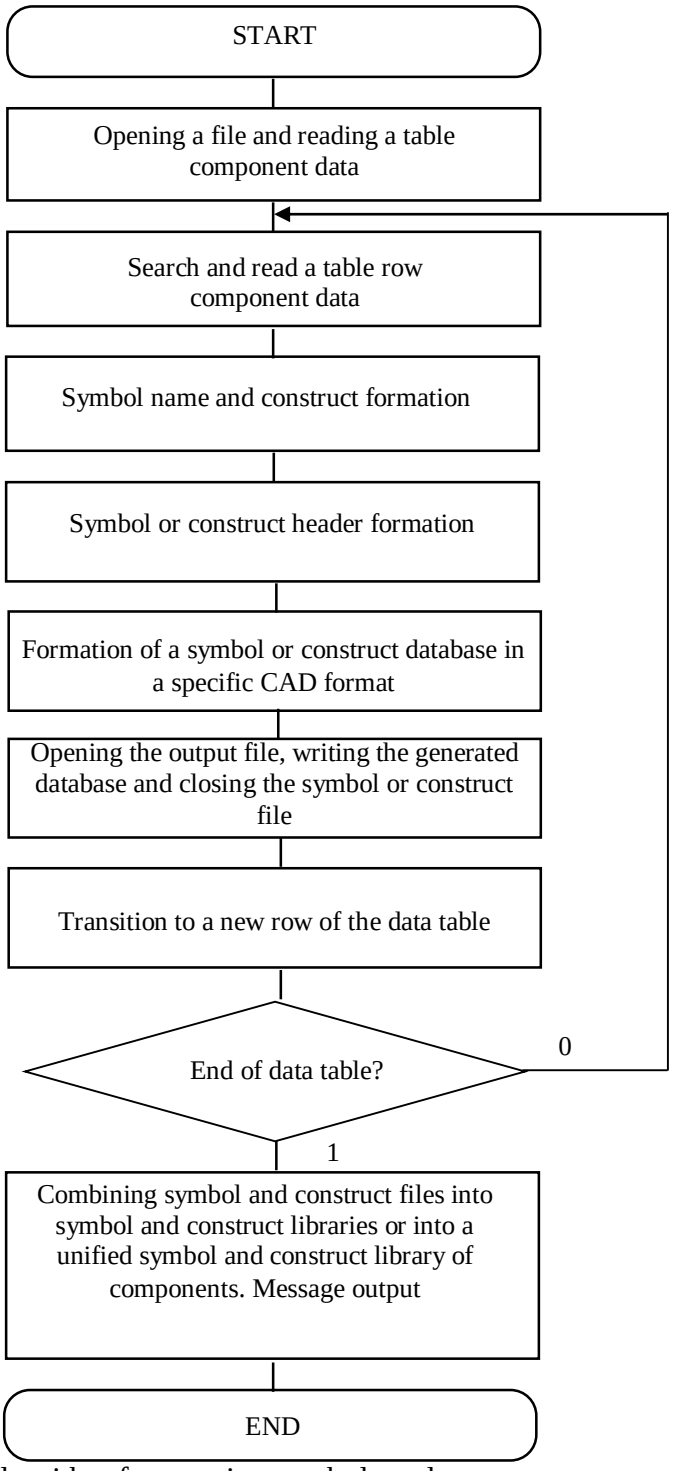

Fig. 3.4. Algorithm for creating symbols and component constructs in the format of a specific CAD with subsequent integration into a library structure

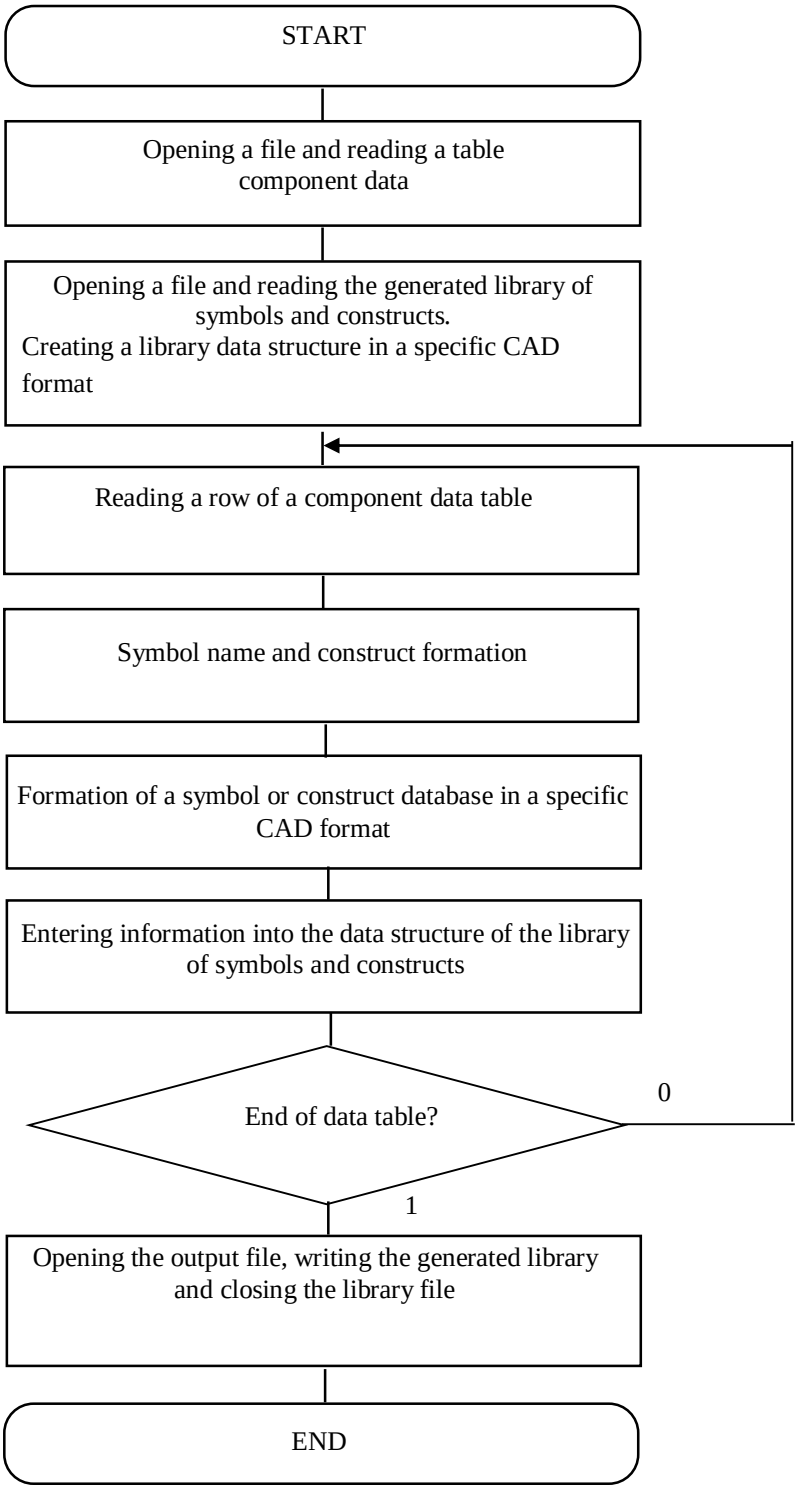

Fig. 3.5. Library Auto-Algorithm
symbols and component constructs in a specific CAD format

To create symbols and component constructs, you need to find the paths, open the uipcad.dbn correspondence table file and load the data. After loading the data, a table row is searched and read, the name of the symbol or construct and the header of the corresponding file are formed. After the header is formed, the tables are filled in the specific CAD format, the output file is opened, the generated tables are recorded and the file is closed. After closing the file, you can go to a new row in the data table, and the process is repeated by searching and

reading the row of the data table. After the component data table has been exhausted, the symbol files and component constructs are combined into libraries.

The program for the direct generation of symbols and constructs in the database format 2.09 CAD PCAD 8.5 consists of modules for reading information about the component from the UIPCAD.DBN table, a module for generating data structures in the format of the PCAD 8.5 database, and a module for writing to the symbol and construct file.

For each component from the table UIPCAD.DBN, symbol files of type SYM and constructs of type PRT are generated, which are subsequently combined into libraries of symbols of type SLB and constructs of type PLB. In the process of the program, the created symbol is displayed on the screen to control the process of forming symbols and constructs.

3.6. Automation of the formation of text descriptions life cycle information support computing systems

Designing begins with the formation of a technical task, which should reflect the purpose of creating a system or device and many limitations. The synthesis of a device (system) in the form of a formalized task is a difficult formalized creative stage performed by an engineer. The type of formalized task depends on the CAD system used, although the information is invariant with respect to the specific CAD system. The formalized task contains information about the types and names of components, the connections between components and the forms of interaction of the components with each other and with the external environment.

The development of processing, information transfer and computer-aided design tools allows you to create information models of objects for the entire product life cycle. The result of the development of information, material and energy technologies has become virtual enterprises – a distributed set of organizations designing, manufacturing and servicing industrial products. The effectiveness of virtual enterprises is determined by the spatial distribution of natural, energy and intellectual resources and the industrial zone. The entire product life cycle, from design and production to servicing and disposal, must be accompanied by information support corresponding to production automation systems and their integration according to the international standard STEP (Standard for Exchange of Product data) ISO 10303 and GOST R ISO 10303–1–99 [104]. The ISO 10303 standard is supplemented by the ISO 15531 MANDATE (Industrial manufacturing management data) data presentation standard for virtual enterprises, the ISO 13584 P-LIB (Parts Library) data presentation on standard components of industrial products, ISO 14959 Parametrics data presentation of product parameters, ISO 15926 (Industrial automation systems and integration) industrial automation.

Information about the product is created during its design during the synthesis procedure, and is used and supplemented during production, operation and disposal throughout the entire life cycle in accordance with GOST R ISO

10303–99 (ISO 10303) [104]. Product information is presented in the description language in accordance with the rules of grammar [146]. Product descriptions should be perceived by both the user and the formal system and are called formalized tasks. A Formalized task (FT) consists of many sections that are syntactically and semantically homogeneous [29, 33, 103]. Therefore, the synthesis and analysis of formalized tasks is performed in sections. The sections of the formalized task (PFT) can describe the initial values, external influences, components and relationships between them, and the management of the analysis process. Product descriptions can correspond to different levels of abstraction and design stages. The formalized CAD tasks PCAD in the form of a text file of the alt type and the CAD PRAM 5.3 of the prm type correspond to the design level of the design, and the formalized CAD tasks of the CAD correspond to the functional level.

The text form is the best for storing and transmitting descriptions, and the graphic form is more convenient for the user to quickly perceive information. Therefore, the problem arises of automatically generating text descriptions and converting them into graphic form for quick human perception. For example, the ISO 10303–99 standard provides the EXPRESS language and the corresponding graphical form EXPRESS-G, which is generated automatically [103].

For complex CAD (CSAR) PRAM 5.3 [32, 103], the formalized task is presented in the form of many sections that are stored in one file. The main sections of the formalized tasks are: “List of elements”, “Connections”, “Passport”. The syntax and semantics of the formalized task sections are described in [29, 32, 87, 101] and in the industry standard OST 4G 0.091.397–85. The design route is described in detail in [30]. The design result, which in PRAM is called the archive, contains, along with the output, sections of the initial formalized task. The full archive for the multi-channel analog-to-digital converter module is located in the R22 file of the PRAM subdirectory. The text of the industry standard is located in the same subdirectory in the PRAM53.HLP file. Description R22 corresponds to the real module,

A formalized task is synthesized by an engineer or automatically generated in CAD COD when selecting in the menu local or remote execution (Execute, Remote Execute) submenu of the text interface – PRAM. A section of a formalized job describing compounds is formed by components or by chains. A prm file contains three sections: “Passport”, “List of elements” and “Connections”, the first being copied from the file X:\fa\pram\pasport.txt.

Formalized CAD tasks PRAM 5.3 and file type alt CAD PCAD correspond to the design level, therefore, it is necessary to convert components of arbitrary capacity to the actual and packaging for the cases.

An alt file is accepted by all versions of PCAD CAD (4.5, 8.5–8.7, 2000–2006), but the latter should not specify the path to the component library that is selected in the PCAD2000 menu. An alt file contains sections (sections): PATH, BOARD, SHEET. The PATH section contains many paths to search for computer module construct files and component construct libraries (PATH=<path 1>; <path n>;). The BOARD section contains the name of the computer module

construct file (BOARD = <full file name>;). The SHEET section describes the list of components and connections in the PARTS and NETS subsections, respectively (SHEET = <sheet name> PARTS <component list> NETS <connection list> ENDSHEET). If you do not specify the sheet name in the SHEET section, then the sheets will be named 01, 02, i.e., in ascending order. The PARTS subsection is an enumeration of circuit components, with the same components being grouped and written in a single line separated by commas (<component type> = <constructive name 1>, <constructive name n>;). The NETS subsection describes the connections in the format <circuit name> = <constructive name 1> / <contact number>, <constructive name n> / <contact number>). In accordance with the design path in PCAD CAD [31], the alt type file is converted by PCNLT to a pkg binary for design engineering, and in PCAD2000 CAD, the structural and description files are loaded in the File / Open menu item of the printed circuit board editor [120, 128]. <constructive name n>;). The NETS subsection describes the connections in the format <circuit name> = <constructive name 1> / <contact number>, <constructive name n> / <contact number>). In accordance with the design path in PCAD CAD [31], the alt type file is converted by PCNLT to a pkg binary for design engineering, and in PCAD2000 CAD, the structural and description files are loaded in the File / Open menu item of the printed circuit board editor [120, 128]. <constructive name n>;). The NETS subsection describes the connections in the format <circuit name> = <constructive name 1> / <contact number>, <constructive name n> / <contact number>). In accordance with the design path in PCAD CAD [31], the alt type file is converted by PCNLT to a pkg binary for design engineering, and in PCAD2000 CAD, the structural and description files are loaded in the File / Open menu item of the printed circuit board editor [120, 128].

The EDIF (Electronic Data Interchange Format) format is an international standard and is used to exchange descriptions of electronic equipment between various CAD systems [104]. EDIF format is supported by major CAD electronic equipment; an exchange file of the EDF type is created and accepted by CAD PCAD and ORCAD. PCAD CAD includes a utility for converting edf files to alt files. In contrast to the description of the design level, in the EDIF format a description of external influences is possible. The development of EDIF and STEP standards in the field of electronics is carried out by the ISO JWG9 working group [128].

The EXPRESS language is intended for the presentation of product data and the exchange of these data between computing systems. The exchange structure in accordance with ISO 10303–21–99 should be a sequential file containing two sections, one of which is the header, and the second contains data. The header section contains information about the entire exchange structure and begins with the HEADER token and ends with the ENDSEC token. The header section must contain object instances: file_description, file_name, file_schema.

Example header section:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('FILE CONTAINS A  UNIT MODEL'),'2:1');
FILE_NAME('COMPUTER UNIT MODEL FP01 #1');
FILE_SCHEMA(('EXAMPLE OF SCHEMA FP01'));
ENDSEC;
```

The structure of the data section is shown below:

```
DATA
<Data section content>
ENDSEC;
END-ISO-10303-21;
```

The adoption of an international standard led to the emergence of software products supporting information support for the product life cycle in accordance with CALS technology [104, 133]. Product life cycle support software products are abbreviated as PLM (Product Lifecycle Management).

ST-DEVELOPER is a development environment for EXPRESS data using universal programming languages, ST-WEB PUBLISHER is a tool for presenting data on the network, ST-EXPRESS converts textual descriptions into graphic ones. The software packages ENOVIA from IBM, IMAN from UNIGRAPHICS can serve as examples of PLM products.

Table 3.10

XML Description Languages

Name	Full name	Appointment
CKML	Conceptual Knowledge Markup Language	Description of knowledge representation methods and data analysis result
UDDI	Universal Description Discovery Integration	Universal Catalog Tool Support
MML	Mathematical markup language	Description of mathematical formulas and data
PGML	Precision Graphics Markup Language	Description of two-dimensional scalable vector graphics
VML	Vector Markup Language	Description of two-dimensional vector graphics
SML	Shema markup language	Description of electronic circuits (SFU)
TML	Tutorial markup language	Description of educational information for its online presentation and assessment of student knowledge
WML	Wireless markup language	Description of general content intended for wireless transmission
RDF	Resource Description Framework	Resource Description Language
OWL	Web ontology language	Ontology Description Language
WSDL	Web Services Description Language	Network Services Description Language

	1	Formalized task (FT)	
2	Search Models for Paths and Names of the Federal Law (CodRead)	Component Models (Comp)	Models of control procedures and signal generation functions (Control)
3	Schema Variant Data Structure (TSCH)		
4	Methods for reading, adding, and writing to the description file data structure	Reading, Writing, and Search Methods in the Schema Variant Data Table (TSCH)	Methods for reading, writing and searching for a component in the table of correspondence (Table)
5	Methods for reading, adding and writing to the component data structure, its outputs and circuits	Methods of packing components in a case in a table (TSCH)	Methods for searching, reading and writing information about component contacts and connected circuits
6	CAD-specific project file or standard data exchange structure		

Fig. 3.6. Layered text interface structure

Markup languages make it easier to work with text descriptions: HTML, SGML, and XML [29, 33, 153]; moreover, SGML and XML allow describing document structures and are the basis of specialized languages given in table 3.10. The absence of an XML-based electronic circuit description language has led to the creation of both the SML language and the automatic generation of a circuit description. Two files are created with the name of the formalized task: the information type xml and the start type htm for loading the schema into a viewer that supports XML, such as Internet Explorer for Windows or multi-platform Mozilla. Display in the viewer is provided by the library of functions in the Parser file. The XML standard and its support have simplified the display of charts and timelines.

Consider the structure and basic functions for the formation of the entire set of representations of text files describing the functional level of computing devices and systems. The multilevel structure of the text interface is shown in fig. 3.6.

3.7. Models of components providing CAD interface COD with CAD CATIA

CAD CATIA is an integrated computer-aided design system (ICAD), covering many subject areas. The effectiveness of the ICAD CATIA is explained by the presence of common design procedures and operations and specialized modules. Functional subsystems correspond to the main subject areas and traditional areas of specialist training: mechanical engineering, design of industrial plants, electronics, electrical engineering, design and placement of

equipment in industrial facilities, building equipment. The work was carried out with CATIA CAD version 5 for personal computers R16 IBM Order No: 8232SP, in which the author is specified by the user.

Designing begins with the synthesis of possible variants of the schemes and their description in the form of formalized tasks of the educational research CAD COD. After analyzing a formalized task with a description of several options for schemes, performance and performance criteria are evaluated and the best option is selected. The schematic diagram can be displayed in standard Internet browsers (Microsoft Internet Explorer (IE), Mozilla). A preliminary image of the module design in CAD COD is displayed in a virtual reality environment (file of type wrl) or X3D (file of type X3D).

Table 3.11

A subset of the CATIA CAD commands

Team assignment	Name teams	Name subcommands	Macro
Create a new file	File	New	Language = "VBSCRIPT" Sub CATMain () Set documents1 = CATIA.Documents Set productDocument1 = documents1.Add ("Product") End sub
Open file		Open	Language = "VBSCRIPT" Sub CATMain () Set documents1 = CATIA.Documents Set productDocument1 = documents1.Open (FileName) End sub
Save file		Save	Language = "VBSCRIPT" Sub CATMain () Set productDocument1 = CATIA.ActiveDocument productDocument1.SaveAs FileName End sub
Copy to clipboard	Edit	Copy	Language = "VBSCRIPT" Sub CATMain () Set productDocument1 = CATIA.ActiveDocument Set selection1 = productDocument1.Selection selection1.Clear Set product1 = productDocument1.Product Set products1 = product1.Products Set product2 = products1.Item (Element) selection1.Add product2 selection1.Copy End sub

Continuation of the table 3.11

Paste from clipboard		Paste	Language = "VBSCRIPT" Sub CATMain () Set productDocument1 = CATIA.ActiveDocument Set selection1 = productDocument1.Selection selection1.Paste End sub

The design of the module is designed in a CAD electronic equipment (CAD EDA). In CAD EDA, component placement and connection tracing are performed, and the module design result is transferred to CATIA CAD. For example, when designing the module design in PCAD CAD, the description is output in PDIF formats (pdf file type) with conversion to IDF format using the IDF2PCAD.exe utility. An idf file is accepted by the CATIA Circuit Board Design CAD subsystem. Upon completion of the module design, issues of placement and pairing with other components of the integrated product are resolved.

The conversion of the formalized job description of the device into a CATIA CAD document is performed by automatically generating a macro file in the VBSCRIPT language with the extension catvbs. The subset of commands used in the conversion is given in table 3.11. The main modules of CATIA CAD interface are presented in table 3.12.

It is recommended that you create a component library in the CATIA CAD object catalog – ObjCat. Components must match the uipcad.dbm or uipcad.dbn tables and the pac.dbt component geometry table in the CODOS\BIN directory. Compound components may consist of many simple components.

Table 3.12

Core Interface Modules

Module name	Module purpose
Comp	Source code for top-level component models
mci, mpi, mji, mcinet, mpinet, mjinet, mcipcb, mpipcb, mjipcb	Source code of modules in programming languages CPP, PL1 and JAVA
Control	Control Unit (Iprint, Aprint, Nprint)
Codread	Functions of reading paths and file names UI CAD COD
Table	Functions for working with component tables uipcad. dbm, uipcad.dbn
Tsch	Functions for working with a variant table

The source code of the general module is contained in the mci.cpp file, the source code of the PCB macro generation module is in the mcipcb.cpp file, the NET macro generation module is in the mcinet.cpp file, function prototypes are in the mci.hpp file. The principle of operation of all three modules is to search for

components in the TSCH options table and write commands to the macro output file in accordance with the components found.

In the PCB module, we use the FindChip() function to search for microcircuits, in which a list of components is formed, their parameters are determined, and these parameters are entered into the Chip structure. Writing commands to the macro file is performed by the draw_Board() function.

The NET module has FindSNet() and FindMNet() functions. Purpose – to search the table of options for stationary and mobile network components. If network components are found in the formalized task, the draw_room() function is launched, which writes commands to create the room, computers and access points to the macro file.

The general module contains all of the above functions. The Kchip variable contains the number of chips found, the Knet variable contains the number of network components equal to the sum of the fixed (KSnet) and mobile (KMnet). The network component can be stationary or mobile servers or personal computers, telecommunication components (switches, routers, access points). If the number of network components is not equal to zero, then premises will be created for network objects. If there are network components and microchips, then the network and module are displayed. A local wireless network outside the building can be defined.

The development of computing technology has led to the emergence of programmable network computing devices. The dictionary of the main variables and functions is given in table 3.13.

As a result of a multivariate formalized task, a lot of macro files are created, the number of which corresponds to the number of options. The full path to the ObjCat component library is shown in table 3.14, determined by the value of the environment variable CODLOC.

Table 3.13

Dictionary of Key Variables and Functions

Function or variable name	Function or Variable Assignment
int mci()	Entry Point, Main Function
void FindSNet()	Fixed network component search function
void FindMNet()	Search function for mobile network components
void FindChip()	Chip search function
void draw_Board()	PCB creation and generation macro function
void draw_Room()	Computer class creation and recording macro function
int KSnet	Number of fixed network components found
int KMnet	Number of mobile network components found
int kchip	Number of circuit components found
int kunet	Number of network devices found
int kurnet	Number of real network devices found
int KUVnet	Number of virtual network devices found

The developed modules are included in the generalized interface library by recompiling the static and dynamic libraries. To compile the static library, we use the command file lcwgi.bat for Windows and lcogi.cmd for OS / 2. To compile a dynamic library, you first need to get a DEF type definition file. DEF type files are created for OS/2 with the gendefo.cmd command and for Windows with the gendefw.bat command. After creating the definition files, you can proceed to creating dynamic libraries such as DLLs with the icogi.cmd and icwgi.bat batch files. As a result, when the Execute->Interface CATIA->Command NET and PCB menu item is selected, the LPEX fti.lx editor command is executed and a micro-command file is created.

Table 3.14

Table of components in the ObjCat catalog

File name	Appointment
A1-A4.CATDrawing	Drawing format A1, A2, A3, A4
Dip14- Dip 48.CATPart	Chassis ICs with the number of pins 14, 16, 20, 24, 28, 32, 42, 48
R.CATPart	Resistor
C.CATPart	Capacitor
IBM.CATPart	IBM PC module
hs_m.CatProduct	Image of a man
hs_w.CatProduct	Image of a woman
PC.CATPart	Computer IBM PC
MPC.CATPart	Mobile IBM PC
TPC.CATPart	Mobile tablet pc
PPC.CATPart	Handheld computer
ap.CATPart	Wireless hotspot
ap_sphere.CATPart	Active access point
ap_konus.CATPart	Active Directional Access Point

The macro file for CATIA CAD is executed by selecting the Tools menu item and the Macro-> Macros submenu. In the Macros window, select the file and run the Run command. The result is checked and the formalized task, library components or the conversion module are adjusted. The result can be saved in CATIA (CATProduct) formats or in VRML (wrl), STEP (stp) formats.

The image of the product at the initial stage of computer module design can be obtained using the interface with the virtual reality environment. You can display the changes in digital signals on the terminals of the chips and the temperature of the chip. However, the three-dimensional representation in CATIA CAD is characterized by increased volume, quality and detail display.

3.8. Models of components for providing CAD interface of COD with virtual reality environment

In a virtual reality environment [3, 33, 136, 152, 153, 154], modules, blocks, computers and computer systems with stationary and mobile objects, as well as their manufacturing processes, can be represented. The basis of the

representation of virtual reality is the description of many objects in the VRML language and programs for converting descriptions into images, called programs for viewing VRML descriptions [29, 33, 55, 56].

A description of the set of objects (representing the appearance of modules, blocks, racks of computers and computers) in the VRML language can be obtained by automatically converting a formalized modeling task. The formalized task can be described in any of the universal programming languages PL/1, C ++, JAVA. To convert a formalized task in OS/2 or Windows, you need to select a description language in the menu, enter the name of the formalized task, and select VRML in the execute menu. The resulting wrl file should be displayed using the viewer. The wrl file type is short for world.

The basis for the automatic conversion of a formalized task in a description in the VRML language is multifunctional component models with a common interface [6]. The presence of component models with a common interface allows you to convert formalized tasks into results for various applications. Depending on the application, models of components and control modules are synthesized, which are combined into static and dynamic libraries and selected depending on the type of application. In order to reduce the complexity of creating models of components and control modules, models of various levels are used. Top-level models pass parameters to mid-level models, which in turn form sections of the output file using the lower-level model. Top-level models are independent of the type of application.

The header of a wrl file in VRML is generated by the control module and written to the output file. After calling all the components of the top-level procedures, the table of the scheme variant is completed and the output file in the VRML format is formed. Geometric models of components are displayed in the order of their description in a formalized task. Regular structures of computing systems can be described in a loop, and irregular structures can be represented in the description of specific components. The control module after completion of the output file closes all open files. Component images are located in the subdirectory X:\FA\VRML\ObjVRML.

Formalized tasks to be converted into a VRML format should not contain functional models of components, since the models must form a VRML file. Therefore, functional models must be enclosed in comments. Component models contain formalized tasks starting with a combination of FP03 and computer model (FP08 – FP12). The descriptions of computer systems contain models of operational processing (MKM) and final data processing (MPRINT).

To create and debug new models of components for transforming a formalized task into a VRML file, you need to include the top-level component interface model in the Federal Law.

3.9. Synthesis of a model and algorithms for the operation of an analog-to-digital subsystem with a USB interface

Analog-digital subsystems with a USB interface on the LSI allow you to enter analog and digital signals, process them and output analog and digital signals. With increasing integration, analog-to-digital devices came closer to signal sources. Thus, they reduced interference when transmitting analog signals over communication lines, the length of analog lines and increased the length of lines for transmitting digital messages. The number of analog signals per device decreased, while the number of devices increased.

Promising are analog-to-digital devices with a USB interface (ADCUSB) and with a radio channel (ADCRCA). ADCUSB devices are not far from the system unit, but there is no need for an additional power source. Devices with a radio channel allow a significant distance from the access point (AP), but this requires an additional power source and a relatively large energy cost for data transfer.

Analog-digital devices with a USB interface are produced by various foreign enterprises. The main characteristics of such devices are given in table 3.15.

As an example, in fig. 3.7 is a functional diagram of the ADCUSB device type NIDAQ 6009 company Natnional Instruments. The device contains a power source for additional components, the USB interface itself, a USB bus microcontroller, an 8-channel ADC, 2 DACs and 12 digital input-output lines in two groups (8 and 4).

Table 3.15

Main features of analog-to-digital devices with USB interface

Title	Price	Number of inputs ADC	Bit ADC	Frequency ADC kHz	DAC Outputs	DAC resolution	Frequency, Hz	Number of Digital Inputs / Outputs	Cost per unit of productivity, cu
NI USB 6009	285	8/4	13/14	48	2	12	150	8 + 4	0,00044
NI DAQ Pad 6016	1345	16/8	16	200	2	16	300	32	0,00042
LabJack U12	119	8/4	12	8	2	10	50	20	0,001
Eagle Daq	700	16/8	14	250	4	14		8 + 8 +8	0,00022
DAT dt9836	2400	12/6	16	225	2	16	500	16 + 16	0,00066
MCC USB 3110	399	–	–	–	4	16	100	8	–
SuperLogics 9801	575	16/8	12	100	–	–	–	16	0,0047
Phidgets	–	8/4	–	–	–	–	–	8 + 8	–

A model of an analog-to-digital system contains a digital computing subsystem or a machine, analog-to-digital subsystems connected to the digital part using standard input-output channels. The ADCUSB analog-to-digital subsystem is connected via a USB channel, through which control information and output data are transmitted and conversion results and digital signals are received. The control information sets the configuration of the subsystem. The data structure can be transmitted directly or the chain numbers to which the data structure corresponds can be transmitted. The transmission of circuit numbers facilitates the visualization of results and complies with the principles of the high-level COD UI CAD. The syntax and semantics of the parameters of the ADCUSB function are given in table 3.16. The MODE identifier corresponds to single or differential analog inputs.

The formalized task fp03nin serves to debug the ADCUSB function model; the task fp03ninm serves to debug the ADCUSB function model when working in conjunction with an external multiplexer that extends the capabilities of the device. A circuit is considered operable if this resistance is within specified limits. This approach has been tested in projects for the Ministry of Geology and is effective for the remote use of computing devices and systems. The formalized task fp04nin corresponds to fp03nin, but differs in the absence of the source code for the ADCUSB function and the use of a functional model from component libraries. The job fp04ninm corresponds to fp03ninm, contains an external multiplexer, and due to the lack of source code, the ADCUSB function is more convenient to use. It is possible to use several analog-to-digital ADCUSB subsystems with external switches and other components. The text of the formalized task fp04ninm is given after the block diagram in section 5.

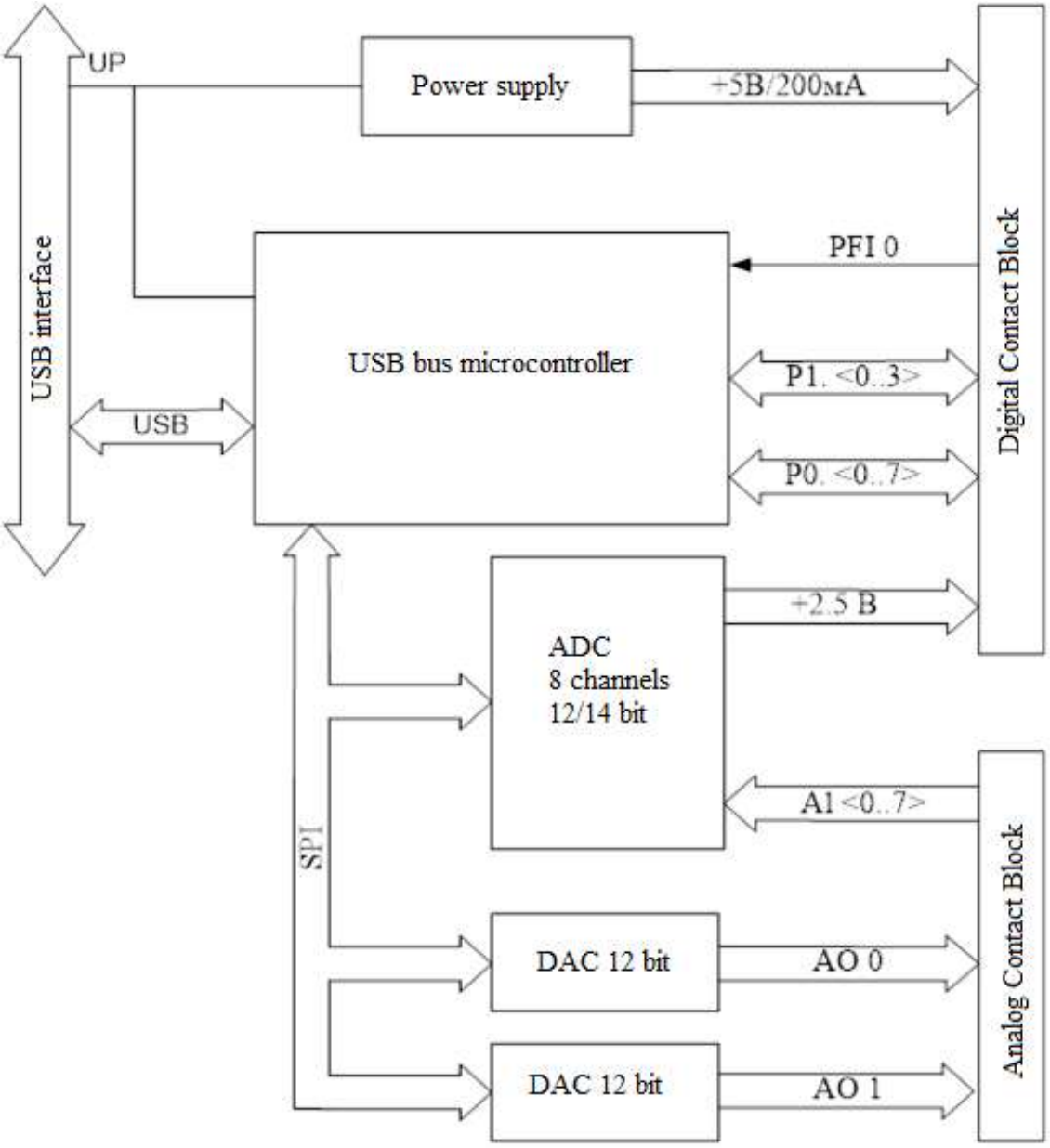

Fig. 3.7. Functional Diagram ADCUSB Type NIDAQ 6009

The ADCUSB subsystem model consists of several sections [29]. The FTYPE section checks the validity of the types of components that are transmitted by the TYP identifier and the NEL component number. The component number is necessary to distinguish the components of the same type. The functions of the FS section are the initialization of variables and the restoration of states. The calculation of the output values is performed in the FOUT section for each component of the subsystem: digital-to-analog converters (DAC1, DAC2), analog-to-digital converter ADC and digital input-output DIO. For each component, a certain sequence of actions (steps) is performed: reading the state of the component, recording the configuration and status, performing the action in accordance with the table of valid commands (table 3.17).

Table 3.17

NI6009 USB Device Model Syntax and Semantics Table

№	Identifier	Data type PL/I	A type data C, C ++, JAVA	A type data ADA	Semantics	Notes
1	TYP	CHAR (*)	char (C ++) string (JAVA)	Character	Item type	NI6009
2	NEL	DEC FIXED (3)	Int	Integer	Item number	
3	NINDAC1		intVector (C ++) int[] (JAVA)		Code number Dac1	
4	NINDAC2				DAC2 Input Number	
5	MNOUTADC				Out ADC. Array of numbers	
6	Mndio1				Digit chain number. in-out. MDIO1	8 digits
7	Mndio2				Digit chain number. in.-out. MDIO2	4 categories
8	MODE				ADC operation mode	10083-one. 10106-diff.

Table 3.18

Valid Commands Table

Team assignment	Team Name	Parameters Used	Return Values
Task creation	DAQmxBaseCreate Task	**TaskName const char[]** – task name	**taskHandle TaskHandle*** – pointer to the created task
Delete task	DAQmxBaseClearT ask	**Taskhandle** TaskHandle – task name	**Status int32** – returns an error code
Creating an analog input channel (description)	DAQmxBaseCreate AIVoltageChan	**TaskHandle TaskHandle –** task name physicalChannel const char **[]** – input channel number, for example "Dev1 / ai0" terminalConfig **int32** – input channel configuration **minVal** float64 – minimum value (V) **maxVal** float64 – maximum value (V) **units** int32 – dimension, leave in DAQmx_Val_Volts (V)	**Status int32** – returns an error code
Analog output	DAQmxBaseWrite AnalogF64	**TaskHandle TaskHandle –** task name **numSampsPerChan** int32 – number of displayed samples **autoStart** bool32 – always false **timeout** float64 – timeout **dataLayout** bool32 – an indication of the method of grouping values **writeArray** float64 [] – an array of 64-bit values for output	**sampsPerChanWritten** int32* – the number of samples written to the buffer **Status int32** – returns an error code
Reading eight bit digital samples	DAQmxBaseReadD igitalU8	**TaskHandle TaskHandle –** task name **numSampsPerChan –** int32 – number of samples **timeout** float64 – timeout **fillMode** bool32– indication of the method of grouping values **arraySizeInSamps** uInt32 – array size, grunting read values	**sampsPerChanRead** int32* – the number of samples written to the buffer **Status int32** – returns an error code
Eight-bit digital readout	DAQmxBaseWrite DigitalU8	**TaskHandle TaskHandle –** task name **numSampsPerChan** int32 – number of displayed samples **autoStart** bool32 – always false **timeout** float64 – timeout **dataLayout bool32 – an indication of the method of grouping values** **writeArray** uInt8 [] – array of output values	**sampsPerChanWritten** int32* – the number of samples written to the buffer **Status int32** – returns an error code

The results obtained are checked and, in the case of reliability, are recorded in a table of digital or analog signals at a given circuit (signal) number.

An example of an ADCUSB model with parameters in the form of circuit numbers in C ++ NIDAQ 6009 is given below.

```
//
// model of NI USB 6009 device
//
#include <NIDAQmx.h>
#include <vextcpp.h>
#define DAQmxErrChk(functionCall) if( DAQmxFailed(error=(functionCall)) ) goto Error;

void ADCUSB(char* TYP, int NEL,int nindac1,int nindac2,int mnoutadc[8],
int mdio1[8], int mndio2[4], int mode) {

int error=0;
TaskHandle taskHandle=0,taskHandle2=0,taskHandle4=0;
char errBuff[2048]={'\0'};
int32 read;
float64 datad[1000];
uInt8 mdio3[4]={out[mndio2[0]].New,out[mndio2[1]].New,0,out[mndio2[3]].New};
float64 data[2] = {as[nindac1].ANew,as[nindac2].ANew};
// convert data from digital circuits
 for (int i=0; i<4; i++) {
                if(mdio3[i]==3) mdio3[i]=1;  }
// Configuration and start of the task DAC1 and DAC2
DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
DAQmxErrChk
(DAQmxCreateAOVoltageChan(taskHandle,"Dev1/ao0","",0.0,5.0,DAQmx_Val_Volts,""));
DAQmxErrChk
(DAQmxCreateAOVoltageChan(taskHandle,"Dev1/ao1","",0.0,5.0,DAQmx_Val_Volts,""));
DAQmxErrChk (DAQmxStartTask(taskHandle));
DAQmxErrChk
(DAQmxWriteAnalogF64(taskHandle,1,1,10.0,DAQmx_Val_GroupByChannel,data,NULL,
NULL));
DAQmxStopTask(taskHandle);
DAQmxClearTask(taskHandle);
// Read from the ADC channel
DAQmxErrChk (DAQmxCreateTask("",&taskHandle2));
DAQmxErrChk
(DAQmxCreateAIVoltageChan(taskHandle2,"Dev1/ai0","",DAQmx_Val_RSE ,-
10.0,10.0,DAQmx_Val_Volts,NULL));
DAQmxErrChk
(DAQmxCfgSampClkTiming(taskHandle2,"",10000.0,DAQmx_Val_Rising,DAQmx_Val_Fi
niteSamps,2));
DAQmxErrChk (DAQmxStartTask(taskHandle2));
DAQmxErrChk
(DAQmxReadAnalogF64(taskHandle2,1000,10.0,DAQmx_Val_GroupByChannel,datad,1000
,&read,NULL));
DAQmxStopTask(taskHandle2);
DAQmxClearTask(taskHandle2);
```

```
// Configure and run the digital output task
// used to control the multiplexer
DAQmxErrChk (DAQmxCreateTask("",&taskHandle4));
DAQmxErrChk
(DAQmxCreateDOChan(taskHandle4,"Dev1/port1/line0:3","",DAQmx_Val_ChanForAllLine
s));
DAQmxErrChk (DAQmxStartTask(taskHandle4));
DAQmxErrChk
(DAQmxWriteDigitalLines(taskHandle4,1,1,10.0,DAQmx_Val_GroupByChannel,mdio3,NU
LL,NULL));
DAQmxStopTask(taskHandle4);
DAQmxClearTask(taskHandle4);
// output the value from the ADC input to the mnoutadc contact number
as[mnoutadc[0]].ANew=datad[0];
// display an error message if a possible incorrect configuration
Error:
        if( DAQmxFailed(error) )
                DAQmxGetExtendedErrorInfo(errBuff,2048);
        if( taskHandle!=0 ) {
                /*********************************************/
                // DAQmx Stop Code
                /*********************************************/
        }
        if( DAQmxFailed(error) )
                printf("DAQmx Error: %s\n",errBuff);
}
```

The ADCUSB NIDAQ 6009 subsystem is the average performance criterion – performance. The transfer of parameters is possible in the form of a data structure and a list of circuit numbers. Using the list of circuit numbers is more convenient when describing systems in the form of formalized tasks. External multiplexers expand the functionality of subsystems and increase efficiency.

Conclusion

The methodology and operation algorithms of generalized multifunctional component models with a common interface are created. New multifunctional models of components with a common interface are associated with common data for various models and applications, and allow you to get many applications from one description of an object. Multifunctional component models with a common interface significantly reduce the complexity of design, especially in the initial stages.

4. IMPLEMENTATION OF A MULTILEVEL SYSTEM OF MODELING AND DESIGN OF HETEROGENEOUS COMPUTING SYSTEMS

4.1. The structure and functions of the software package

The subsystem for selecting a formalized job and the initial values of the **SETSEL** parameters is presented in the form

SETSEL = <SETFZ, SETLD, SETLM, SETLIB, SETCADIN, SETSERV, SETLOC>,

where **SETFZ** is the set of names of formalized tasks;

SETLD – many valid description languages;

SETLM – many valid message languages;

SETLIB – many valid types of libraries (static - lib, dynamic dll);

SETCADIN – many valid descriptions for input CAD systems;

SETSERV – many available servers for viewing materials or completing tasks;

SETLOC – the name of the disk (letter) of the location of the system catalogs (COD, CODOS), examples and object libraries (FA) of the complex.

The PRJSEL design results selection subsystem is represented as

PRJSEL = <SETCADOUT, SETINTF>,

where SETCADOUT is the set of valid CAD output;

SETINTF – many used interfaces for output CAD systems.

The subsystem for choosing the presentation of RESSEL design results is presented in the form

RESSEL = <VIEWTXT, VIEWAD, VIEWSCH, VIEWMOD, VIEWNET>,

where **VIEWTXT** is a message viewer for one or many options;

VIEWAD – many programs for viewing diagrams of digital and analog signals for multivariate analysis;

VIEWSCH, VIEWMOD, VIEWNET – many programs display circuits, modules and network objects;

The set of RPRJ design rules consists of the following subsets:

RPRJ = <RNAMEVAR, RMCADIN, RMCADOUT, RFTSCH, RFTAB>

where **RNAMEVAR** – rules for the formation of variants of names of design results;

RMCADIN – rules for selecting modules and functions for filling out a table of a variant of a scheme;

RMCADOUT – rules for choosing modules and functions for creating a variant of a specific CAD design;

RFTSCH – rules for selecting modules and functions for extracting data from a table of a variant of a scheme;

RFTAB – rules for selecting modules and functions for extracting data from the table of information about the components of the circuit.

During the design process, a formalized task in a high-level language, representing many technical solutions, is converted into a text or tabular form of the level of one solution, which can be converted into a text, command or tabular format of a specific CAD system or into a formalized task [12–14]. The number of options is determined by a formalized task.

	1. Formalized task (FT)	
2. Search models for paths and names of the FT (CodRead)	Component Models (Comp)	Models of control procedures and signal generation functions (Control)
3. Data structure schema variant (TSCH)		
4. Methods for reading component libraries adding and writing a data structure of a schema or macro file	Reading, Writing, and Search Methods in the Schema Variant Data Table (TSCH)	Methods for reading, writing and searching for a component in the table of correspondence (Table)
5. Methods for reading, adding and writing data structure of a component, its conclusions and circuits	Methods for reading, adding, and writing symbol tables and constructs	Methods for searching, reading and writing information about component contacts and connected circuits
CAD-specific project file or standard data exchange structure		

Fig. 4.1. The generalized structure of multi-level CAD

The process of converting a formalized task into a description for a specific industrial CAD system using the COD software-methodical complex is as follows. For each description variant, a TSCH table is generated (see fig. 4.1). The formalized task is translated and edited with the corresponding model of procedures integrated into the library. When performing a formalized task, the model of a specific component calls up the information table and places information about the component in the table of the variant diagram (see fig. 4.2). Links or connections are populated in the TSCH table from the formalized job. Thus, in the table of a variant of the circuit (see table 4.6), all information about the components, the outputs of the components, and the relationships with other components appears. After the formation of the TSCH, the functions of generating a description of the variant of the scheme for a particular CAD are included. The functions of the fourth level (see fig. 4.1) provide the processing of components, and the functions of the fifth level provide the conversion of information on the outputs of the components.

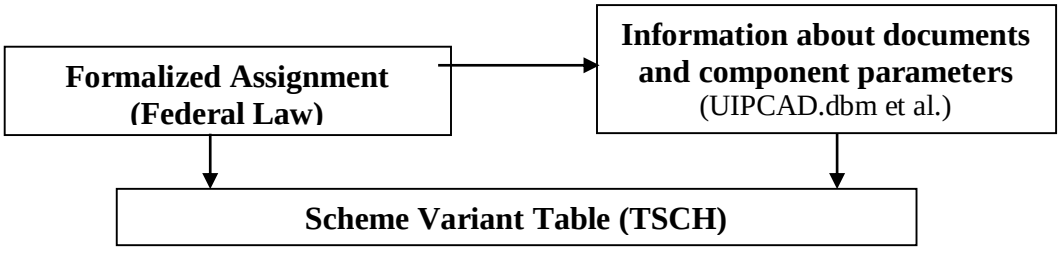

Fig. 4.2. Formation of a table of a variant of a scheme

Function prefixes are given in table 4.2, the main functions – in table 4.3. For example, the function for opening correspondence tables is designated DOpen or DbnOpen, and for closing them it is designated DClose or DbnClose. The function of opening a component library file is indicated by LOpen or LibOpen, closing закрытия LClose or LibClose. The function of adding circuits is NetAdd, the function of adding elements is ElmAdd, the function of adding circuits is SchAdd. The rules for generating the names of COD shell commands are presented in table 4.4, a table of specific procedures and interface functions is given in table 4.5.

Table 4.1

Main File Table

Module name	Module purpose
uiorcad, uipcad	Lower Level Procedures and Functions
control	Control modules
Comp	Top-Level Component Models
Codread codini	Functions for reading paths and file names of CAD UI
table	Functions for working with uipcad component tables
Tsch	Functions for Working with Tsch Schema Variant Tables
Errors	Message Output Functions

Table 4.2

Function Prefix Table

Function prefix	An object
P, Pif	CAD interface <COD> with industrial CAD
L lib	Component library
D, dbt, dbn	table of correspondence
TS, TSch	Chart table
TE, Telm	Item table
Name	Object name
Typ	Object type
S, Sch	Scheme
E, Elm	Element
N net	Chain
C, Con	Conclusion
Pin	Contact
Msg, Err	Error messages
PinList	Contact list
Netlist	Net list

Table 4.3

Table of main functions

Name the functions	Assigning Object Processing Functions
Open	Opening files
Close	File closing
Srch, find	File search, table search

Continuation of the table 4.3

Copy	Copy files, rows in a table
Update, Upd	Updating files, rows in a table
Add	Adding records to a file, rows to a table, chains to a component
Del	Delete records, contacts, circuits
Draw	Adding a conductor bus

The TSCH interface (scheme variant table) represents the output of the scheme description structure to the file <FT name>.tsh in text format. This file may be required in case of debugging modules, as well as if full information about the components of the circuit is required. Similar XML-structured information is output to an xml file. An XML file type is used to graphically display schemas.

Table 4.4

COD Shell Command Name Rules

Symbol Assignment	room	Value	Semantics of meaning
The way to perform a formalized task	1	F	Local
		R	OS / 2 server
		P	P390 Server
		H	S390 Server
		W	WWW server
		L	Create LIB
		I	DLL creation
		M	Common interface module
Formalization job description language	2	P	PL / 1 (PLI)
		C	C ++ (CPP)
		J	Java
		A	ADA
		H	Vhdl
Type of operating system	3	D	Dos
		O	OS / 2
		W	WINDOWS
		V	VM
		H	MVS (OS390)
		U	Unix
		L	LINUX
CAD package for a specific field	4	F	Functional models
		G	Common interface
		R	PRAM5.3
		P	PCAD
		O	ORCAD
		C	CADDY
		A	AUTOCAD, EAGLE
		V	VRML
		E	Edif
		EI	Eagle command interface
		S	STEP (EXPRESS)
		X	XML
		Z	Tsch

Continuation of the table 4.4

		V	VRML
		I	CATIA
		L	ALTIUM
CAD version	5, 6	Figures	Version number
Type of models (interface with CAD)	5, 6 or 7	F	Functional models
		I	Command Line Models
		T	Table Interface Models
		C	Component Description
		N	Circuit Description
		Z	Synthesis of FZ COD

The table of rules for the formation of COD shell command names (table 4.4) is used for local and remote complexes and refers to the base model [87], although it is also used in the information model. The local execution of functional modeling and resource estimation in the Windows environment is performed by the fcwf.bat, fpwf.bat and fjwf.bat commands for the C ++, PL/1 and java description languages, for the OS2 or ECS environment – fcof.cmd, fpof.cmd, fjof.cmd. The application server uses local commands, for example fcop8t.cmd and fcop8i.cmd, for the table and command interface with CAD PCAD8.5, respectively. For CATIA v5 CAD, the macro file command files will be fcoi.cmd, fpoi.cmd, and fjoi.cmd.

Commands for generating static and dynamic modeling libraries – lcof.cmd, icof.cmd, commands for generating libraries of a unified interface – lcogi.cmd, icogi.cmd.

Table 4.5

Table of interface procedures and functions

Object name	Object Assignment	Calling objects (procedures, functions)	Return value
Formalized task (FT)	Description of one or multiple technical solutions	Models of external influences	–
		Component Models	
		Control modules	
Models of external influences	Description of external influences	Elmcopy	–
		Netadd	
Component Models	Description of component models	Elmcopy	–
		Netadd	
Control Procedure Models	Description of control modules for applications	Dbnclose	FT section number
		PifClose, PClose	
PifOpen, Popen	Initial installation of paths and objects on the first call (comp, signal)	–	–
Pifclose	Checking the completion of creating a schema table and displaying the results	Tsclose	–

Continuation of the table 4.5

Tsclose	Creating a macro file body	Codread	–
TSOpen	Creating a schema table	–	–
TSAdd	Adding components to a schema table	–	–
Object name	Object Assignment	Calling objects (procedures, functions)	Return value
CodRead	Search for job name, drawing template, paths and library types	Codsrch	0 – not found 1 – found
ElmCopy	Creating an item in the chart table	Dbnopen	0 – not created N – element number in the chart table
		Elmrsrch	
		Elmadd	
		Absname	
ElmAdd	Adding a component to a schema table	–	0 – not added N – component number in the table
DbnOpen	Search for the correspondence table file sequentially uipcad.dbn, uipcad.dbt and download	Getpin	0 – file not found 1 – file found
		DbnAdd	
DbnClose	Finishing a match table and deleting a table from memory	–	–
ElmSrch	Search for components in the correspondence table by its name	–	0 – not found N – record number
PinSrch	Search for item contact	–	0 – not found 1 – index
AbsName	Checking component type for abstract type	–	0 – type specific 1 – type abstract
ErrOpen	Opens a msg message file	–	0 – not open 1 – open
ErrClose	Closes a msg message file	–	–
ErrAdd	Outputting messages to a msg file	–	–
GetPin	Read contact information from the match table	–	N – contact position
DbnAdd	Increase table space in memory	–	–

Table 4.6

Chart Variant Table

Name	Level	Appointment	C ++, JAVA	PL / 1	Note
Dbnrec	1	Dbm table structure	–	–	–
Nel	2	Item number	Int	Dec Fixed (3)	–
Name	Level	Appointment	C ++, JAVA	PL/1	Note

Continuation of the table 4.6

ElmNamR ElmNamL ElmNamA	2	The name of the component from Russian or Latin letters and numbers or the name of the analogue	char (C ++) String (Java)	Char (*)	*
DOCNamR DOCNamL	2	Name of the document for components from Russian or Latin letters and numbers			*
DOCNamA	2	Analog Document Name			*
P0	2	Static power	Float	Dec float	–
P01	2	Component Switching Energy			–
S	2	Component Area			–
Massa	2	Component weight			–
Price Rel	2	Price relative to two-way valve			–
TYPEElm FunName	2	Function Type and Component Function Name	char (C ++) String (Java)	Char (*)	*
FunCode	2	Component Function Code	Int	Dec Fixed (3)	–
KElmPack	2	The number of elements in the housing		Dec Fixed (3)	–
TYPEPack	2	Type of shell	char (C ++) String (Java)	Char (*)	–
KPinPack	2	The number of contacts in the housing	Int	Dec Fixed (3)	–
NElmPack	3	Case Number		Dec Fixed (3)	–
MPinPack	2	Array of contacts		Dec Fixed (3)	**
KPinIn	2	Number of Input Contacts		Dec Fixed (3)	**
KPinOut	2	Number of Output Contacts		Dec Fixed (3)	**
MPinIn	2	Array of inputs		Dec Fixed (3)	**
MPinOut	2	Array of outputs		Dec Fixed (3)	**
PinListE	2	Item Pin List	EPinList		–
PinType	3	Contact type	Int	Dec fixed	–
PinLeq	3	Type of logical equivalence		Dec Fixed (3)	–
PinPCAD	3	Output name in CAD PCAD	char (C ++) String (Java)	Char(*)	–
PinCod	3	Output Name in COD		Char(*)	–

Continuation of the table 4.6

PinN	3	Contact number	Int	Dec Fixed(3)	–
PinName	3	Contact name	char (C ++) String (Java)	Char(*)	*
NetTYP	3	Type of circuit connected to the pin		Char(*)	*
NetNum	3	Circuit number	Int	Dec Fixed(3)	–

* The absence of a component corresponds to the word null (lowercase letters).

** The field is not written to the file and is valid only in RAM.

Digital input signals are controlled using the parameters of the SIGNAL or SIGNALD generation procedure, calculation of the current bit strings or array of strings. The purpose of controlling the input signals may be to evaluate the maximum frequency of the device's health. The goal contains conflicting requirements for maximum performance (signal frequency) and reliability of the results. Therefore, it is possible to implement an algorithm for controlling the change in the frequency of signals, starting with the maximum, and analyzing the array of estimates of the results in accordance with the expected values. Knowledge is represented by the correspondence of the rules for the formation of signals and expected results. Therefore, the conformity of actual and expected results is a condition for achieving the goal and completing the process. If the condition for achieving the goal is not fulfilled, then the analysis of the options is forcibly completed at the given maximum value. To assess the reliability of the results, it is better to use various types of input signals.

You can quickly manage the current bit string of the generated signal or fill an array of bit strings. The use of external signal models is possible with single or multiple input from a file. A single entry reduces time but increases the size of the model buffer. Examples of working with model buffers are given in the examples.

The input analog signals are controlled using the parameters of the SIGNALA generation procedure, and the actual and estimated analog signals are compared using the SIGNALAC procedure. The first parameter to the SIGNALA procedure is the analog signal circuit number. The second parameter of type float can be a value or an expression. The third and fourth parameters determine the beginning and end of the change in the analog signal. In the absence of additional restrictions, the minimum parameter is NTMIN as the third parameter, and the maximum NTMAX as the fourth parameter.

The number of the analyzed variant can be used as one of the indices of the two-dimensional array of signal models, which will provide a convenient form of multivariate analysis. For univariate and multivariate analysis, you can use any type of monitor: IPRINT, CPRINT or APRINT. The structure of the system can be controlled statically (before it starts to work) and dynamically (during operation). The modular structure of computers and systems allows you to change the configuration before starting work. To change the connections during operation, it is necessary to introduce special switching components, – multiplexers and demultiplexers. In multiprocessor computing systems, switching

components are the basis for matching the structure of the system with an algorithm for solving the problem.

High-level language tools allow you to enter variables with a set of valid values as types of components and relationships. Thus, it is possible to realize the representation of the tree of technical solutions.

During the design process, you need to manage the output of the CAPP. As part of the research system, in the analysis control section, the IPRINT or CPRINT control module with the parameters of the minimum and maximum circuit numbers and the boundaries of the time interval is used to output digital signals, the APRINT module with the parameters of the lower and upper boundaries of the digital circuits, the time interval is used to output digital signals and analog circuits. The output of digital or analog-to-digital signals is controlled by the format when accessing any control module. The LRES variable defines the output options. The value 0 corresponds to the graphical conclusion of the results, 1 – the formation of the symbolic time diagram of the work and the resource assessment table, 2 – additionnally displays a signal comparison table.

The user model is represented by the LU structure with various values: initial, old and new. The value 0 corresponds to the prohibition of controlling the design process, 1 – the possibility of a single-variant analysis, 2 – the possibility of a multivariate analysis, 3 – the control of external influences, 4 – the change in structure and 5 – optimization.

The main task in the development of research CAD was a single description of the object for various applications. Various applications are selected from the menu and run with the same basic description. This saves the labor of an engineer in the design process. Examples of various applications are: assessment of resources, formation of specifications, conversion of descriptions to a PCAD system file, import of descriptions from various CAD systems. The design paths in CAD COD during the execution of various applications are shown in fig. 4.3.

Important is the analysis of computing devices with component failures in various ways.

The synthesis of descriptions of heterogeneous computing systems and the execution of various applications is provided by various CAD components, including software and information.

Consider the basics of information support on the example of research CAD.

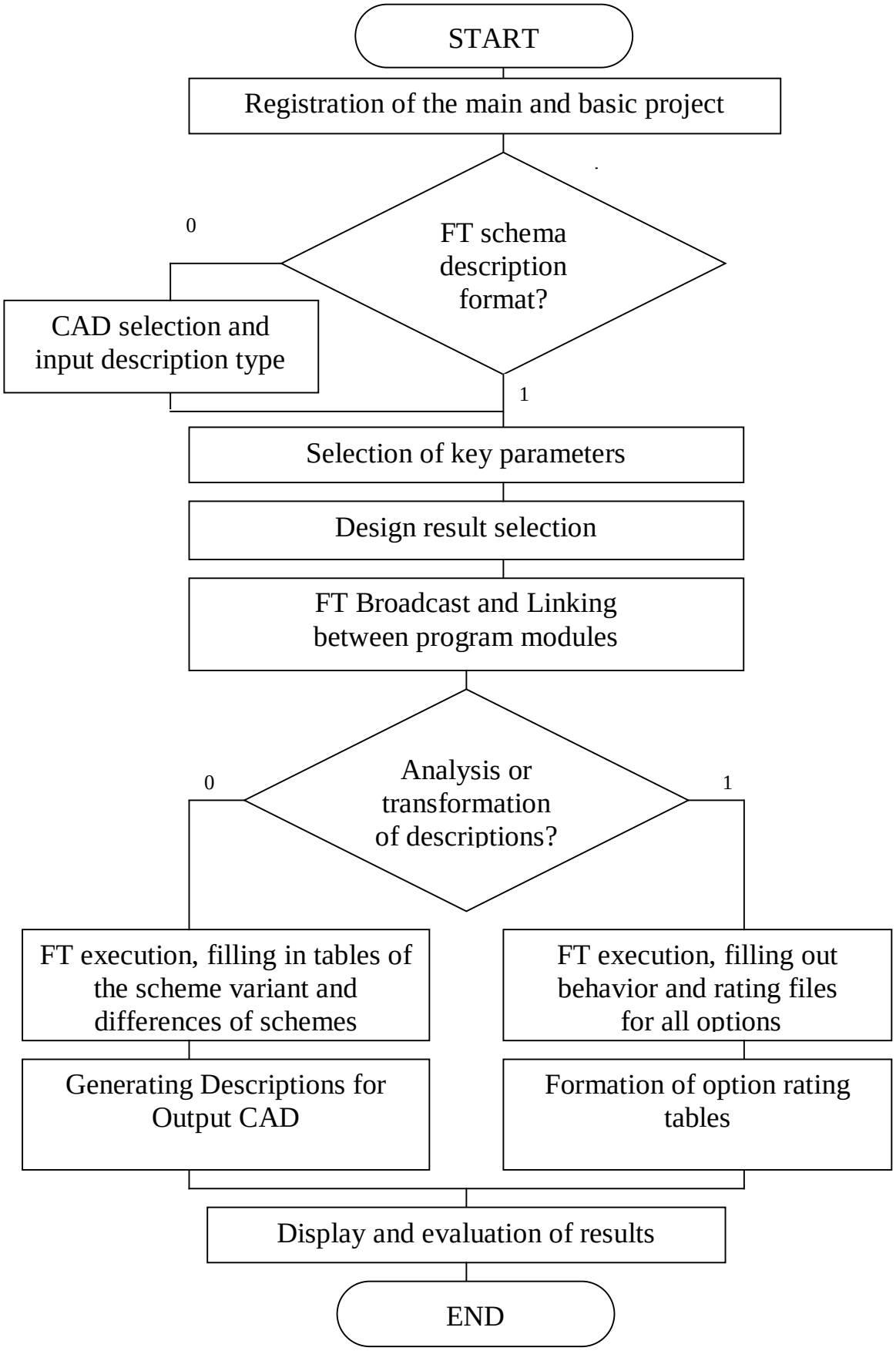

Fig. 4.3. Design Routes in CAD COD
(FT – formalized task)

4.2. Information support of modeling and design of computing devices and systems

The states of signals in the analysis process are represented by the values of data structures. The type of data depends on the power of multiple states. Real signals correspond to analog signals, binary signals – one bit. However, binary signal models are used for analysis only at the logical level. With an increase in the power of many signal states, the transient is analyzed in more detail, but the analysis time significantly increases. Therefore, they use minimum complexity models of signals sufficient to achieve the goal of analysis. It is known to use five-digit and even nine-digit models of digital signals [31, 84], but ternary ones are most often used.

Binary signal models in synchronous models are used to detect device malfunction caused by gross synthesis errors. Trinity models allow [31, 113, 114] to detect the static risk of failure, consisting in an erroneous change in the signal at the output of the element due to delays in the input signals.

The dynamic risk of failure [31, 113] corresponds to a multiple change in the output signal instead of a single one and can be caused by adverse combinations of input signal delays or resonance in the input circuit at the threshold level. A five-digit representation of the signal values allows you to detect the dynamic risk of failure. Examples of solving a system of logical equations in binary and ternary bases are given in [31]. An example of a system of logical equations is given for the circuit in fig. 4.4. The equations are made in the reverse order of the signal propagation:

```
NEW(010)=NEX(008)!NEX(009);
NEW(009)=^(NEX(007)&NEX(005));
NEW(008)=NEX(006)&NEX(007);
NEW(007)=NEX(004)&NEX(005);
NEW(006)=NEX(002)!NEX(003);
```

The depth of the KDEL logic, or rank, is three. Therefore, the maximum number of iterations does not exceed KDEL + 1, i. e., four. Additional iteration is necessary to test the stability of presults. The data structure of the representation of digital and analog signals contains fields of new (NEW) values and resultsprevious iterations (NEX). Separation of signal value fields allows implementing the principle of single assignment and eliminating the influence of the number of iterations on the result. Writing is done in the fields of new values (NEW), and reading from the fields (NEX) according to chain number. Therefore, the description of any logical circuit can be compiled in an arbitrary order.

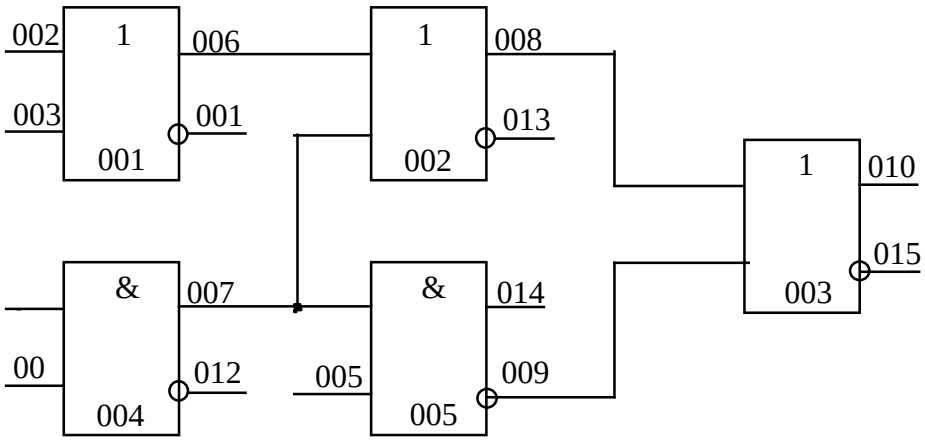

Fig. 4.4. Logic for calculating an iterative process

In order to reduce analysis time, one should strive to describe the circuits along the propagation of signals, for which the usual practice of designing circuits from left to right and from top to bottom is sufficient.

In multilevel CAD systems at the upper levels of abstraction, generalized models of components are used, which are concretized as we move to the lower levels. For the transition from abstract models to concrete, additional information is required. Consider the transformation of descriptions in the transition from the functional level to the design level.

At the functional level, the main is the function performed by the component or node of the computing device, and at the design level -A geometric model of the component, which includes the model of the case, the spatial arrangement and numbering of the contacts. On the functional level, the result of converting information suitable for a variety of input circuits, is fed to the output circuit. At the same time, the spatial arrangement of the circuits does not matter, but the level of interference affects the reliability of the conversion of digital information and determines the error in the operation of analog and analog-to-digital nodes.

For automatic conversion of descriptions of computing devices of a functional level into descriptions of the design level of a particular CAD, software and information support are required. Information support may be presented in the form of tables or databases. In CAD UI and manuals [33], database files are of the db or dbf type, and table filestype dbt, dbn or dbm. In the file of the dbm type, along with the component function code for CAD PCAD, the function name and the type of function that corresponds to the component model procedure name are contained. Databases and tables contain identical information about the components, but differ in the data format [28]. Component property tables are intended for information support of component symbol formation, automatic conversion of a formalized task option into a specific CAD scheme, import and export of exchange structure [1]. Therefore, tables of type dbt in the description of the outputs contain only the type of output, the name of the output in CAD

PCAD and the name of the output in the UI CAD COD. The formal description of the output (DO) is presented in the form

<DO>::=<output type> <output name in CAD PCAD>
<output name in UI CAD COD>.

Tables of the dbn or dbm type are intended for information support of the formation of symbols and component constructs and automatic conversion of a formalized task option into a diagram or description of the design level of a particular CAD system. A formal description of the output for a table of type dbn is represented as

< DO>::= <output type> <equivalence group> <contact number>
<output name in CAD PCAD>
<output name in CAD UI "COD">.

The formal description of the output for a table of type dbm has the form

< DO>::= <output type> <equivalence group> <contact number>
<output name in CAD PCAD>
<output name in CAD UI "COD">
<circuit type> <circuit number>.

All the above tables are formally described as follows:

<Table>::= <text describing the set of components (DSC)>
< DSC>=::<component description (CD)> [<CD>]
<CD>::=<description of component parameters>
<description of component pins (DCP)>
<DCP>::=<description of the group of conclusions (DGC)>; [<DGC>; ... < DGC>]
< DGC>::=< DO>, [<DO>, ... < DO>]

The names of the parameters and properties of the contacts and circuits are given in table. 4.6. Examples of type tables dbt, dbn, dbm are located in subdirectories X:\COD\BIN and X:\CODOS\BIN, and can also be found in subdirectories with component libraries for industrial CAD systems. For CAD ORCAD version 9, this subdirectory will be X:\FA\O9, for CAD PCAD8.5 the subdirectory will be X:\FA\P8. The first two rows of the tables are marked with a comment sign (/), and column headings are placed in them.

Attention should be paid to the description of components for which the functions of individual sections of logical elements have different or equivalent functions. In this case, the component power leads are not described in each section, but in the last description of the terminal group. The descriptions of the

output group of the individual sections are separated by a semicolon, and their number is not limited. After the last section, a semicolon may be missing.

In CAD COD at the functional level, power pins are not used, but they are necessary to go to the design level. To distinguish the power leads, the following notation rules have been adopted: 0V – digital common conductor; 0VA – analog common conductor; UP, UP1, UP2 – output of positive supply voltage in ascending order; UN, UN1, UN2 – output of negative supply voltage in increasing order in absolute value.

Descriptions of computing devices in high-level languages [9, 14] may contain abstract and specific types of components. Specific types of components are characterized by the function they perform, resources and parameters are packaged in a housing. Abstract types of components are characterized by a generalized or specific function. An example of an abstract component is a multiplexer of analog and digital signals, a serial or parallel adder of arbitrary capacity. Existing industrial CAD systems typically only work with specific types of components. Examples of such CAD systems are PRAM5.3 and PCAD (MD). The volume of descriptions of computing devices in low-level languages in graphical or textual form significantly exceeds the similar description in high-level languages. Therefore, it is effective to describe technical solutions at a higher level of abstraction, in high-level languages, and automatically convert them into descriptions for industrial CAD. The implemented interface of research CAD with industrial CAD, features of software and information support are considered.

4.3. Formalized task for automated analysis

The formalized task for automated analysis consists of the following sections:

INIT – declaration and initialization of parameters and components;

INPUT – input of external signals;

UNIT – description of the circuit;

MOD – internal procedures – component models;

CTRL – control module.

A feature of a formalized task is the obligatory numbering of all chains that can be declared. Single or group chains are permitted to be named.

The section for declaring and setting initial parameter values contains a common part that is included by the preprocessor using the %INCLUDE operators – TXTPAR sections. The individual parameters for a specific job can be changed using assignment operators, starting with the INIT label. For example, record the required number of signals – thirteen and the maximum number of ticks – sixteen:

```
INIT: NS=13; NTMAX=16;
```

then declare arrays of structures of analog and digital signals, registers – %INCLUDE TXTDCL. In the same section, all single and group signals, individual for a specific task, are announced. For example, they declare a bit string – VXOD input signal model – DCL VXOD BIT (16). The beginning of the external signal input section is the INPUT label, and external signals are generated using the special SIGNAL and SIGNALA procedures.

If the signal change (the difference between the fourth and third parameter) exceeds the length of the bit string, the signal periodically repeats the sequence specified by the bit string. Signal generation is explained in detail in a separate section. Analog signals are generated using standard functions.

The schema description section begins with the UNIT label. The description of the circuit is done component-wise by calling the generalized procedures of the component class. The procedure names for all multiplexers are MX, for logic LO. A particular component is identified by a type – a character string. The type of multiplexer is – '564KP1', the type of logic circuit is – '564LA7', or their analogs corresponding to the library. Connections are described component by component by setting the parameters of the circuits. For example, descriptions of the component and compounds in PL/1 are of the form:

```
UNIT: CALL LO('564LE10', 001,006,009,007,003);
CALL LO('564LA7', 002.007.011.004.002);
```

in C and C ++:

```
UNIT: LO('564LE10', 001,006,009,007,003);
LO('564LA7', 002.007.011.004.002);
```

in JAVA language:

```
case 3: {// UNIT:
        comp.LO('564LE10', 1,6,9,7,3);
        comp.LO('564LA7', 2,7,11,4,2);}
```

various examples are given in [27, 31].

Section of internal procedures component models – necessary when incorporating new elements or inadequacy of existing models. The procedure parameters are circuit numbers. The software model of the component performs actions on data structures with an array index corresponding to the chain number. The procedures provide input and output control of forbidden code combinations of signal values, for example, '10'B [31, 112–114]. Each internal procedure is checked by test examples.

The control module section consists of IPRINT procedures for outputting digital signals and an APRINT module for outputting digital and analog signals. IPRINT procedure parameters:

1) the minimum circuit number, starting with which the signal is printed;

2) the maximum number of the circuit whose signal is supposed to be printed;

3) the minimum measure number from which the signal is printed;

4) the maximum measure number for which the signal is printed.

In order to reduce translation time, the control module is connected by the editor from the module library.

The included text W is a conditional operator: it is used to control the iterative process of automated analysis and transfers control to input external signals (INPUT), initial setup (INIT), or circuit description (UNIT). In the source include library, the names of the entry point select statements correspond to the names of the procedures. For example, when analyzing logic circuits, use the %INCLUDE WLO operator. For PL/1, all declarations of procedure entry points are described in the WALL.INC file, and for C ++, all function and procedure definitions are described in the UICPP.H file.

4.4. CAD parameters and data structures

In order to use the minimum resources of RAM and computer time, user-controlled parameters have been introduced. The parameter text (TXTPAR) when using only static libraries is as follows:

```
DCL(NS INIT(64),NAS INIT(09),LRG INIT(8), KAPT INIT(8),
NM INIT(1), NRG INIT(16))EXT DEC FIXED(3);
DCL(NT INIT(0),NPT,NTMIN INIT(1),
NTMAX INIT(32))EXT DEC FIXED(6);
DCL(TTIME INIT(0),DELT,DELTP INIT(400),
DELTN INIT(600))EXT DEC FLOAT;
DCL(M_S INIT('1'B),PPT)BIT(1)EXT,ERC CHAR(1)EXT;
DCL LAB(3) LABEL INIT(INIT,INPUT,UNIT), (NL INIT(0),
LRES INIT(1), LU INIT(2)) DEC FIXED(3) EXT;
DCL (C0 INIT('00'B), CF INIT('01'B), CZ INIT('10'B),
C1 INIT('11'B)) EXT BIT(2);
DCL(XMAX, YMAX) EXT DEC FLOAT;
```

The text is included in the formalized task using the preprocessor operator %INCLUDE TXTPAR or TEXTPAR, which must be before the ad text included by the %INCLUDE TXTDCL or TEXTDCL operator.

The text of PL/1 declarations when using static libraries looks like

```
DCL RG(NRG)EXT CTL BIT(LRG);
ALLOCATE RG;RG='0'B;
DCL RGM(NRG,NM)EXT CTL BIT(LRG);
ALLOCATE RGM;RGM='0'B;
DCL 1 OUT(0:NS)EXT CTL,2((NEW,NEX,OLD,OLS,FUT)
BIT(2), DELD DEC FIXED(3)); ALLOCATE 1 OUT;
OUT='00'B;OUT(001)='11'B; OUT.DELD=0;
DCL 1 AS(0:NAS)EXT CTL,2((ANEW,ANEX,AOLD,AOLS,AFUT)
DEC FLOAT,DELD DEC FIXED(3));
DCL 1 AM(0:NAS,KAPT)EXT CTL,2((ANEW,ANEX,AOLD,
```

```
AOLS,AFUT) DEC FLOAT,DELD DEC FIXED(3));
ALLOCATE 1 AS, 1 AM; AS=0; AM=0;
DELT=DELTP+DELTN; NT=0;TTIME=0;
DCL SIGNAL ENTRY (DEC FIXED(3),BIT(*),
DEC FIXED(6), DEC FIXED(6)) EXT;
DCL TCC(KCC,KCT)EXT CTL DEC FIXED(3);
ALLOCATE TCC;TCC=0; TCC(*,3)=NTMIN;TCC(*,4)=NTMAX;
  DCL TPC(NS,KCT)EXT CTL DEC FIXED(3);
  ALLOCATE TPC;TPC=0;
  DCL(GG,GC,GS,GM,GP0,GP01,GX,GY)
  INIT(0) EXT DEC FLOAT;
  GG=0;GC=0;GS=0;GM=0;GP0=0;GP01=0;GX=0;GY=0;PP=-1
```

When using both static and dynamic libraries, all declarations are executed in the parameter text (TXTPAR.INC), and the text of the TXTDCL.INC declarations consists of a single line of the call to the procedure for placing objects in memory: CALL ALLOUT.

The text of the procedure is in the ALLOUT.INC file and is given below:

```
/* TEXT AN OBLIGATORY FOR MOGOBAP AND ANCHANT AHALYSIS
*/
 ALLOUT:PROC;
 ALLOCATE RG; RG='0'B;
 ALLOCATE RGM;RGM='0'B;
 ALLOCATE 1 OUT; OUT='00'B;OUT(001)='11'B;OUT.DELD=0;
 ALLOCATE 1 AS,1 AM;AS=0;AM=0;
 PPT=^M_S;NPT=1+^M_S;DELT=DELTP+DELTN;
 NT=0;TTIME=0;XT=TTIME/DELT;JCC=1;
 ALLOCATE TCC;TCC=0;
 TCC(*,3)=NTMIN;TCC(*,4)=NTMAX;
 ALLOCATE TPC;TPC=0;
 DCL (MPP(NVARMAX),MKEFC(NVARMAX),
 MKEFM(NVARMAX),MKEFP(NVARMAX))
 CTL EXT DEC FLOAT;
 IF NVAR=1 THEN ALLOCATE MPP,MKEFC,MKEFM,MKEFP;
 GG=0;GC=0;GS=0;GM=0;GP0=0;GP01=0;GX=0;GY=0;PP=-1;GPF=0;
 END ALLOUT;
```

The text of the TXTPAR.INC parameters when using both static and dynamic libraries has the form

```
/ * Ad text for external DLL procedures * /
DCL(NS,NAS,NRG,NM,KAPT,LRG,ERC,NL,NPPD,NPPA)EXT DEC
FIXED(3)
```

```
RESERVED (IMPORTED);
DCL(NT,NPT,NTMIN,NTMAX)EXT DEC FIXED(6) RESERVED
(IMPORTED);
DCL(TTIME,XT,DELT,DELTP,DELTN,PP,LRA,ERA)EXT
DEC FLOAT RESERVED (IMPORTED);
DCL(NVAR,NVARMAX,KCT,KCC,JCC,KPC,LRES,LU)EXT
DEC FIXED(3) RESERVED (IMPORTED);
DCL(M_S,PPT)EXT BIT(1);
DCL RG(NRG)EXT CTL BIT(LRG);
DCL RGM(NRG,NM)EXT CTL BIT(LRG);
DCL ( C0 INIT('00'B),CF INIT('01'B),CZ INIT('10'B),
   C1 INIT('11'B)) EXT BIT(2);
DCL 1 OUT(0:NS)EXT CTL,2((NEW,NEX,OLD,OLS,FUT)BIT(2),
   DELD DEC FIXED(3));
DCL 1 AS(0:NAS)EXT CTL,2((ANEW,ANEX,AOLD,AOLS,AFUT)
   DEC FLOAT,DELD DEC FIXED(3));
DCL 1 AM(0:NAS,KAPT)EXT CTL,2((ANEW,ANEX,AOLD,AOLS,AFUT)
   DEC FLOAT,DELD DEC FIXED(3));
DCL(TCC(KCC,KCT),TPC(NS,KCT))EXT CTL DEC FIXED(3);
DCL(GC,GS,GG,GM,GP0,GP01,GX,GY,XMAX,YMAX)EXT DEC FLOAT ;
DCL (GPF,KEFC,KEFM,KEFP)  EXT DEC FLOAT;
```

The text of declarations for multivariate analysis in C ++ is located in the TXTDCL file and is given below:

```
if (nrg!=0)
     rg = new BitString[nrg];
  {
     register int i;
     for ( i=0 ; i<nrg ; i++ )
        { rg[i].value=NULL;
          rg[i].Alloc(lrg);  }
    }
  if (nrg!=0 && nm!=0)
     rgm = new BitString[nrg,nm];
  {
     register int i,j;
     for ( i=0 ; i<nrg ; i++ )
      for ( j=0 ; j<nm ; j++ )
       { rgm[i,j].value=NULL;
          rgm[i,j].Alloc(lrg);     }
    }
  out = new DigOut[ns+1];
  memset(out,0,sizeof(DigOut)*(ns+1));
  out[1].New = out[1].Nex = out[1].Old = out[1].Ols = out[1].Fut = C1;
```

```
as = new AnalogOut[nas+1];
memset(as,0,sizeof(AnalogOut)*(nas+1));
am = new AnalogArray[nas+1];
memset(am,0,sizeof(AnalogArray)*(nas+1));
tcc = new tstr[kcc];
memset(tcc,0,sizeof(tstr)*kcc);
tpc = new tstr[ns+1];
memset(tpc,0,sizeof(tstr)*(ns+1));
tcas = new tsas[kcas];
memset(tcas,0,sizeof(tsas)*kcas);
ppt = !m_s;    npt = 1;
delt = deltp + deltn;
jcc=0;// index of table tcc in proc. SignlDC
gs=gc=gm=gpf=gp0=gp01=gx=gy=0;
pp=-1;
if (nvar==1)
{
  mpp=new float[nvarmax];mkefc=new float[nvarmax];
  mkefm=new float[nvarmax];mkefp=new float[nvarmax];
  memset(mpp,0,sizeof(float)*nvarmax);
  memset(mkefc,0,sizeof(float)*nvarmax);
  memset(mkefm,0,sizeof(float)*nvarmax);
  memset(mkefp,0,sizeof(float)*nvarmax);
 }
```

The above ad text places objects in RAM. The formalized task includes the text UICPP.H, including the necessary header files, prototypes of the functions of component models and control modules, as well as the texts of declarations of external variables and data structures TXTPAR and VEXTCPP.H, which are located in the INCLUDE subdirectory.

The text of declarations of external variables and data structures, as well as methods for their placement in SetOut() memory in the JAVA language, are located in the VEXT.JAVA file, which after translation is placed in the COD subdirectory and in application packages located in the JAVA subdirectory:

```
package cod;
import cod.outstruct;
import cod.asstruct;
import java.util.Vector;
public class vext
{
 public static Vector outlist = new Vector();
 public static int lres=-1, nvar = 1, nvarmax = 1, nl = 1, lu, lun, nppd=0, nppa=0;
 public static boolean m_s=true, ppt;
 public static int nt, ntmax=8, ntmin=0, npt, erc=0, niter=0;
```

```
public static int kct = 8, kcc = 1,jcc=0, ns=0, nm=1, nas=0, lrg=8, nrg=0;
public static float delt, deltp=1, deltn=1, ttime, xt, pp , lra=-1, era=-1;
public static outstruct out;
public static asstruct as;
public static int tcc[][], tpc[][];
public final static byte C0 = 0, C1 = 3, CF = 1, CZ = 2;
public static BitString rg[], rgm[][];
public static float[] mpp, mkefc, mkefm, mkefp;
public static String bin(byte i)
{  if(i == C0) return "00";    if(i == C1) return "11";
   if(i == CF) return "01";    return "10"; }
public static void SetOut()
{  int n = ns + 1;
   out.New = new byte[n];    out.Nex = new byte[n];
   out.Old = new byte[n];    out.Ols = new byte[n];
   out.Fut = new byte[n];     out.Deld = new int[n];
   out.New[1] =  C1;          out.Nex[1] =  C1;
   out.Old[1] =  C1;          out.Ols[1] =  C1;
   out.Fut[1] =  C1;
 if(nas > 0)
   SetAout();
   SetRg();
 delt = deltp + deltn;   ppt = !vext.m_s;  npt = 1;
 tcc=new int[kcc][kct];  tpc=new int[ns+1][kct];
 mpp = new float[vext.nvarmax+1]; mkefc= new float[vext.nvarmax+1];
 mkefm = new float[vext.nvarmax+1];mkefp= new float[vext.nvarmax+1];  }
 public static void SetAout()
{   int n = nas + 1;
   as.ANew = new float[n];  as.ANex = new float[n];
   as.AOld = new float[n];   as.AOls = new float[n];
   as.Deld = new int[n]; } public static void SetRg()
{   int i,im;
   rg = new BitString[nrg+1];
   for(i=1;i<nrg+1;i++) rg[i] = new BitString(vext.lrg);
   rgm = new BitString[nm+1][nrg+1];
   for(im=1;im<nm+1;im++) for(i=1;i<nrg+1;i++)
   rgm[im][i] = new BitString(vext.lrg); }
}
```

The text of declarations places arrays of structures of the required volume in RAM. The maximum size of the arrays is limited by the format of the parameters and the actual amount of RAM.

Memory is allocated for RG registers with an LRG length and an NRG capacity. The number of digital signals is set by the NS parameter of the analog NAS signals, the duration of the steady-state sub act is determined by DELTP,

transition – DELTN. The duration of a measure is calculated as the sum of the results. The name NT defines the number of the current measure, and the interval boundaries are named NTMIN and NTMAX. The current time is called TTIME, the sub act number is – NPT. The ternary model of digital signals corresponds to a single value of the variable M_S, and the binary – corresponds to zero. The ERROR byte is used to indicate the level of error when using various programs. The general dictionary of models is given in table 4.7.

Table 4.7

General Model Dictionary

No. p / p	Name	Appointment	Formats	data
			PL/1	C, CPP, JAVA
1	2	3	4	5
1	ANEW	The new value of the analog signal	Dec float	Float
2	ANEX	The following value of the analog signal		
3	AOLD	The old value of the analog signal in the steady-state sub-cycle		
4	AFUT	The future value of the analog signal in the steady-state sub-tact		
5	AS	An array of analog signal structures	(0: *) EXTCTL STRUCTURE	Analog Out
6	DELT	Tact duration	Dec float	Float
7	DELTN	Transitional Subtact Duration		
8	DELTP	Steady-State Subtact Duration		
9	INIT	Description of initial parameter values	LABEL CONSTANT	
10	INPUT	Beginning of the description of external signals	LABEL CONSTANT	
11	KCC	The number of compared signals	DEC FIXED (3.0)	Int
12	KCT	Table dimension		
13	LAB	Tag array	(3) AUTOMATIC, INITIAL, LABEL	
14	LRG	Register length	DEC FIXED (3.0)	Int
15	NAS	Number of analog signals	Dec FIXED (3.0)	
16	NEW	New signal value	IN OUT (0: *) BIT (2)	unsigned int: 2
17	NEX	Next signal value		
18	NITER	Iteration number	DEC FIXED (3.0)	Int
19	NITERMAX	Maximum number of iterations		
20	NRG	Number of registers		
21	NS	Number of Digital Signals		

Continuation of the table 4.7

22	NT	Measure number	DEC FIXED (6.0)	Int
23	NTMAX	Max measure number		
24	NTMIN	Min measure number		
25	NVAR	Option Number		
26	NVARMAX	Option Number Maximum		
27	OLD	The old value of the signal in the steady-state sub-cycle	IN OUT (0: *) BIT (2)	unsigned int: 2
28	FUT	Future signal value		
29	OUT	Array of digital signal structures	(0: *), EXT, CTL STRUCTURE	Digout
30	PPT	Type of Subtact	IN OUT (0: *) BIT (1)	unsigned short int
31	TCC	Signal Comparison Chart	(*) CTL, EXT, DEC, FIXED (3.0)	Tstr
32	TPC	Signal parameter estimation table	(*) CTL, EXT, DEC,	
33	UNIT	Schema Description Label	LABEL CONSTANT	LABEL
34	C0	Signal Zero	C0 BIT (2) INIT ('00 ')	Const unsigned C0 = 0
35	C1	The unit value of the signal	C1 BIT (2) INIT ('11 ')	const unsigned C1 = 3
36	CF	Signal Edge Value	CF BIT (2) INIT ('01 ')	Const unsigned CF = 1
37	CZ	Prohibited Signal Value	CZ BIT (2) INIT ('10 ')	Const unsigned Cz = 2
38	GC	Cost, rel. units	Dec float	FLOAT
39	GS	Area, mm sq.		
40	GM	Mass g		
41	GP0	Power at low frequency, mW		
42	GP01	Switching energy, mW · s		
43	GX	Current abscissa		
44	GY	Current ordinate		
45	GPF	Power, mw		
46	PP	Performance		
47	KEFC	Criterion: “cost – performance”		
48	KEFM	Criterion: “mass – productivity”		
49	KEFP	Criterion: “power - performance”		
50	DELD	Output Time Delay	DEC FIXED (3.0)	Int
51	AM	An array of analog signal structures	(0: *) EXT CTL STRUCTURE	Analog Out

Continuation of the table 4.7

52	NPPD	PP Rating Digital Signal Number	DEC FIXED (3.0)	Int
53	NPPA	PP Rating Analog Signal Number		
54	LRA	Effective bit depth		
55	ERA	Error	Dec float	Float
56	CADNAME	CAD output	CHAR (*)	* char
57	CADIN	Input CAD		
58	CODLOC	CAD drive name	CHAR (1)	* char [1]
59	CODE	Job Name (FT)	CHAR (*)	* char
60	FRAME	Type of format (A1 – A4)	CHAR (2)	* char [2]
61	Language	FT description language	CHAR (*)	* char
62	Langmsg	Message Language		

The states of digital and analog signals are stored in OUT, AS, and AM structures, respectively. To fulfill the rule of single assignment, the names are divided: new (NEW), next (NEX), old established (OLD) and future (FUT) values. In procedures (functional models of components), the names of the new values of the output signals are placed in the left parts of the assignment operators, and the names of the following values for combinational circuits and various combinations of names for automata are located in the right parts.

The array of LAB labels contains values of valid constant labels INIT, INPUT, UNIT with the possibility of expansion, and the current label number is determined by the variable NL.

Declaring an entry point to the SIGNAL signal generation procedure is necessary to use bit strings – digital signal models of indefinite length.

The procedures of the component model and signal generators are calculated, and the values are entered into the elements of the arrays of structures of digital and analog signals in accordance with the circuit numbers specified when the procedures were called. In this way, the operation of interconnected components of varying complexity is simulated.

4.5. Interfaces of research CAD COD with PCAD design automation system

The PCAD Data Interchange Format (PDIF) format is a symbolic counterpart to internal file formats such as SYM, SCH, PRT, PCB, PS. PDIF file formats are discussed in detail in [134] and are used to convert circuit and component files during the transition from the younger versions of PCAD to the older ones, as well as for interfaces with other design automation systems. For example, in PCAD 2000-2006 CAD, PCAD version 8.5 schematic files are downloaded directly, and PCAD version 4.5 schematic files, after being converted by PDIFOUT.EXE, are converted into PDIF format as a PDF file. It is possible to use the PDIF format for automated library generation.

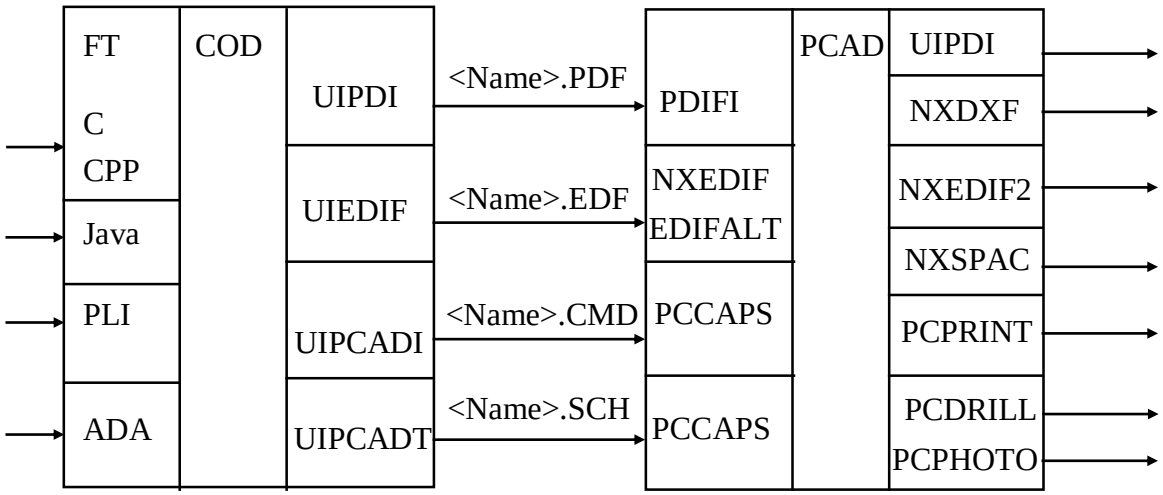

Fig. 4.5. Interfaces of research CAD COD and PCAD

The interface of the research CAD CAD with the PCAD system can be implemented by generating batch files for the interactive graphics editor PCCAPS with its subsequent execution. The execution of batch files is convenient for observing the process of input and placement of component symbols and input connections of component outputs. The component pins are connected if the same circuit is connected to different component pins. Component symbols must be in the appropriate library of type SLB in the directory X:\FA\P4 or X:\FA\P8. The batch file is launched for execution from the research CAD menu or using the command line: PCCAPS @ <file name>.CMD. The batch file contains the command to enter the PCCAPS environment setup file, including the frame of one of the standard formats.

	1 Formalized task (FT)	
2 Search Models for Paths and Names of the FT (CodRead)	Component Models (Comp)	Models of control procedures and signal generation functions (Control)
	3 Chart Variant Table (TSCH)	
4 Methods for reading, adding, and writing schemas (Sch)	Methods for reading, searching library files (Slb, Plb)	Methods for reading and searching for a component in the correspondence table (Table)
5 Methods for reading, adding, and writing PCAD tables	Methods for reading, adding, and writing symbol tables (SYM) and constructs (PRT)	Methods for searching, reading and writing component contact information

Fig. 4.6. The multilevel structure of the tabular interface UI CAD with CAD PCAD

The CAD UI interface with PCAD can be performed using functions of various levels. The lower-level functions work with the data structures of a particular lower-level CAD system, such as PCAD. The functions of the upper

levels do not depend on the specific CAD of the lower level and on how the interface is implemented. In fig. 4.6 shows the multi-level structure of the table interface with CAD PCAD, and fig. 4.7 – a multi-level structure of the command interface with various low-level CAD systems. The command interface of CAD UI with one low-level CAD system, for example PCAD, can be implemented using functions of even one level. However, in this case, the component models will be different for each specific CAD system. The multi-level structure of the interface allows the use of common component models found in the comp file, and highlight the many interface functions for various lower-level CAD systems. The functions for finding resource paths and the names of formalized tasks are in the codread file. The search is performed first in the environment variables, then in the cod.ini file.

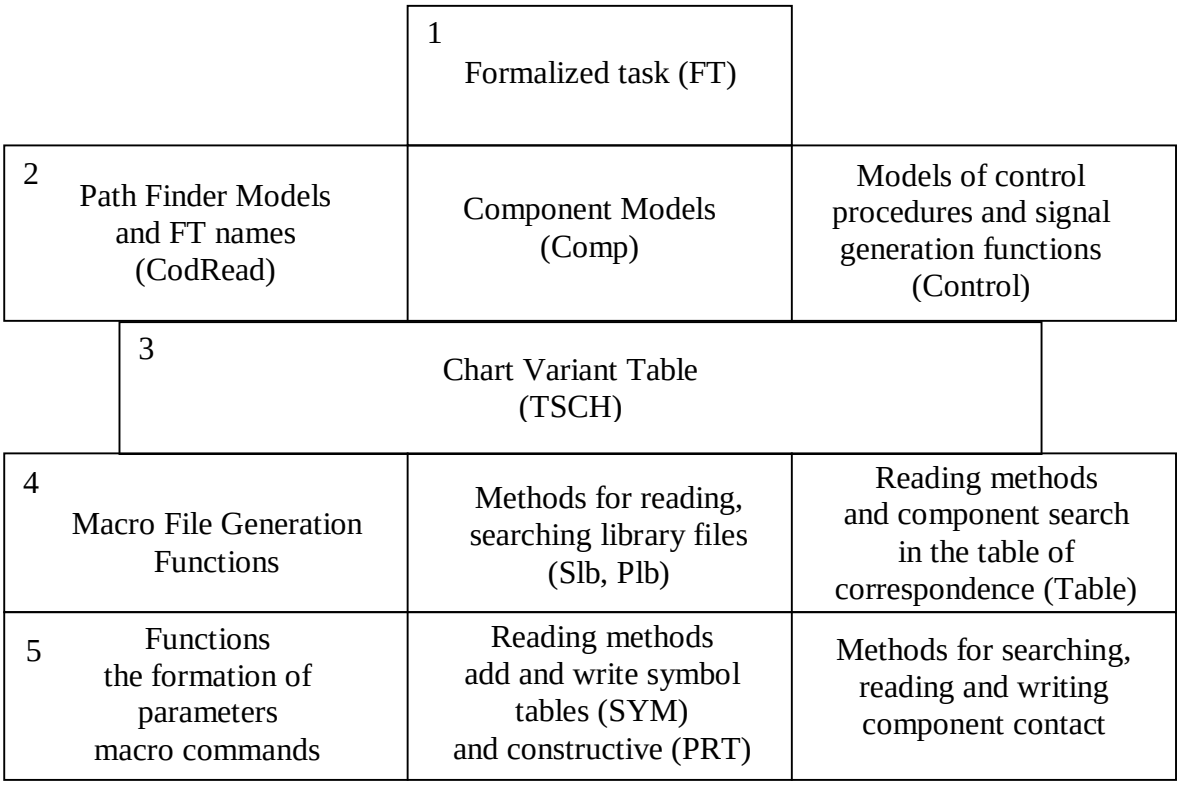

Fig. 4.7. The multi-level structure of the command interface UI CAD with CAD PCAD, ORCAD

This approach allows the use of common interface software on the workstation and on the server. On the workstation, the information is in the cod.ini file, and registration on the server does not make sense and the information is located in the environment variables of the job being performed. Models of control procedures and signal generation functions are more dependent on lower-level CAD systems and are combined into a control file. Functions of the middle-level component model (C, Component) are placed in the same file, which ensure the independence of the top-level component models from the type of applications. Between the functions of the third and fourth levels, a data structure is formed that fully describes the components and the connections between them.

The data structure is called a schema table and is denoted by TSCH. The TSCH data structure is similar to the uipcad.dbn mapping table, functions of work with which are summarized in the table file. Compared to uipcad.dbn entries, the TSCH schema table is supplemented by the names of the circuits connected to the corresponding component pins. The lower (fifth) level of functions forms the methods of reading, adding and writing data in the CAD table PCAD shown in fig. 4.6. Methods of reading, adding and writing symbol tables (SYM) and constructs (PRT) provide work with components in the CAD environment PCAD. At the same level are methods for searching, reading and writing information about the contacts of a component. Based on the methods of working with symbols and component constructs, methods are created for searching and reading symbol library files such as slb and plb construct libraries, which constitute functions of the fourth level.

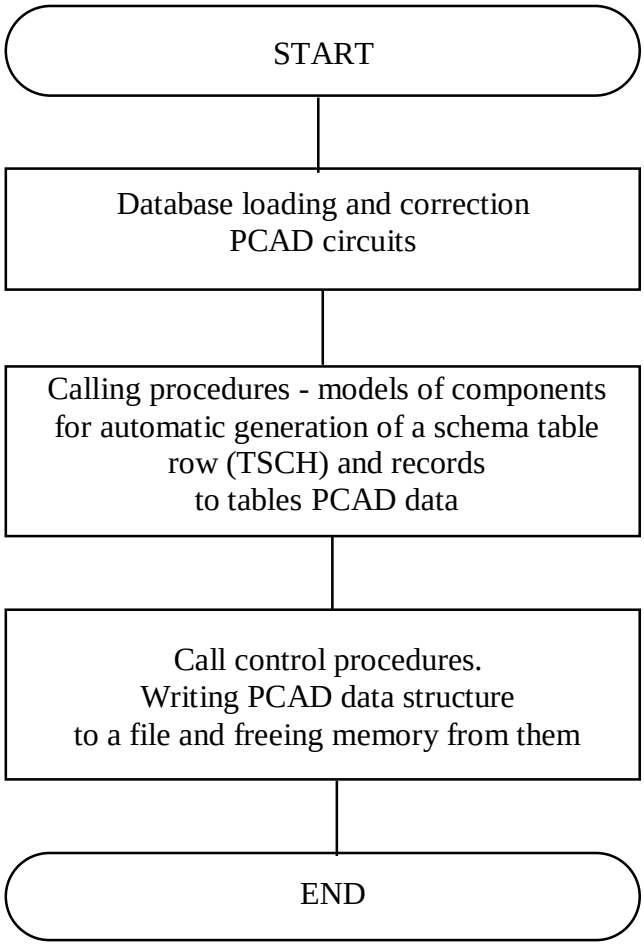

Fig. 4.8. Block diagram of the CAD interface with PCAD by automatically generating component symbols and links in the PCAD database format (tabular interface)

The automatic generation of device descriptions in the PCAD database format is fast and lacks visibility. The diagram of the algorithm for implementing the CAD interface with PCAD is shown in fig. 4.8. The description of the device in the PCAD database format is a set of relationships in the form of tables with cross-references, schematically shown in fig. 4.9. It is necessary to distinguish between table formats for CAD versions PCAD up to and including 4.5 and for

versions from 6 and higher. The former are 16-bit versions of PCAD CAD and have database version 1 (1.04 for PCAD 4.5), and the latter are 32-bit versions with database 2 (2.09 for PCAD 8.5). Starting from version 8.5, a component list table (complist) has been added that contains the component serial number, component names and additional data.

The interface of research CAD with industrial is implemented by redefining the functions of component models. A feature of UI CAD is the multiple use of a single formalized task for various applications and multifunctional component models with a common interface. Component models are combined into libraries in accordance with the application and are named in accordance with table 4.4. For modeling in C++ in the MS DOS environment, the library of component models LCDF.LIB is used, for the interactive interface – the library of component models LCD8I, and for the implementation of the tabular interface - the library of model models LCD8T. Depending on the PCAD version, a number is added to the library name according to the table 4.4. For example, for PCAD 4.5, the table interface model library is named LCD4T, and for version 8.5 it is called LCD8T. All model libraries should be in the X:\COD\LIB directory for the MS DOS environment, in the X:\CODOS\LIB directory for the OS/2 environment, and in the X:\CODOS\LIBWIN directory for the Windows environment. If several versions of the PCAD system are installed on the machine, the one used must have a PCAD root directory. In OS/2 and Windows, the shell allows you to select the current version from PCAD4.5 and PCAD8.5. The symbol libraries of components of the SLB type and constructs of the PLB type must be located in the subdirectory P4 or P8 of the FA directory, which additionally contains circuit files of the A1–A4 format and computer module constructs of the BRD type. In OS/2 and Windows, the shell allows you to select the current version from PCAD4.5 and PCAD8.5. The symbol libraries of components of the SLB type and constructs of the PLB type must be located in the subdirectory P4 or P8 of the FA directory, which additionally contains circuit files of the A1–A4 format and computer module constructs of the BRD type. In OS/2 and Windows, the shell allows you to select the current version from PCAD4.5 and PCAD8.5. The symbol libraries of components of the SLB type and constructs of the PLB type must be located in the subdirectory P4 or P8 of the FA directory, which additionally contains circuit files of the A1–A4 format and computer module constructs of the BRD type.

Each type of interface corresponds to its own library of component models with a single interface defined in the UICPP.H file for the C++ language, in the WALL.INC file for the PL/1 language and in the COMP.JAVA file for the JAVA language. As for modeling, all procedures of component models have two entry points that differ for PL/1 using arrays of inputs or outputs – ending the name of the NIN procedure, or specifying the minimum circuit number and their number – ending the name of the MIN procedure. The main entry point uses an array of chains, and in the procedure with the second entry point, the arrays of circuit numbers are filled and the main procedure is called. Declaring a common name and selecting a specific entry point for the PL/1 language is done by the

GENERIC operator. Examples of declarations are provided in the WALL.INC file for OS/2 and Windows environments, and for the VM environment, in the WALL file.

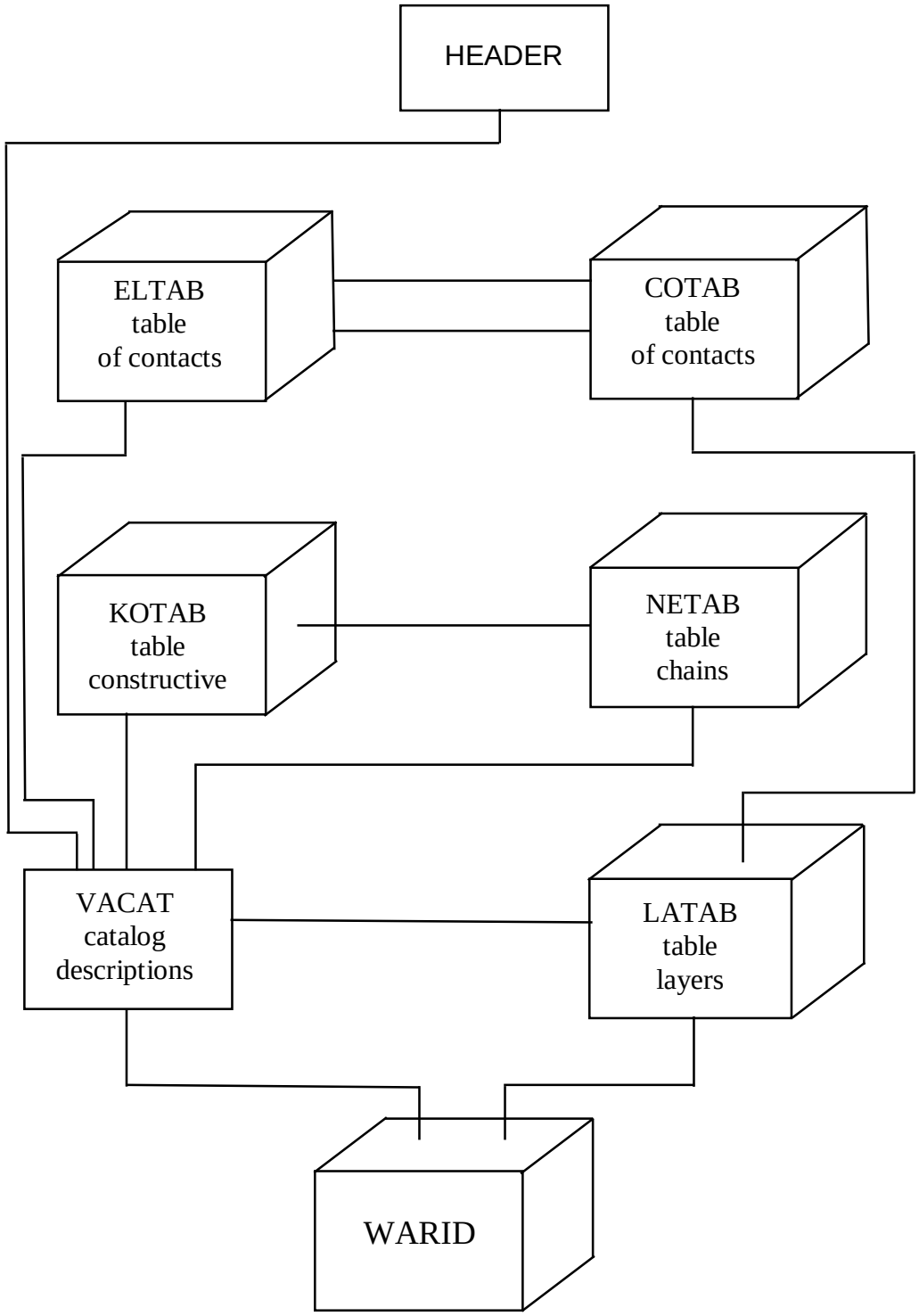

Fig. 4.9. Basic circuit description tables in PCAD

To familiarize yourself with the automated PCAD CAD interface, you should select a formalized task with specific types of components contained in the tables UIPCAD.DBT, UIPCAD.DBN. Carry out a simulation and, using a tabular interface, get a schematic diagram for the selected version of PCAD. Check the resulting *.SCH file using the PCCAPS interactive editor and simulate the circuit

according to the route. The simulation results in the research system are compared with the results obtained in PCAD. Compare the complexity of text and graphical input description schemes.

When using one version of CAD engineering design, for example PCAD, the following subdirectories in the catalog of research CAD COD are sufficient:

BIN – executable modules, tables UIPCAD.DBT, UIPCAD.DBN, PAC.DBT;

LIB – for the MS DOS environment, a library of functions with bit strings lcdbits.lib (bitscpp.lib), functions for working with arrays of integers lcdintv.lib (intvect.lib), functional models of components lcdf.lib (uicpp.lib), interface models components and libraries for other applications in accordance with table. 4.4. For the OS/2 environment, libraries for working with bit strings lcobits.lib, with arrays of integers lcointv.lib, and additional libraries in accordance with |table 4.4;

LIBWIN – for Windows environment libraries of functions for working with bit strings lcwbits.lib, with arrays of integers lcwintv.lib and additional libraries in accordance with table 4.4;

LIBLIN – for the LINUX environment, UNIX library of functions for working with bit strings lclbits, with arrays of integers lclintv and additional libraries in accordance with table 4.4;

INCLUDE – included texts txtpar, txtdcl, vextcpp, vextc, header files uicpp.h, uic.h, access to table and other header files;

INCPLI – the included texts txtpar.inc, txtdcl.inc, vext.inc, the text of the transition to the section label of the formalized task w.inc and the file for selecting entry points to wall.inc procedures;

DOC – documentation;

EFF – programs, tables and diagrams for evaluating the effectiveness and choosing a variant of a computing device or system. Interactive examples of parametric and structural optimization;

JAVA – archives of simulation classes and a unified interface, archives of classes for displaying XML files and an applet for displaying time diagrams. In a separate directory there are class archives for displaying the language of virtual reality.

The interface with CAD PCAD can be performed both on the workstation and on the server. At the workstation, the user registers a formalized task, determines the format of the drawing, and selects its local or remote execution. Therefore, on the workstation, all information about the task is in the COD.INI file, in the X:\COD\BIN or X:\CODOS\BIN directory. Registration on the server does not make sense, since tasks for execution arrive at random times, and the received message should be the source of information about the formalized task. The received message sets the CODE environment variables with the job name and FRAME with the drawing format. Therefore, the search for information about the task is performed first in the variables of the operating environment and only if it is not in the COD.INI file. The formalized task is transferred to the server, it is executed, and the results are taken to the workstation and deleted on the server.

The results are displayed in graphical form on the workstation. This separation of functions provides an effective solution to CAD problems in a computer network. As the operating systems on the server, Windows, OS/2, Linux, VM/S390 and zVM can be installed.

4.6. Design routes for computer modules in the PCAD200x system

The PCAD200x system [29] runs in the Windows environment and is used for designing computer modules and training thanks to the visual graphic editors of the circuitry (SCHEMATIC) and design (PCB) levels.

CAD PCAD200x is a development of PCAD CAD [17, 18, 22] in the Windows environment. Therefore, computer module files in the format PCAD v8.5 and higher are perceived in the PCAD200x system. Files in PCAD 4.5 format are accepted after conversion to PDIF format by PDIFOUT utility. PCAD200x files are imported by CAD ALTIUM DESIGNER.

CAD PCAD200x consists of three interactive editors: Schematic diagrams, PCB constructs, and Library Manager. By opening the PCAD2001 folder, you can see three icons corresponding to the interactive editors. You can call any of them. It is only necessary to ensure that the paths to the subdirectories and settings are in the configuration files sch.ini, pcb.ini and comp.ini CAD PCAD2001. Changing the settings is performed in the menu by the Options group of commands. For design purposes, the pcb.ini file should contain information about the trace program. If the program [QuickRoute] is used, then the operator should be replaced with RouterExe = C:\<dirn>\QROUTE.EXE. In the described section, <dirn> corresponds to the PCAD200x CAD subdirectory, and <dirw> corresponds to the subdirectory in which Windows is installed.

Work in CAD PCAD2001 is performed in a windowed environment with typical groups of commands represented by drop-down menus and icons of "quick" buttons. The correspondence of the commands and macros used to automate the synthesis of circuits is given in table. 4.8.

Consider the route of designing computer modules using CAD PCAD2001, shown in fig. 4.8. The main design path in the PCAD2001 system uses a graphical representation of the computer module descriptions created using the Schematic graphics editor. To work with a graphical editor, libraries must be prepared containing all the components used, which are stored in files of the LIB type.

Table 4.8

A subset of the PCAD2001 commands – PCAD2006 for the interface

Team assignment	Team name	Subcommand Name	Macro Team (v. 15)
Open new file	File	New	SendKeys"{alt+f}{n}"
Open file		Open	SendKeys"{alt+f}{o}{filename*}{enter}"
Save file		Save	SendKeys"{alt+f}{s}"
Setting units	Options	Configure	SendKeys"{alt+o}{c}{m}{enter}"

Continuation of the table 4.8

Library installation	Library	Setup	SendKeys“{alt+l}{s}{tab}{a}{libname}{enter}{o}”
To cut	Edit	Cut	SendKeys“{alt+e}{t}”
Copy		Copy	SendKeys“{alt+e}{c}”
Embed		Paste	SendKeys“{alt+e}{p}”
Cancel		Undo	SendKeys“{alt+e}{u}”
Name the circuit (place port)	Place	Port	SendKeys“{alt+p}{o}{space}{space}{portname}{enter}”
Enter character in PCAD2000		Part	SendKeys“{alt+p}{p}{space}{space}{partname}{enter}” In PCAD2001 {space} delete
Insert wire		Wire	SendKeys“{alt+p}{w}
Enter bus		Bus	SendKeys“{alt+p}{b}{enter}”

Note. Instead of libname and filename, the full name is written, including the path. Decryption of special characters: {:} – {shift + semicolon}, {\} – {backslash}, {.} – {period}.

The synthesis of the circuit occurs in the Schematic editor.

One of the following options for obtaining the scheme is possible:

- manual creation of a diagram in the Schematic editor;
- getting the circuit created in PCAD 8.5 from a file of type SCH.

After the circuit is synthesized, it is necessary to create a net-type netlist file (Utils Generate Netlist command). Having created the circuit list file, you can proceed to the stages of design and tracing, which are performed in the PCAD200x PCB editor. To load the circuit, select the Utils Load Netlist command, then place the components on the board. Tracing is performed automatically (Route Avtorouters command) with a graphical representation of the process.

The input of the trace program is a description of the module with the components placed and the rules (strategy) of the trace in a file of type STR. If the trace results are unsatisfactory in terms of the number of undiluted connections, the length of individual conductors or other parameters, re-arrange the components and repeat the tracing process until satisfactory results are obtained.

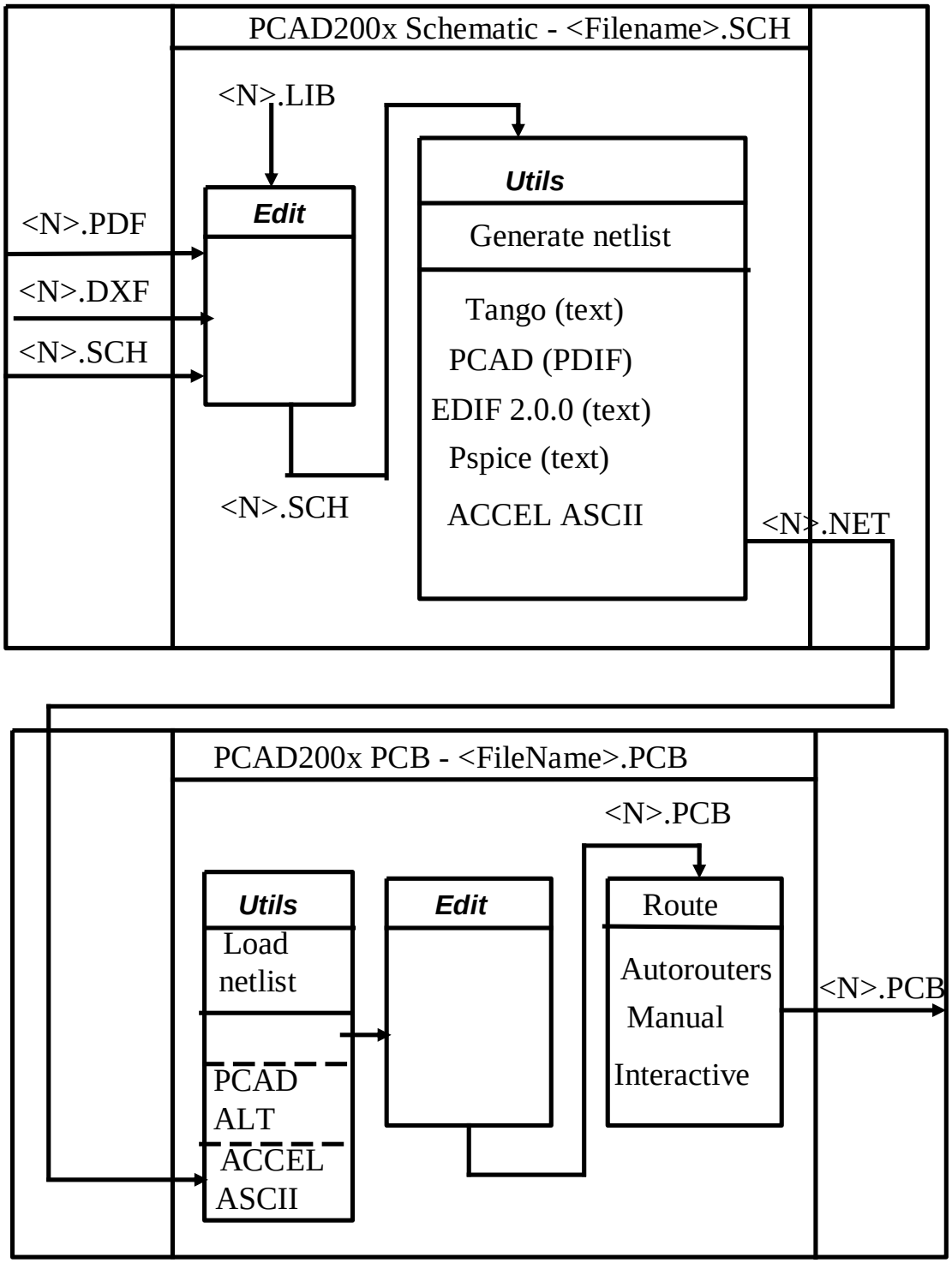

Fig. 4.10. The route of designing computer modules using CAD systems PCAD2000, PCAD2001 – PCAD2006

4.7. The results of formalized tasks

An example of an optimal variant of an ADC unit is a micromodel of a roughly accurate interval ADC in the file fp06rgt.cpp. With the addition of 15 % of the hardware, the performance of the ADC is increased by an order of magnitude due to the interval method of operation. Two additional comparators determine the position of the input signal inside or outside the interval

[17, 18, 22]. Outside the du interval from the average value, only the highest (coarse) section of the counter is allowed to work, and when the input signal is inside the interval, the work of the exact section is also allowed. Thus, the performance of the ADC is increased. In multi-channel mode, the estimated value is entered into the counter through the installation inputs, and the conversion time per channel is determined by the deviation from it. In fig. 2.6 is a diagram, and in fig. 4.11, 4.12 – timing diagrams of the second and third variants of the rough-accurate ADC in the fp06rgt.cpp file.

```
/* Roughly accurate ADC FP06RGT.CPP */
#include <uicpp.h>
void main () {
int lct = 4, ninp  = 14, noutp  = 15, ninp1=17,noutp1=27;
  BitString b01("01");
 /* Announcement of output and input circuits */
IntVector ctout1 (4,6,7,8,9), ctout(4,10,11,12,13),
   ctin (4,0,0,0,0), kin (4,2,3,0,4),
      mloin(2,19,20), mp(2,14,30), m15(2,15,15), m26(2,26,26);
  #include <txtpar>
INIT: /* Initial installation */
lres=-1; ns=32; ntmax=40; nas=5;nppd=4;nppa=3;lra=8;
 nvarmax=3; ttime=0; deltn=1000; deltp=9000; //ns
  #include <txtdcl>
INPUT: /* External influences */
 float du=0.2*(nvar-1);
  SignalA(2,2*cos(xt/40+1)+2,ntmin,ntmax);
  SignalA(4,as[3].ANex+du,ntmin,ntmax);
  SignalA(5,as[3].ANex-du,ntmin,ntmax);
  Signal(3,b01,ntmin,ntmax);
  Signal(2,b01,ntmin,ntmin+2);
UNIT: /* Device Description */
CA   ("K554CA3" ,1,1,19,22,2,4); // UDAC+du
  CA   ("K554CA3" ,2,1,20,23,5,2); // UDAC-du
  CA   ("K554CA3" ,3,1,4,21,2,3);
  LO   ("K561LE5" ,5,5,26,mloin); // Coarse ADC mode - asynchronous
  LO   ("K561LE5" ,7,29,28,m26);  // Coarse ADC mode - asynchronous
  MTJK ("K561TV1" ,4,14,16,0,0,3,28,26);// Synchronous mode output
  MCTR ("K561IE11",6,lct,ctout1,noutp1,ctin,ninp1,kin);// Rough section
  LO ("K561LE5", 10.18.30, m15); // Transfer Permission
  LO ("K561LE5", 11.31, 17, mp); // Transfer Resolution 17
  MCTR ("K561IE11", 8, lct, ctout, noutp, ctin, ninp, kin); // Exact section
  DAC ("K572PA1", 9.3.6.8);
  APrint (2.23, ntmin, ntmax, 2.5);
  #include <w>
  }
```

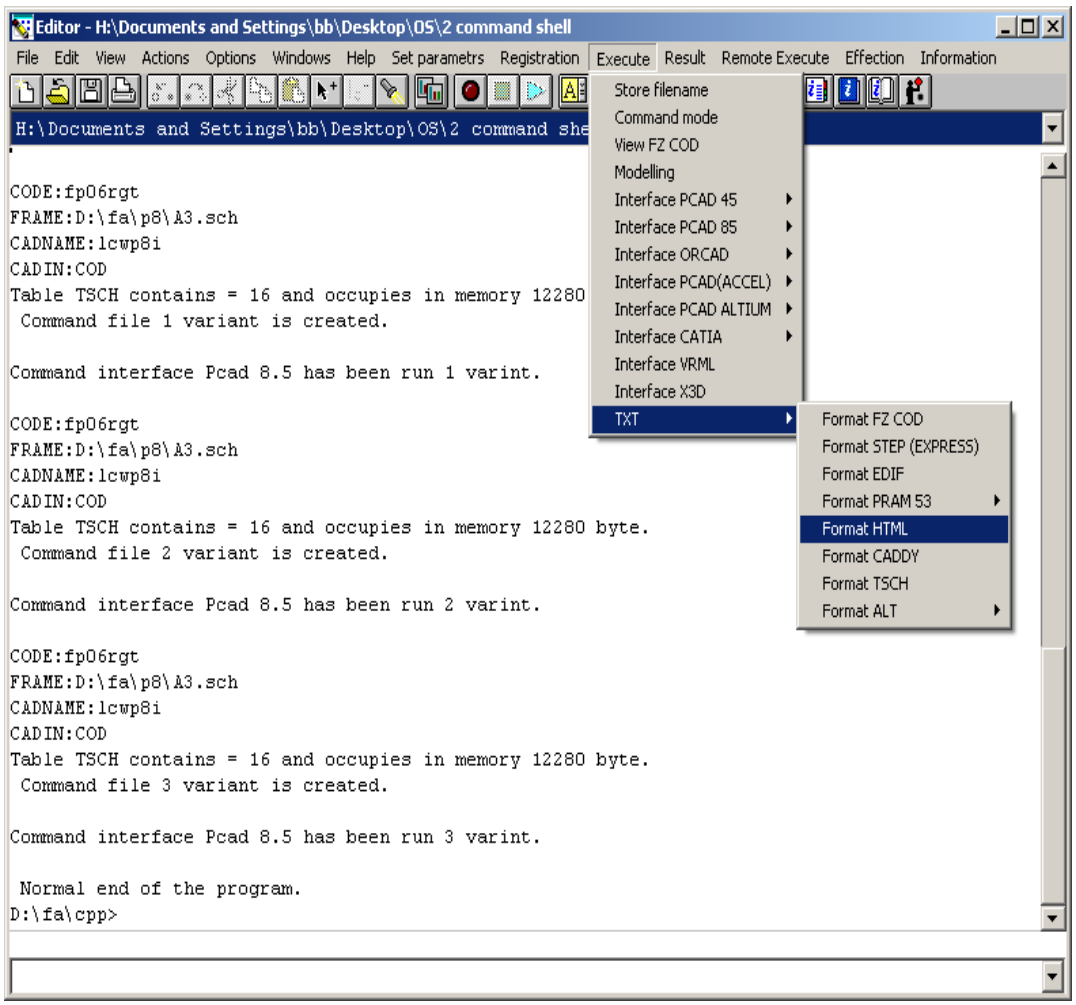

Fig. 4.11. CAD menu COD SFU in LPEX editor IBM Visual Age

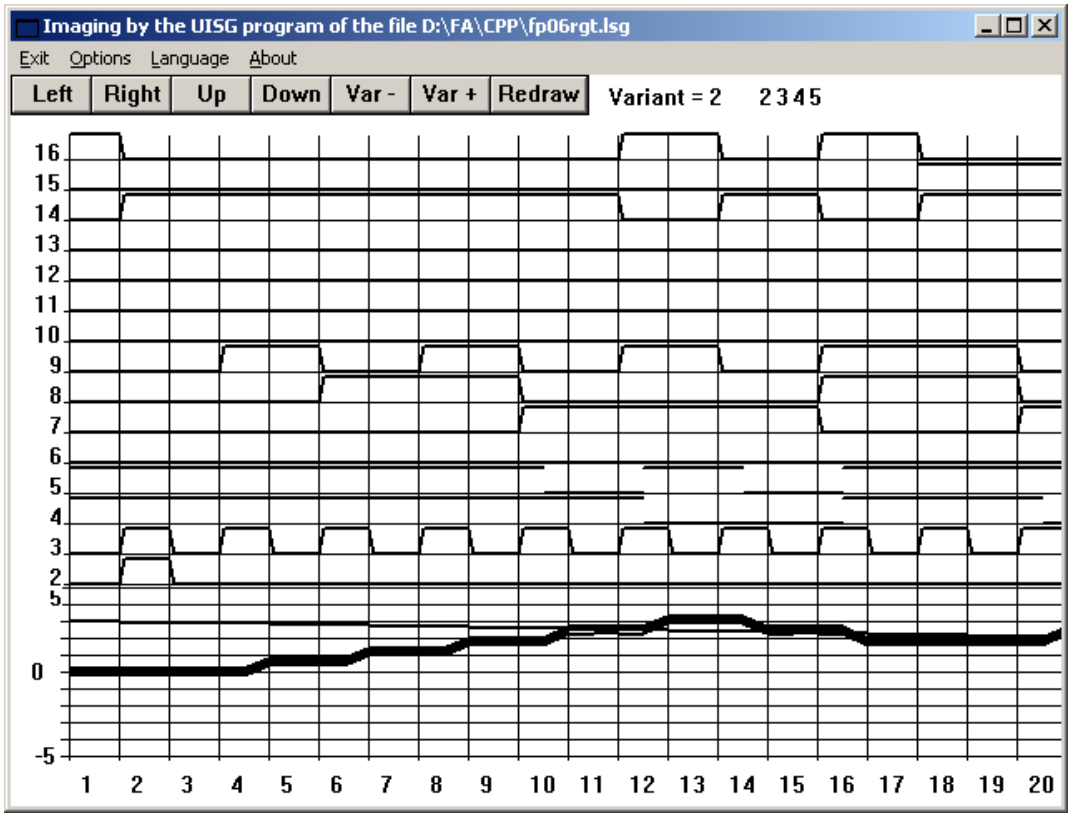

Fig. 4.12. Timing diagram of the second version of the ADC in the file fp06rgt

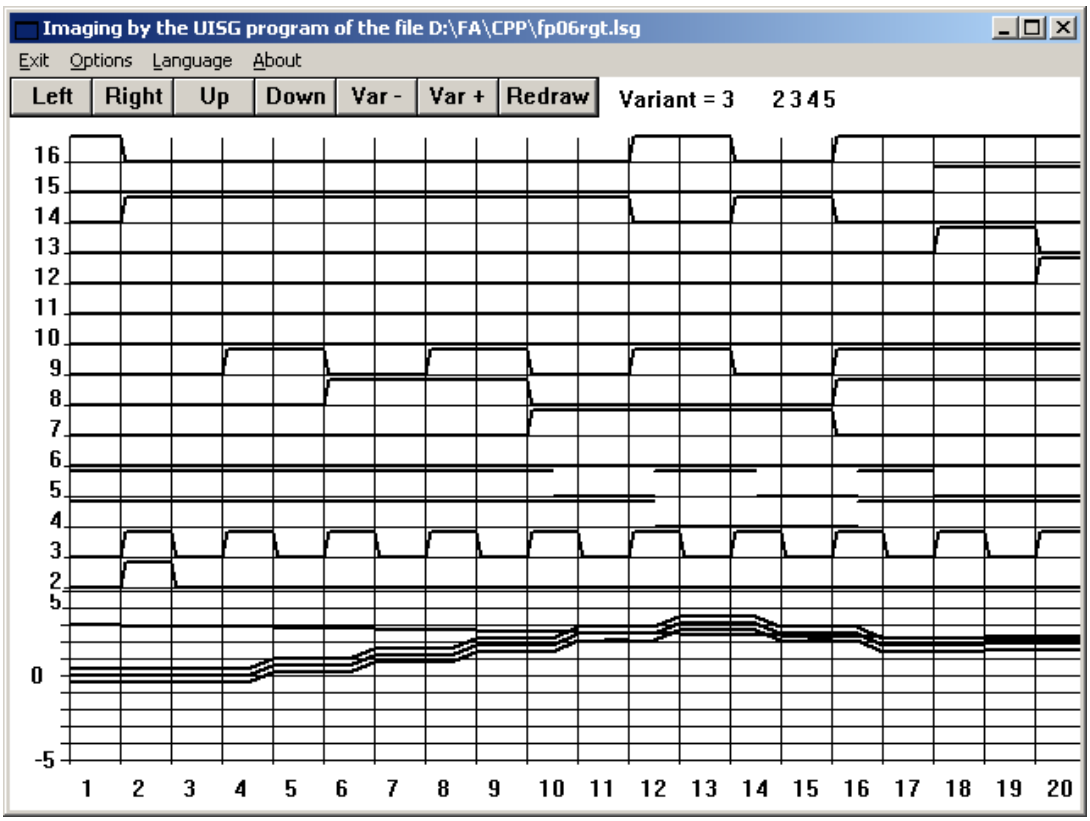

Fig. 4.13. Timing diagram of the third version of the ADC in the file fp06rgt.

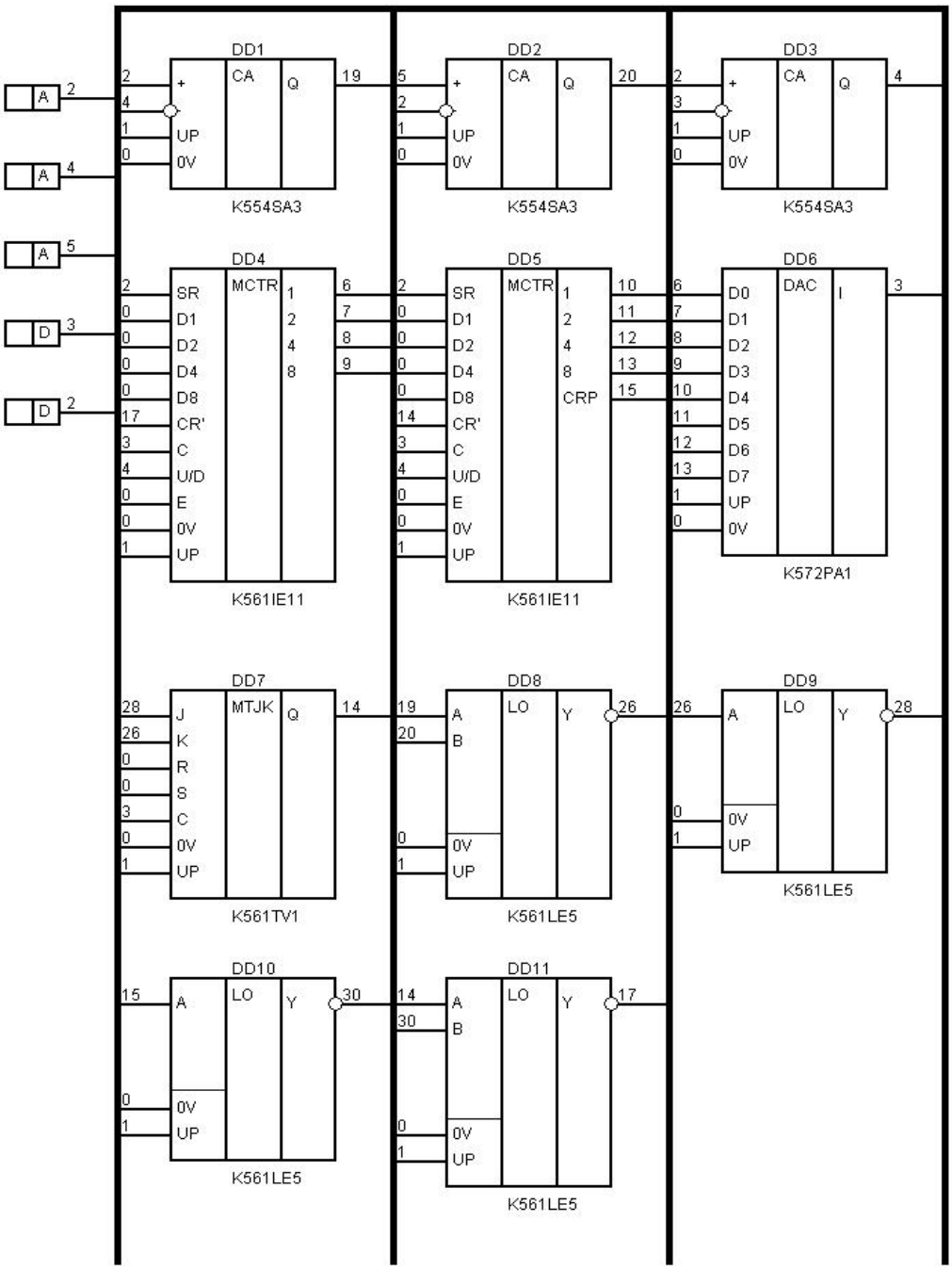

Figure 4.14. Schematic diagram of the fp06rgt rough-precision ADC, automatically built in a network viewer (Internet Explorer, Mozilla)

Below is the file fp06rgt.htm to display the first version of the circuit diagram:

```
<html> <body>
<applet code=CODXML.SCHView.class
 archive=..\..\CODOS\JAVA\CODXML.jar width=1000 height=2000>
<param name=xmlname value="D:\FA\CPP\fp06rgt.xml">
</applet> </body> </html>
```

The xml fragment of the fp06rgt.xml file for creating a schematic diagram of the first variant of the rough-accurate ADC fp06rgt.cpp is presented below:

```
<!--DOCTYPE root SYSTEM
"D:\CODOS\BIN\PARSER.DTD"-->
<s_schema>
 <s_title>Schema fp06rgt</s_title>
 <s_components>
  <s_component
   ElmNamR="K554CA3"
   ElmNamL="K554SA3"
   ElmNamA="LM111"
   DOCNameR="БК0.348.279ТУ1"
   DOCNameL="BK0.348.279TU1"
   DOCNameA="NONE"
   P0="120.000000"
   P01="0.000100"
   T01="100.000000"
   PNOM="150.000000"
   S="162.000000"
   Massa="1.000000"
   TPmin="-10.000000"
   TPmax="70.000000"
   Rt="41.200001"
   Tau="0.000000"
   PriceRel="4.000000"
   TYPEElm="CA"
   FunName="CA"
   FunCode="49"
   KElmPack="1"
   TYPEPack="201.14-1"
   NumTYPE="0"
   NumPack="1"    >
   <s_outputs KPinOut="2">
    <s_pin
     NElmPack="0"
     PinLeq="0"
     PinN="2"
     PinPCAD="Q"
     PinCod="NOUTP"
     NetNum="19"
     NetName="ND19"
     NetTYPE="ND"       />
   </s_outputs>
   <s_inputs KPinIn="6">
    <s_pin
     NElmPack="0"
     PinLeq="0"
     PinN="3"
     PinPCAD="+"
     PinCod="NINP"
     NetNum="2"
     NetName="NA2"
     NetTYPE="NA"       />
    <s_pin
     NElmPack="0"
     PinLeq="0"
     PinN="4"
     PinPCAD="-"
     PinCod="NINN"
     NetNum="4"
     NetName="NA4"
     NetTYPE="NA"       />
    <s_pin
     NElmPack="1"
     PinLeq="0"
     PinN="10"
     PinPCAD="UP"
     PinCod="UP"
     NetNum="1"
     NetName="ND1"
     NetTYPE="ND"       />
    <s_pin
     NElmPack="1"
     PinLeq="0"
     PinN="14"
     PinPCAD="0V"
     PinCod="GND"
     NetNum="0"
     NetName="ND0"
     NetTYPE="ND"       />
   </s_inputs>
  </s_component>
```

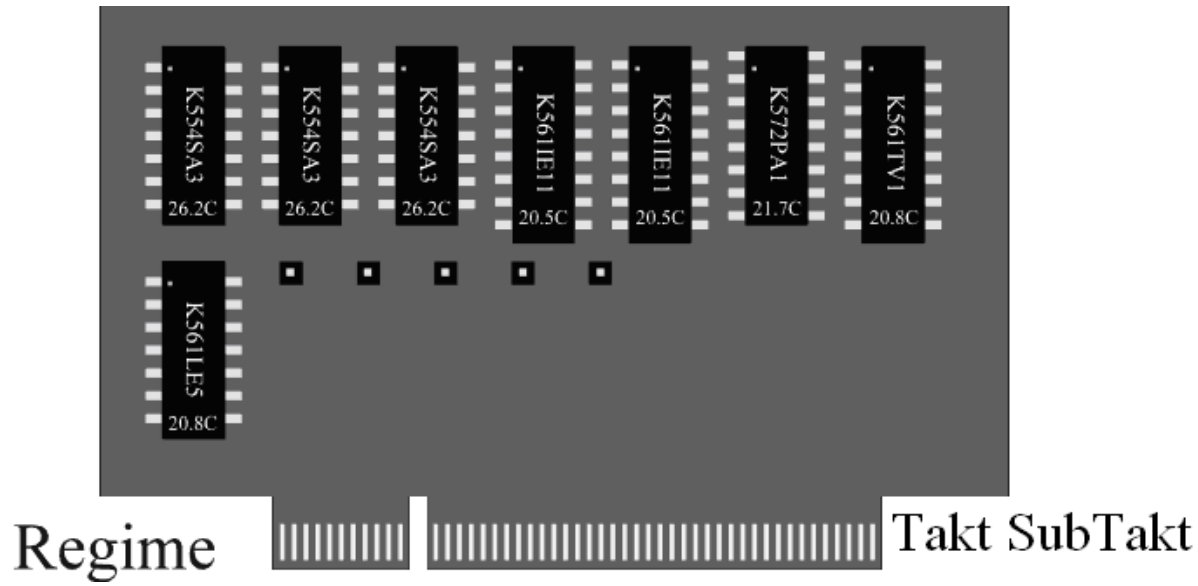

Fig. 4.15. Three-dimensional model of the ADC module FP06RGT with temperature display of components without changing digital signals

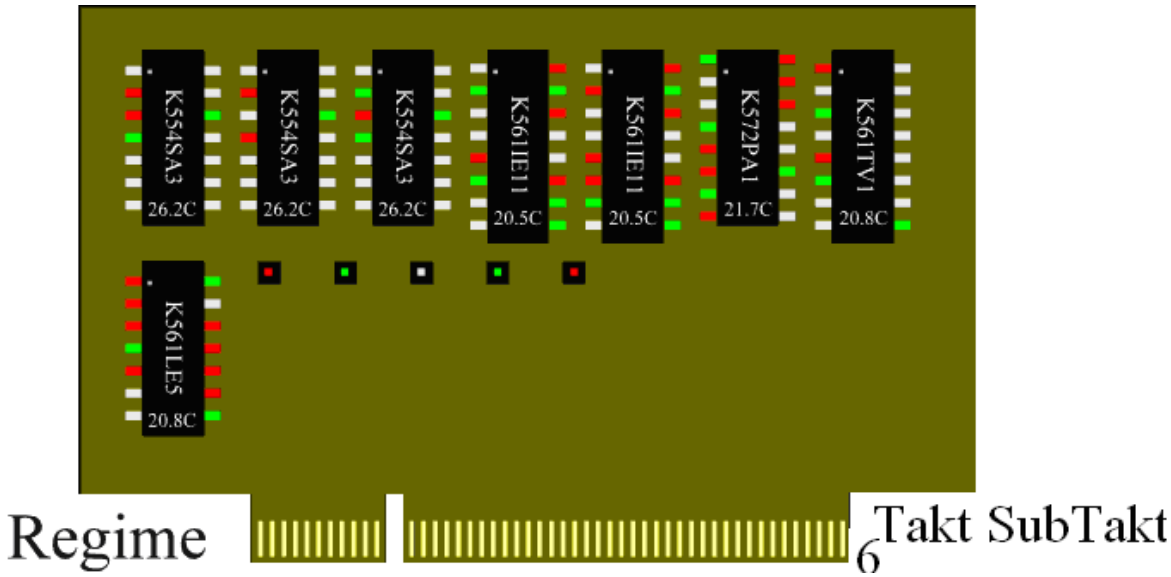

Fig. 4.16. Three-dimensional model of the ADC module FP06RGT with the temperature display of components with a change in digital signals

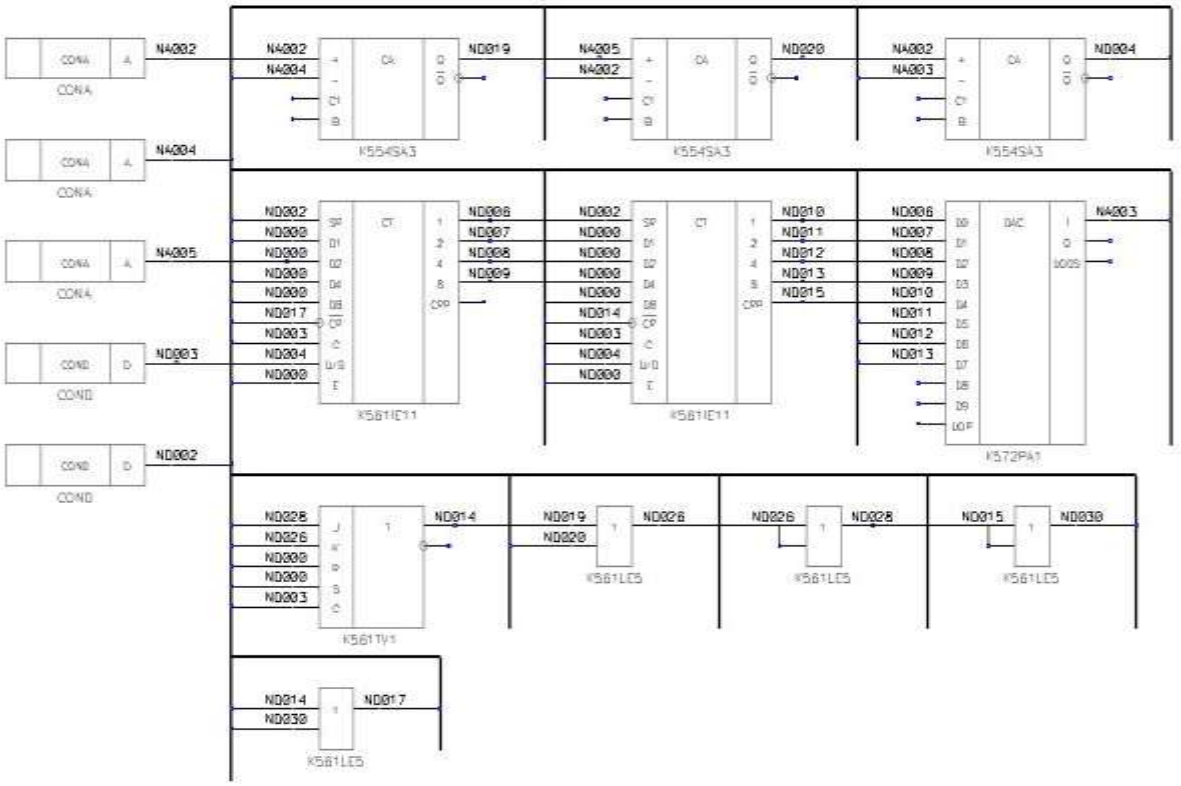

Fig. 4.17. Schematic diagram of a rough-precision ADC assembly using the table interface in PCAD8.5 from the fp06rgt file

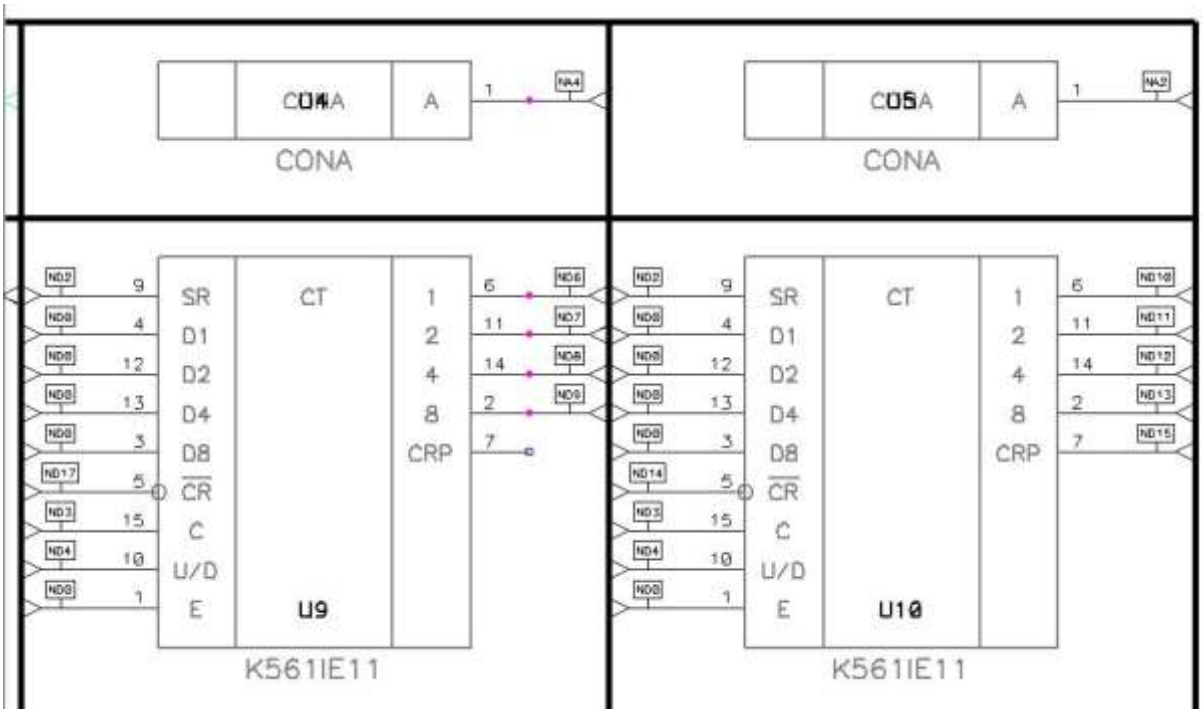

Fig. 4.18. Fragment of a schematic diagram of a rough-accurate ADC assembly obtained using a macro file (fp06rgt.mac) in PCAD2004

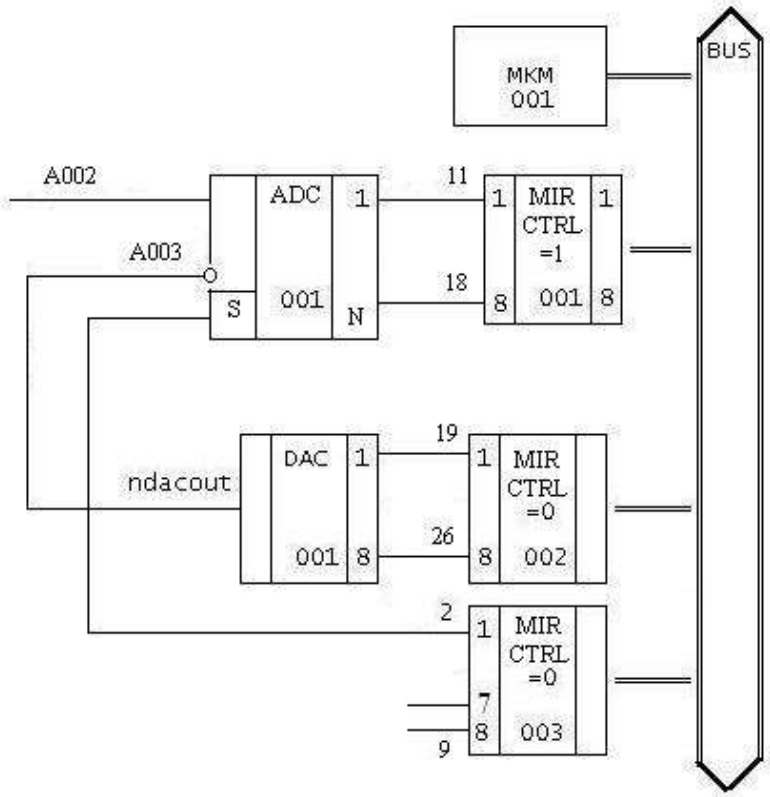

Fig. 4.19. Microcomputer-based computing system diagram with input of deviations from the intended signal using one-digit ADC

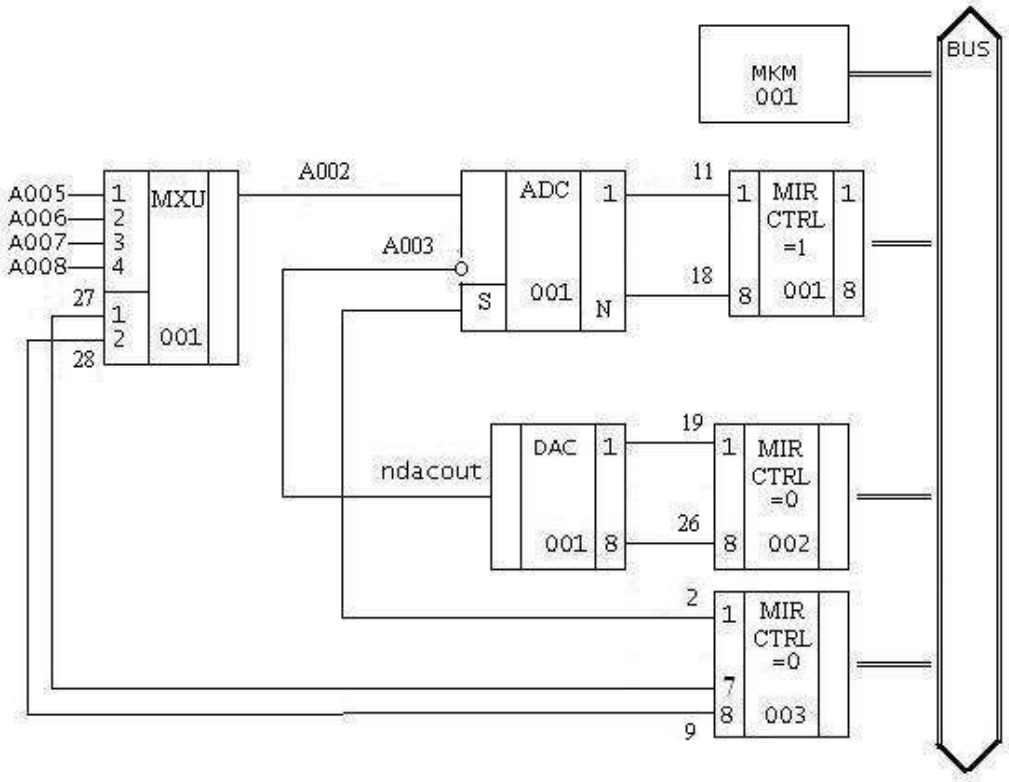

Fig. 4.20. Circuit diagram of a circuit-switched system based on a microcomputer with input of deviations from the intended signal using an ADC of one reference

Systems based on microcomputers allow you to abandon digital components and reduce the number of switching digital-to-analog components. In fig. 2.8 shows a diagram of a subsystem based on a microcomputer with interval prediction and estimation of signal deviations, and the corresponding formalized task in the file fp08adcp.cpp.

```
#include <uicpp.h>
void main () {// file fp08adcp.cpp
   BitString b01("01");
   int ninc = 6, sign=2, dacout=3, noutca=4, noutpca=15,noutmca=16,
   pdu=4, mdu=5;   float du=0.75;
   IntVector noutct(8, 7,8,9,10,11,12,13,14);
   IntVector mirtmp(8, 4,noutpca,noutmca,0,0,0,0,0);
  #include <txtpar>
INIT: / * Initial installation * /
   nvarmax = 1;  lres = -5;    ns = 40;    ntmax = 1800; nrg = 4;   lrg = 8;
   deltn = 1000; deltp = 9000; //ns
  #include <txtdcl>
INPUT: / * External influences * /
   Signal(ninc,b01,ntmin,ntmax);
   SignalA(2,cos(xt/50)+7,ntmin,ntmax);
   SignalA(pdu,as[3].ANex+du,ntmin,ntmax);
   SignalA(mdu,as[3].ANex-du,ntmin,ntmax);
UNIT: / * Description of the device * /
   MIR ("MIR", 1, 1, ninc, mirtmp);
   MIR ("MIR", 2, 0, ninc, noutct);
   CA  ("K554SA3",  3,  1,  noutca, 5,  sign,  dacout);
   CA  ("K554SA3",  4,  1,  noutpca, 5,  sign,  pdu);
   CA  ("K554SA3",  5,  1,  noutmca,  5, mdu,  sign);
   DAC  ("K572PA1",  6,  dacout,  noutct);
   MKM ("M686", 7);
   APrint(2, 20, ntmin, ntmax, 2, 5);
 #include <w>
}
```

The operation diagrams are similar to those shown in fig. 4.12–4.13.

An equivalent circuit using an ADC of one sample with a small number of bits for estimating deviations from the expected signal for one channel is shown in fig. 4.19 and for many analog signals in fig. 4.20. The main attention should be paid to systems with a forecast of the interval of signals or with a model of an object. With the accumulation of knowledge about signals and objects during the life cycle of systems, the flow of input information decreases and the flow of predictive estimates of signals increases.

Conclusion

A modular multi-level program-methodical complex of multivariate modeling and design of heterogeneous computing systems is proposed. New is the unified representation of a variant of the object, which can significantly reduce the number of modules and the amount of program code. Complex processing of formalized tasks requires significant energy costs, therefore, it is recommended that the processing is localized with visualization at the workplace.

Models and implementation algorithms for modular interfaces with circuitry and design engineering systems have been created. A single representation of a variant of an object is transformed by a modular complex into representations perceived by a wide variety of specialized and complex systems of circuitry and design engineering. Tabular, command and text versions of interface models have been created. The table interface is distinguished by the highest speed, and the command is clear. Of particular note is the automatic creation of formalized tasks in various syntactic environments.

5. AUTOMATION OF DESIGN OF INFORMATION MODELS OF OBJECTS AND COMPUTER SYSTEMS

5.1. Information models of objects in CAD

The main approach is the one-time input of the objects in the form of formalized tasks for various level analysis and end-to-stage. Typical graphic documents for the engineer in the form of time diagrams, circuit diagrams, and block diagrams must be obtained automatically as a result of the interpretation of the data. At the same time, the labor productivity of students and engineers increases and active training in the project team becomes real. The main focus of specialists is on decision-making in the automated evaluation of options.

Symbolic models of systems that contain essential information about functions or structure occupy an important place in the learning and design processes. The information model is described in GOST 34.003–90 and then repeatedly specified. The interpretation of a model or its changes over time depends on the subject area and facilitates perception [1, 2, 3, 6].

Symbolic models of systems can exist in the environment of an object that provides a comfortableconditions, energy and information transmission network. An object may appear to be a complex (interdisciplinary) information model or a set of embedded complex information models.

At the first stage, a measurement system is selected and floor levels are determined, all necessary components are created that are not in the libraries. At the second design stage, an object model is developed. There are three degrees of elaboration of the model.

1. Building a general plan. At this stage, the model is presented as a system of boxes, but does not include a detailed study of each individual object.

2. Each individual system object is worked out in more detail, the layout of rooms, floors, stairs is created, the location of doors and windows is set.

3. Modeling of premises. At this stage, several types of premises are created that have the same layout in the building and are often repeated on the plan.

An object can be created in a virtual reality environment or X3D without restrictions on the type of object. The computing network can be located in a building or in an open space. For stationary objects, such as buildings, Autodesk REVIT is suitable with the ability to automatically place network workstations in rooms in accordance with the standards. REVIT allows the user to create additional tools that must be described in the revit.ini file in the ExternalCommands section. The revit.ini file is loaded once when REVIT is called. Additional commands are located in the << Tools >> menu. For example, the additional Space Naming Utility command assigns the names and numbers of rooms in the MEP space.

All components in REVIT are in libraries in the form of families (Famili) of the class of parameterized objects. Specific objects have parameter values; their files have the rfa extension. Typically, component libraries are freely available. Examples of families are doors, windows, workstations with computers and an

armchair. It is better to present the recommended components for the project in the form of a table with the appearance, name in Russian and English, and dimensions in cm. The analog digital subsystem with sensors (ACVS) is located closer to the signal sources, and transmission and processing is done only in digital form.

For example, the optimal placement of access points in a building can be done only after evaluating the absorbing capacity of all structures. The project in REVIT has the rvt extension.

Many walls of MWall consist of objects for various purposes: Exterior, Interior, Retaining, Foundation and Curtain wall. Exterior Walls are load-bearing and must have thermal resistance in accordance with a set of rules and regulations. Interior walls (Interior Walls) are used as the internal division of the volume of the object and have a non-bearing character. Retaining Walls must hold the ground and form relief. Foundation Walls form the foundation of the building. Curtain Walls consist of panels. On the SFU2 campus, most buildings have load-bearing walls, and only Building B of the Radio Engineering Department is built using industrial panels. Building design begins with creating a grid (Grid) and determining the levels (Level). The grid of axes is distributed in the plan (Floor plan), and the levels from the facade (Elevation).

The university campus consists of many buildings, roads and underground utilities and is a higher-level facility with Autodesk INFRAWORKS. The university campus is part of the city facility. The process of designing models starts from the lower level and ends with the upper one, but iterates until the requirements of the task are met. It is possible to design many options for the object to choose the best.

The international standard ISO 15926 has established a three-tier model architecture. External (External) level of the models corresponds to user requests, internal (Internal) models correspond to the table of the UI CAD <COD> variant. The conceptual level in the software and methodological complex is represented by a formalized design assignment (FT). The basic concepts of the standard GOST R ISO 15926 are given in table 5.1.

In accordance with the standard, an analysis of the design process of various objects in different areas was carried out. The main results are presented in table 5.2. When designing devices for computing systems, variants are modeled and the optimal one is selected according to the efficiency criterion [1, 2, 3, 4]. For terrestrial systems, this is the cost per unit of performance. When designing the design of the printed circuit board, the components are placed according to the criterion of the minimum sum of the lengths of the conductors, and when tracing the connections, the penalty for vias and the possible level of interference are taken into account. A real device circuit is modeled under certain external influences and compliance with the task is checked.

Building design begins with an analysis of the site and possible external influences. In accordance with the main function and performance criteria, select the area and number of storeys of the building are given. The design of the building is being designed taking into account energy efficiency. The main

equipment of the building is located in accordance with the standards. Then the connection tracing for the entire system and its modeling are performed.

Information model of the object (BIM – Building Information Model) can be built in Autodesk REVIT using commands in the absence or presence of floor plans. The best results are obtained with the automatic construction of objects using software modules. The scheme of the algorithm for automatic construction of objects, in the presence of floor plans in the form of dwg files, is shown in fig. 5.1. For software modules in CAD, NetScriptCAD is used [3, 4, 5, 6]. The placement of equipment is divided according to the purpose of power supply, communication, technological, life support and building management [4, 5, 6]. Software modules allow you to place equipment, in accordance with the standards, places in lecture halls, workplaces in laboratories and offices. The time-consuming process is the connection of equipment.

The project directory must contain a subdirectory dwg for floor plans; rvt directory for Autodesk REVIT files for relative addressing capabilities. Software modules are best created for reuse by passing parameters. An example is the floor plan loading module.

CAD Autodesk REVIT involves three different professionals: architects, builders, builders, and engineers Systems Engineers (SYSTEM). Engineering systems specialists must have basic knowledge in various fields and programming fundamentals for the effective use of complex CAD systems like REVIT and building information model (BIM) during its life cycle. The state of equipment, facilities and structures in accordance with a single facility management protocol, such as BACNET, is sent to the campus dispatcher for operational management and prediction of the behavior of facilities. Information networks should be built on various principles to maintain operability in emergency situations.

Traditional is the industry training of specialists in narrow subject areas. Need correctthe curriculum for training specialists in engineering systems, taking into account information support by integrated CAD and related network services for stationary and mobile computing devices [4, 5, 6]. Table 5.1 and 5.2 confirm the general approach for systems of various levels.

Table 5.1

ISO 15926 Concepts

Unique Title	Description	Note	Superclass	I ISO 15926-2 entity
Auxiliary system (help = body system)	A system that is part of a large system and performs the functions necessary for the operation of a larger system, but which are not part of the functions of a large system		System	class_of_inanimate_ physical_object

Continuation of the table 5.1

Component (component)	A physical object that is used as part of a large object		physical object class_of_inanimate_physical_object	
Connection (communication)	A physical object that is intended to be connected		physical object	class_of_inanimate_physical_object
Connector (connector)	Two-part compound	Plug and socket	connection class_of_inanimate_physical_object	Connector ((connector)
End (end)	Function that is the end of the event		feature	class_of_feature
Plant (an object)	Physical objects: land, buildings, machines, tools used in trade and industry		physical object class_of_inanimate_physical_object	
Plug (cork)	A physical object that allows hermetically closing a hole		physical object	class_of_inanimate_physical_object
Section (part of the object)	A physical object that is one of the main parts of a large object		physical object	class_of_inanimate_physical_object
Start point (starting point)	Function that provides the start of an event		feature	class_of_feature
Support (support)	A physical object that is able to support some other things from a fall, dive, or defect		physical object	class_of_inanimate_physical_object
System (system)	A functional object, which is a set of functional objects that form a diagram for the provision of services or employees to perform one task	Examples: heating system, highway system, data processing systems.	functional physical object	class_of_inanimate_physical_object

Interpretation of system models and simulation results is performed by a specialist in the domain directly or after software processing. Processing the results allows to reduce the time of data perception by specialists in different subject areas.

5.2. Designing information models of objects at various levels of the hierarchy

The basic concepts in the field of information technology and automated systems are given in GOST 34.003–90. The concept of functional components and the corresponding symbols on the circuit diagrams are defined in the ESKD standard GOST 2.743–91, and the concept of "product model" in GOST 2.052–2006. The relations between functional components (TREL) comprise a computational module, which can be considered as part of a formalized task (FT, FZ), a module description section (UNIT), and presented (FVIEW) as a circuit diagram or a volume information model (BIM).

The use of ontologies in the design and maintenance of objects is recommended by international ISO 15926 and the Russian standard R ISO 15926, adopted in 2008. Due to the large number of specific components and terms, a multilevel model with a small number of basic concepts at the top level is used. As the working conditions of the object are clarified, the components are specified and their diversity increases. Multilevel presentation allows you to even out the complexity of decision-making at various levels.

An object's environment can be created in virtual reality, X3D or in a specific CAD environment [1, 2, 3, 4] without restrictions on the type of object. The computing network can be located in a building or in an open space. For stationary objects, such as buildings, Autodesk REVIT [3, 4, 5, 6] is suitable with the ability to automatically place network workstations in rooms in accordance with the standards. The university campus consists of many buildings, roads and underground utilities and is a higher-level facility with Autodesk INFRAWORKS. The university campus is part of the city facility [3, 4, 5, 6]. For mobile objects, CAD CATIA [1, 2, 3, 4] is used. Designing systems on a chip can be done in a free environment. FSF (Free Software Foundation) CAD Electric v.9, written in Java, or in industrial systems. Bipolar or field devices can be used. Designed and printed circuit boards. The process of designing models starts from the lower level and ends with the upper one, but is iterative until the requirements of the task are satisfied. The minimum volume of the description of the object is presented in symbolic form, and the maximum – in graphic form. Therefore, the input of symbolic descriptions and the automatic conversion of descriptions into forms convenient for human perception are distinguished by the minimum complexity.

The logical level is characterized by abstractions of the spatial distribution of components and their physical implementation. However, when coordinating a computing system (CS) with external channels, it is necessary to specify the physical implementation of the components. At the physical level, the state change energy for the binary component is estimated – the switching energy,

which determines the power dissipated by the object for a particular frequency and limits the degree of integration of the components [1, 2, 3, 4]. At the spatial level, the placement of elements tends to the maximum volumetric or surface density of the components, limited by technological standards and allowable power dissipation. Spatial placement is taken into account only in the design of objects with stationary components [5]. The level of the heterogeneous physical environment provides stable processing of information under possible external influences.

At each level of abstraction, functional models of systems as a whole are considered, which are called macro models and are described by the functions of outputs and transitions [3]. At the next level of abstraction are structural models that reflect the internal structure of objects and components.

Such models will be called micromodels. The results of using macromodels are always repeated. With the use of micromodels, it is possible to simulate the normal and malfunctioning system. Analysis of the discrepancy between the expected and actual results allows us to make a decision on the modification of the rules for the synthesis of a class of objects. Synthesis rules are represented by a decision table and can be described in a formalized task using conditional operators or selection conditions. For structural optimization, performance criteria are needed [1, 2, 3, 4]. For stationary computing systems, the criterion of efficiency is the cost of a unit of performance. For mobile objects, such a criterion may be the mass of a unit of productivity or the mass of a computing system and an energy source per unit of productivity [3, 4, 5].

Table 5.2

Conformity of operations when designing objects of various levels

Operation	Printed circuit boards	Building	Infrastructure	Organization of staff for collaboration
Conceptual modeling	Conceptual modeling and performance criteria	Conceptual modeling and performance criteria	Conceptual modeling and performance criteria	Conceptual modeling and performance criteria
Area limitation	Circuit board	Plot plan	Site Plan (Campus)	Job placement plan
Setting Layers	Layer definition	Floor Assignment	Communication layers	Messaging levels
Placement of objects	Component Installation	Equipment installation	Installation of buildings and facilities	Staff placement
Tracing connections	Connection tracing	System trace	Tracing of aboveground and underground communications	Messaging Rules

Continuation of the table 5.2

Modeling the system with an assessment of performance criteria	Modeling a real scheme with an assessment of performance criteria	Modeling the system with an assessment of performance criteria	Modeling the operation of the complex (campus) with an assessment of performance criteria	Modeling the work of the team in various conditions with an assessment of performance criteria

5.3. Creating an Object in CAD REVIT

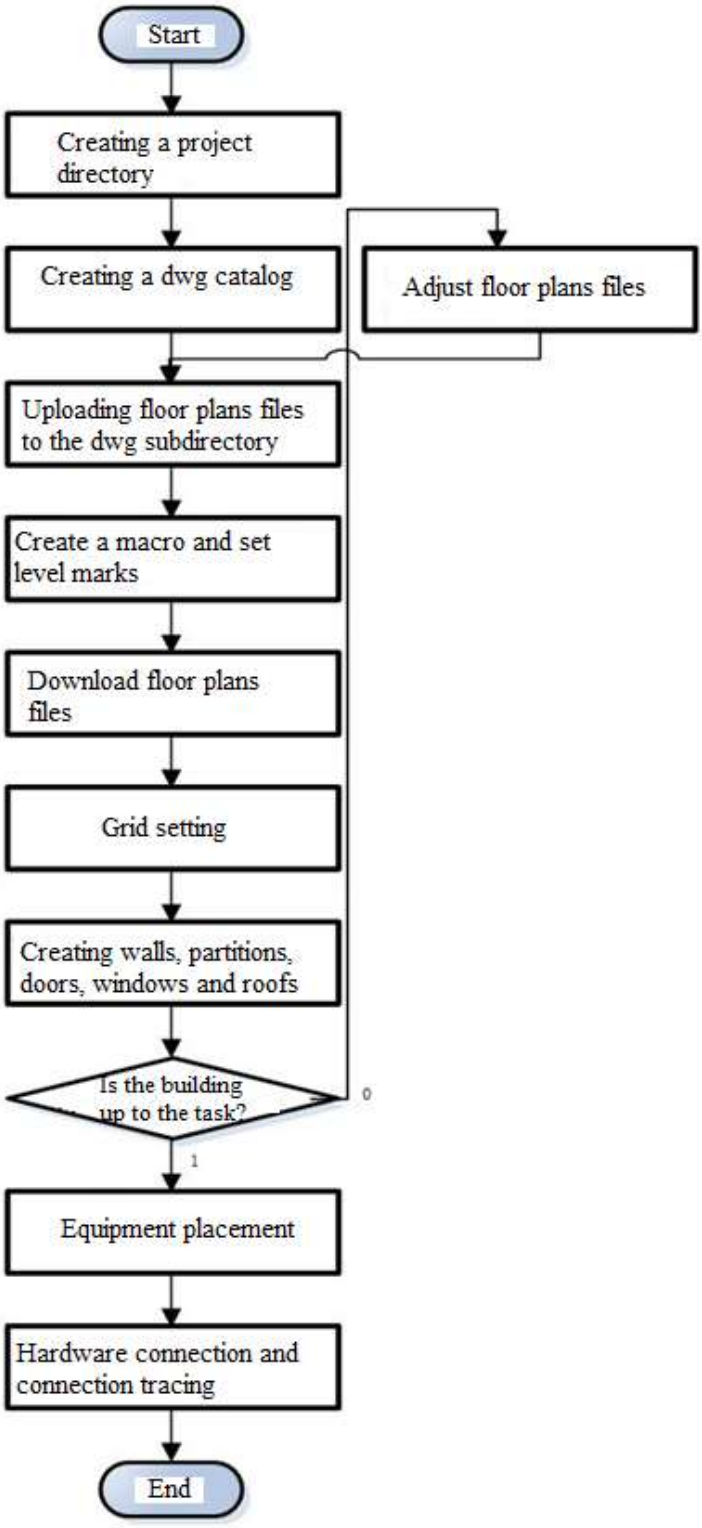

Fig. 5.1. The scheme of the algorithm for automatic construction of objects

DWG (drawing) – a binary file format used to store two-dimensional (2D) and three-dimensional (3D) project data and metadata. It is the main format for some CAD systems: AutoCAD, nanoCAD, IntelliCAD. DWG format is supported by import-export functions. Formats.dws ("drawing standards" –

drawing standards),.dwt (“drawing template”) – are also a DWG format. The project used 5 dwg files SFU2H8L1-5. In fig. 5.2 is a dwg file of the second floor of the SFU2H8L2.

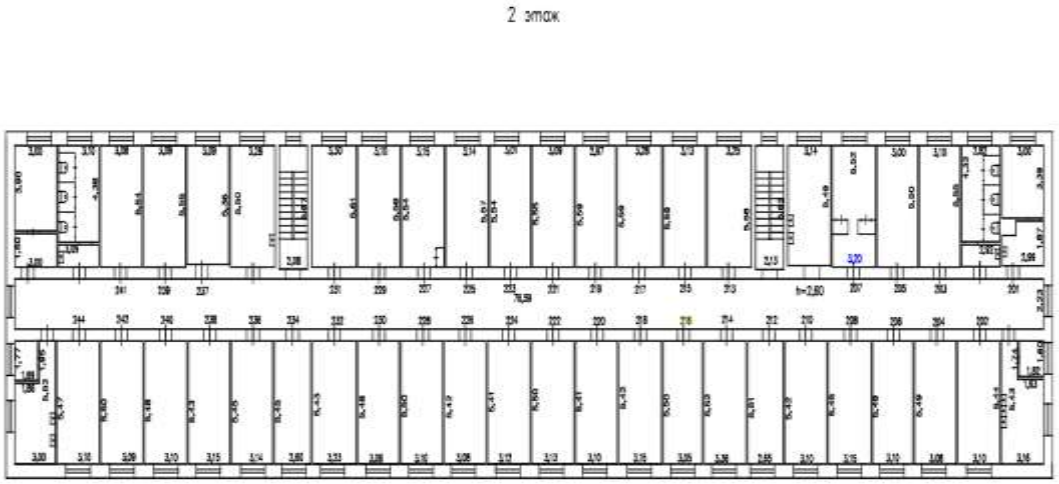

Fig. 5.2. Dwg-file with the plan of the second floor of the hostel 8 SFU2

Fig. 5.3. View of the real hostel SFU2H8

```
using System; // The program of floor loading files.
using System.Collections.Generic;
using System.Linq;
using Autodesk.Revit.DB;
using Autodesk.Revit.DB.Architecture;
using Autodesk.Revit.UI;
using Autodesk.Revit.UI.Selection;
using Autodesk.Revit.ApplicationServices;
using Autodesk.Revit.Attributes;
using Autodesk.Revit.Extension;
public class LevelCreation

  {
```

```
            public static List<Level> CreateLevels(RvtDocument doc, double[]
elevations, string[] names = null, string[] pathes = null, )
            {
               List<Level> list = new List<Level>();
               for (int i = 0; i < elevations.Length; i++)
               {
                  double elevation = elevations[i];
                  Level level = doc.Create.NewLevel(elevation);
                  if (names != null && names.Length >= i + 1)
                     level.Name = names[i];
        if (pathes != null && pathes.Length >= i + 1)
        {
                     level.Import_path = pathes[i];
                 level.SAPRImport(pathes[i], NULL, NULL);
        }
                  list.Add(level);
               }
               return list;
            }
        }
        CachedDoc.Delete(new FilteredElementCollector(CachedDoc)
                     .WherePasses(new ElementClassFilter(typeof(Level), false))
                     .Select(e => e.Id)
                     .ToList());
        List<Level> levels = LevelCreation.CreateLevels(
            CachedDoc,
        // In quotation marks indicate the path to the file
        StreamReader objReader = new StreamReader("path");
            string sLine="";
            ArrayList arrText = new ArrayList();
            while (sLine != null)
        {
           sLine = objReader.ReadLine();
           if (sLine != null)
                arrText.Add(sLine);
        },
        // Loop a lot of dwg files from the dwg subdirectory
        new string[]
            {
              for (int i = 1; i < 6; i++)
              {
        " \dwg\L" + i + ".dwg" ;
              }
            }    );
```

Errors in the original dwg files should be noted, but after corrections from the SFU2H8.rvt information model, you can export files without errors. Such estimates for many solutions represent a time-consuming task, the real solution of which requires tools and information support.

All SFU2 object models were built in a similar way, files with centers of coordinates of objects in the LL84 system were created. To build roads, you need a file with the centers of coordinates of the break points and a file of break points for the communications of engineering networks of information, electricity, water supply and sewage.

5.4. Campus Design at INFRAWORKS

For the design of a campus or complex of buildings, an analysis of the site and possible external action. The design environment can be Autodesk INFRAWORKS CAD. The boundaries of the site and objects are set in angular coordinates, which makes it possible to place the campus on another planet. Buildings and campus facilities can be placed in two versions. In the form of symbols in the form of boxes or in the form of full-fledged buildings created in CAD REVIT. Placing symbols requires less resources and is recommended for checking the coordinates and relative position of objects. After checking the coordinates, you can place full-fledged buildings from files of the rvt type. For the campus project, you need to create the project directory and the dwg and rvt subdirectories in it, where all the necessary objects are located and relative addressing is possible. All SFU2 object models were built in a similar way, files with centers of coordinates of objects in the LL84 system were created.

In CAD Autodesk INFRAWORKS designing is possible using commands or software modules. Team mode is recommended for the original differences of one campus from another. Using software modules is recommended for iteratively designing a campus or creating similar campuses.

We give for measures of the function of adding CAD objects REVIT to the model of the second campus of the Campus of the Siberian Federal University in CAD Autodesk INFRAWORKS and its use function AddRvtModel (name)

```
{
        var model = buildingsCoords[name];

        var table = app.ActiveModelDb.Table('BUILDINGS');
        table.BeginWriteBatch();
        var w = table.GetWriteRow();

        w.NAME = model.name;
        w.DATA_PATH = "..//rvt" + model.name + ".rvt"
        w.DATA_SOURCE_ID = -1;
        w.MODEL_SCALE_X = model.sx;
        w.MODEL_SCALE_Y = model.sy;
```

```
        w.MODEL_SCALE_Z = 1;
        w.MODEL_ROTATE_X = 0
        w.MODEL_ROTATE_Y = 0;
        w.MODEL_ROTATE_Z = 0;
        w.ROOF_HEIGHT = model.sz * floorHeight;
        w.ROOF_MATERIAL = roofMaterial;

        var geom = new adsk.Geometry({
                "type": "Polygon",
                "coordinates": [[
                        model.x, model.y
                ]]
        });
        w.GEOMETRY = geom;
        table.Insert(w);
        table.CommitWriteBatch();
};
// Add building models
AddRvtModel('SFU2G2014full');
AddRvtModel('SFU2J2015');
AddRvtModel('SFU2H82015');
```

The campus combines roads of various classes, which are set at break points. The road class can be selected and changed using the menu. Campus communications are conducted at different depths and are displayed differently flowers. Communications are best added after the placement of buildings.

The coordinates of all objects are made separately files are three arrays: buildings Coords-coordinates of buildings, roads Coords-coordinates of roads, trees Poligons Coords-coordinates of a polygon with trees. This allows you to use the coordinates in any project, including the corresponding javascript file. Multivariate design with change of coordinate files is possible.

Information models of the campus and buildings are important not only for the design of buildings and their complexes. Much more important is the model at all stages of the life cycle, taking into account repair and changes in communications. After deformations were detected in the northern and southern wings of the main building, students, under the guidance of the author, developed a project for monitoring deformations of the campus of SFU2 campus based on fiber-optic sensors with Bragg gratings with centralized processing of results for decision making. The project with the cost of equipment was transferred to the vice-rector for the development of SibFU in 2014.

Computing systems should be designed with due regard to purpose and environmental conditions. If there are sample modules, the student can complete the project or create a similar module during the semester.

The ways of development of the program-methodological complex, which allows for multivariate design and maintenance of the object, with automatic

assessment of parameters and performance criteria and automatic conversion of results to CAD formats or CAD files of design and design at various levels of the hierarchy, are investigated. A general approach to design automation for objects at different levels of the hierarchy is proposed despite significant differences in technologies. The iterative nature of the design and the accumulation of knowledge about the object activates the learning process.

6. LOCAL EXECUTION AND NETWORK SERVICES FOR MODELING AND DESIGNING HETEROGENEOUS COMPUTING SYSTEMS

6.1. Software and hardware systems CAD

CAD is designed to synthesize descriptions of many technical analog-to-digital computing devices and systems. Descriptions are possible at various levels of abstraction. Descriptions of one or more options may be provided to the design engineering systems.

Scope CAD design training for students and engineers in the design of computing devices and computer systems based on microcomputers from an early stage. UI CAD works on computers with operating systems Windows, Linux, OS2 and ZVM from personal to powerful virtual machines with the operating system. The client and server modes are provided, which allows using the complex for local and distance learning. Network services are available for Android mobile devices. On the server are: a software package, teaching materials, study guides in Russian and English.

Features:

1) the ability to describe computing systems at various levels of abstraction;
2) the ability to describe many technical solutions;
3) unification of hardware descriptions in high-level languages in various syntactic environments;
4) automatic comparison of estimated and actual results;
5) automatic resource assessment;
6) interface with design engineering systems;
7) automation of the formation of symbol libraries and component constructs;
8) alternative routes for performing formalized tasks to enable comparison of results. The difference in routes may be in the description language, operating system or components.

Main functions: synthesis of descriptions of many technical solutions; automatic multivariate analysis of the operation of computing devices with an assessment of resources and performance criteria; automatic comparison of estimated and actual results; import and export of descriptions in specific CAD formats.

System composition: libraries of components, nodes, devices and systems; software and methodological complex, including a dialogue monitor and various functional modules, manuals with the stamp of the Ministry of Education [28–33].

Requirements for hardware and software depend on the version of training and research CAD. The first version of the UI CAD Code (CODVI) is intended for interactive systems of collective use on IBM 360 in the environment of OS/360 and compatible with them technical (EC computers) and software.

The second version of the UI CAD Code (CODV2) operates in a shared system on IBM S370 – S390 computers and new IBM PC ServerS390/Z personal servers with local or remote terminals or personal computers with a terminal software emulation module. Using a personal computer provides a graphical output of the results. The environment is a virtual machine system with a dialog processing subsystem. The software is available to all users for reading at the same time and is carried out from two mini-disks. The number of users is limited only by the number of terminals. The results are displayed on a PC and network stations in graphical form, and on alphanumeric terminals in symbolic form.

The third version of the UI CAD Code (CODV3) is intended for personal IBM-compatible machines in the environment of MS DOS, PC DOS, or OS/2 DOS session. When using translators from Borland C, C++ version 3.1, PCAD v8, 4 MB of RAM is sufficient, but 8 MB is recommended. Using a local area network with a file server reduces disk space requirements. The version is characterized by an automatic interface with industrial CAD systems PCAD, GRIF4, ORCAD, CADDy, Eagle and PRAM5.3. Local and remote execution of formalized tasks in the computer network is provided, which expands the capabilities of the workplace where the FT synthesis and viewing the results are performed. It is possible to execute the FT on servers that are not feasible on the workstation itself. An example is formalized tasks in PL/1, JAVA and complex tasks in C++.

Version Four UI CAD Code (CODV4) is designed for 32-bit IBM PC machines for the multi-tasking OS/2 environment – Windows, Linux. The CAD UI shell for synthesis, local and remote execution of the Federal Law was created on the basis of IBM's Visual Age toolkits. Visual Age software packages are designed for the main universal programming languages PL/1, C++, JAVA and work on various platforms. The source text can be used to create software systems on various platforms. Directly, the UI CAD system is created on the basis of the multifunctional text editor LPEX (Live Parsing Extensible editor) from Visual Age packages. LPEX editor works on various platforms, can be launched independently or from a browser, for example Mozilla Firefox and Google Chrome. Version four UI CAD the code can work both on the client’s machine, so on the server. The server operating environment can be OS/2, Windows, Linux, VM / ESA S390, ZVM. The server operating environment can be expanded with systems for which Visual Age or Web Sphere tools are available.

The main focus is the transition to 64-bit products of the FSF – Free Software Foundation. The fifth version (CODV5) is designed for web servers in the local and distance education system and can be used to solve problems of 64-bit CAD systems with the support of information modeling standards GOST R 57563-2017/ISO/TS 12911:2012, GOST R 57310–2016.

The teaching material provides intensive training in the synthesis and implementation of formalized tasks. It is better to place the teaching material on the server of the Siberian Federal University (e.sfu-kras.ru) and the library of the Siberian Federal University (bik.sfu-kras.ru), and the tasks should be performed on the application server. The advantage of the fifth version is the execution of

formalized tasks on application servers in various Windows and Linux environments. At the same time, Linux uses FCF GCC, and for Windows FSF mingw-64 with support for the GCC and GNU ADA languages. Creating a menu as a browser extension allows you to fully use mobile devices.

6.2. Local execution of a formalized task on personal computers in Windows, Linux, and OS/2

In the environment of the multi-tasking OS/2 operating system [145], various versions of research CAD systems can work. Version 3 can be run as a DOS program with a shell for MSDOS or with a specially designed shell for OS/2 based on the REXX language. Version 4 of the CAD system COD (CODOS) is designed for the environment of OS/2 and Windows. Version 4 of CODOS consists of software modules, static and dynamic libraries, batch files, and a shell with a mode of regulated dialogue. The shell serves to carry out formalized tasks by various design procedures in local and remote versions with minimal time. As a result of the implementation of design procedures, design solutions are obtained in the environments of various CAD systems. One formalized task corresponds to one or many design decisions of CAD systems of a lower level. The main subdirectories of CODOS are: \ BIN, \ DOC, \ INCLUDE, \ INCPLI, \ JAVA, \ LIB, \ LIBWIN.

Automated system PCAD4.5–8.5 technical design is performed in DOS sessions, while ORCAD, CADDY, and PCAD2001–2006 are performed in Windows. In the OS/2 environment, you can use the Toolkit for OS/2, WorkFrame/2 tools, IBM C Set++ and IBM PL/1 translators, the REXX interpreter, tools for organizing a local network and solving problems in the client-server mode. However, it is better to use IBM's integrated Visual Age integrated tools for OS/2 and Windows and the C++, PL/1, and JAVA languages. The Visual Age complex includes the LPEX editor, on the basis of which it is convenient to build a shell for executing and editing formalized design tasks with various programming languages. At the same time, the LPEX editor allows you to detect syntax errors. In the environment of the LPEX text editor, parsers have been developed for the textual description of the circuit in the ALT formats of the PCAD file and CAD files PRAM 5.3. The shell allows, in a regulated dialogue mode, to perform a formalized task on local machines in an OS/2 or Windows server environment, as well as on IBM S390 servers in an operating environment of virtual machines (VM) or OS390 (MVS). The common language environments of IBM VM, MVS, Linux, and OS/2 make it easy to learn and move from one system to another.

The source files for the fourth version correspond to the COD directories for hosting CAD and X:\FA\CPP for placing test cases in C++. For PL/1, test cases are located in the X:\FA\PLI subdirectory, for ADA (VHDL), X:\FA\ADA. For the JAVA language, there is a subdirectory X:\FA\JAV with a three-character JAV file type and the ability to place the shared FA directory on disks with the FAT file system. To perform formalized tasks in the JAVA language under OS/2,

the file is copied to the Y:\FA\JAV subdirectory of drive Y with the HPFS file system, with the JAVA file type. The task is executed, and the results are sent to the main subdirectory X:\FA\JAV. In a Windows environment, a JAV file is copied to a JAVA file in the same subdirectory, followed by execution. After the job, the intermediate files are deleted. The CODOS directory contains the same subdirectories, like COD for MSDOS, an additional subdirectory is INCPLI, which contains INC type files. All files for selecting entry points to procedures are combined into one common WALL.INC file and are used for a large number of different types of components. Statically-linked libraries of type LIB, dynamically-linked libraries of type DLL and import libraries of type LIB are located in the LIB subdirectory for OS/2 and the LIBWIN subdirectory for Windows. The names of libraries and commands for performing formalized tasks are formed in accordance with table 4.4. dynamic link libraries such as DLLs and import libraries of type LIB are located in the LIB subdirectory for OS/2 and the LIBWIN subdirectory for Windows. The names of libraries and commands for performing formalized tasks are formed in accordance with table 4.4. dynamic link libraries such as DLLs and import libraries of type LIB are located in the LIB subdirectory for OS/2 and the LIBWIN subdirectory for Windows. The names of libraries and commands for performing formalized tasks are formed in accordance with table 4.4.

In a Windows environment, paths are included in environment variables or in AUTOEXEC.BAT:

SET PATH=X:\CODOS\BIN; X:\CODOS\LIBWIN; X:\PCAD\EXE;
SET INCLUDE=X:\CODOS\INCLUDE; X:\CODOS\INCPLI;
SET LIB=X:\CODOS\LIBWIN;
SET LPATH=X:\CODOS\BIN; (First recording)
SET PLILPATH4=X:\CODOS\BIN; (First recording)
SET CODLOC=X; (uppercase).

The analysis of the behavior of objects and the assessment of resources obtained in different environments using different tools, increases the reliability of the results.

Formalized tasks in the FSF C++ language are independent of the environment and type of translator (GCC and CSET2 from IBM), therefore, the formalized tasks in C++ version 3 COD and version 4 CODOS are the same, and you can use the general subdirectory X:\FA\CPP. The texts of formalized tasks in the PL/1 language (of the PLI type) for VM and OS/2 environments also coincide. Formalized PLI jobs can be performed autonomously on a workstation, on a server on a local network, or on a remote server. A formalized task with component models will be performed on S390 servers when the model is included in the main procedure. In operations of logical addition and coupling, instead of the vertical bar (|) symbol, an exclamation point (!) is used. In OS/2 and Windows environments, the operation symbol can be omitted.

Table 6.1

Table of results of formalized tasks

Subcommand Name	Appointment	Result Type	Presentation of the result
Command mode	Command Mode		
Modelling	Modeling	name.lsg, name.msg	Work chart
Interface PCAD 45	PCAD 45 Interface	Table	Binary version of the scheme
		Command	Batch file
Interface PCAD 85	PCAD 85 Interface	Table	Binary version of the scheme
		Command	Batch file
Interface ORCAD	ORCAD Interface	Command	Batch file
Interface PCAD 2001-2006	PCAD Interface 2001-2006	Command PCAD 2001-2006	PCAD batch file 2001-2006
Interface CATIA	CATIA Interface	Command CATIA	Project at CATIA
Interface VRML	VRML interface	name.wrl	3D Image
Interface X3D	X3D Interface	name.x3d	3D Image
Interface WebGL	WebGL Interface		3D Image
Format FT COD	Format FT UI CAD	FT: C++, ADA, JAVA, PL/1	Federal Law in C++, JAVA, PL/1
Format STEP (EXPRESS)	Format P ISO 10303	name.stp, name.exp	Project description
Format EDIF	EDIF format	name.edf	Project description
Format PRAM 53	PRAM format 53	On components	By component
		On nets	In chains
HTML format	HTML, XML format	name.htm, name.xml	Circuit diagram
Format CADDY	CADDY format		Circuit diagram
Format EAGLE	EAGLE format	name.xml	Circuit diagram
Format TSCH	Layout Option Format	name.tsh	Chart Variant Table
Format ALT	Constructive (ALT)	PCAD 45	PCAD 45
		PCAD 85	PCAD 85
		PCAD 2001-2006	PCAD 2001-2006

Working in a CAD UI environment for OS/2 and Windows is basically the same as working in an MSDOS environment. Local and remote execution of formalized tasks is provided on the workstation, on OS/2, Windows, VM S390 (VMz) and OS390 (zOS) servers. A graphical display of the results on the workstation is performed by the UISGO for OS/2 and UISGW for Windows programs. The display program is called up in the menu of Result in the item Graphics, but to perform the next task you need to exit it. Before starting work, you can call the display program and the editor, and then make the transition between the synthesis processes of the formalized task in the editor and the graphic display of the analysis results. Comparison and signal evaluation tables, resource estimates and performance criteria for all automated analysis options are displayed in message files of the MSG type.

New is the ability to display descriptions in electronic circuit language (SML) with display in any viewer that supports the XML language [28]. Simultaneously with viewing the circuit or module design, you can observe the simulation results.

Regulated dialogue mode can be built on the basis of various tools: multifunctional editors EPM, LPEX or web browsers, such as Firefox add-ons. Web browser add-ons are preferred for mobile devices. Multifunctional editors can be called from web browsers after setting the correspondence of application programs by file type. The synthesis of formalized tasks in high-level languages is best done in a multifunctional editor's environment with parsing of tasks in various languages. Formalized tasks are viewed in various editor windows, and using the drop-down menu, automated analysis and assessment of resources and performance criteria are organized. In the "simulation" mode, a window opens with the command line mode, where the broadcast results are displayed, editing and performing a formalized task. If the result level specified by the LRES parameter is zero or negative, the symbol information is displayed in the same window, and the time chart is displayed graphically in a separate window. Different information is displayed depending on the set result level. With a value of level 2, actual results are compared with the expected ones and measures with differences are displayed. You can view the circuit in the PCAD environment, the specific module of which is selected in the menu. The automatic UI CAD interface is also supported by the menu. Frequently used functions are duplicated by icons. Different information is displayed depending on the set result level. With a value of level 2, actual results are compared with the expected ones and measures with differences are displayed. You can view the circuit in the PCAD environment, the specific module of which is selected in the menu. Frequently used functions are duplicated by icons.

In the table 6.2 shows the basic commands of the EPM editor. It is possible to open multiple files for editing by specifying an asterisk (*) in the file name or type. You can open the file for editing even after the DIR command, moving the pointer to the desired file and pressing Alt + 1.

The EPM editor is configured using a special file (PROFFILE.ERX) with the PROFILE ON command. The mode should be saved with the SAVE command in the OPTIONS menu. In the editor environment, command line mode for OS/2 and MSDOS is possible. In fact, actions are performed using batch files in REXX. For example, an automated analysis of the behavior of an object, the structure of which is described by a formalized task in C++, is performed using the FCOF.CMD command file corresponding to the Visual Age C++ software package. An automated analysis of the object's behavior, the structure of which is described in a formalized PL/1 task, is performed using the FPOF.CMD batch file, and JAVA tasks are performed using the FJOF.CMD file. All batch files are located in the X:\CODOS\BIN directory. In the Visual Age complex, the utility for working with the ILIB.EXE libraries, with import libraries – IMPLIB.EXE. CAD UI includes static and dynamic libraries. Static and dynamic libraries for various applications are combined into a subdirectory X:\CODOS\LIB for the

OS/2 environment and X:\CODOS\BIN for the Windows environment. Despite all the advantages of the EPM editor, on the basis of which the complex shell was developed, preference was given to the multifunctional LPEX editor, which is part of the IBM Visual Age, Web Sphere and Firefox and Chrome browsers. Visual Age tools are available from IBM for various platforms and languages. Using the tools of Visual Age, you can have a single source code for various platforms.

Browser-based versions of menus for application servers are preferred when migrating to online educational technologies and online courses. But this became possible after the unification of menu additions for the latest versions of Firefox and Chrome browsers. The LPEX editor supports the main programming languages, has several profiling files and provides the interface of the EPM editors (table 6.2), Firefox (table 6.1). Jobs are launched from the menu if there is a command window. Character results are displayed in the editor window, and graphic results are displayed in a special window.

Table 6.2

Editor Commands EPM and LPEX (EPM Mode)

Functional key	Appointment
F1	Help
F2	Save current file
F3	Close the current file. Exit Editor
F4	Save and close the current file
F7	Delete current file
F8	Open a new file for editing
F9	Return to current line
F11 or Ctrl + N	Go to the previous open file
F12 or Ctrl + P	Go to the next open file
Home	Go to beginning of line
End	Go to end of line
Ctrl + Home	Go to beginning of file
Ctrl + End	Go to end of file
Ctrl + Backspace	Delete current row
Ctrl + E	Delete part of a line, from cursor to end
Alt + B	Select text fragment
Alt + C	Copy selection
Alt + D	Delete selection
Alt + =	Executing the current line as a command

In accordance with the methodological support of UI CAD, the menu of the LPEX editor or the Firefox browser is supplemented with the following items: Set parameters, Execute, Registration, Result, Remote Execute, Effection, Information.

Parameter setting item allows you to: select a remote server (Choice HOST), programming language (Language), description type for import (CADIN), interface type with CAD, drawing format (Format of frame), type of libraries (Library).

The Registration item allows you to complete Full registration or enter the name of the formalized task (Name of task). The name of the formalized task is entered without a file type (extension). Full registration allows you to specify the data necessary to fill out the drawing stamp.

Paragraph *Execute* allows the workstation to perform modeling (Modeling) or an interface with one of the industrial CAD systems. The interface can be a command (Command) or tabular (Table). A file is created in the command interface containing the sequence of commands needed to create the circuit. The tabular interface allows you to convert a formalized task into the internal structure of a circuit in a specific CAD system. The table and command interfaces are implemented for CAD PCAD. For CAD, ORCAD, PCAD200x, Eagle, Cadstar implemented command and CAD systems CADdy – software interfaces. Command interfaces are integrated into the common libraries LGOI and LGWI.

Paragraph *Result* allows you to view the analysis result in a graphical form (Graphics) or a message file in text form (View MSG file), get acquainted with the diagram (Table) or observe the execution of a command file (Command) in CAD. To view the results, in a specific CAD, you need to open the diagram format file and execute the macro file with the name of the formalized task.

Specialization of the editor is performed using the PROFILE.LX file, which is located in the X:\CODOS\BIN subdirectory, and files of the LX type. The path to them must be specified first in the SET LPATH statement for the LPEX editor of Visual Age or IBM Web Sphere. Several profiling files are used, of which PROFINIT.LX is loaded first, then an edited file, profiling the PROFSYS.LX file, and then PROFILE.LX. Features of the language are specified using files of the LXL type, whose names correspond to the file types CPP, JAVA (JAV), PLI, CMD of C++, REXX languages.

A feature of the PL/1 translator is the ability to set the correspondence between a file and a data set using the SET command. For example, the execution of a formalized task uses the SYSIN file for the input data type DAT, the DBT file – for the UIPCAD.DBT data set, the SYSOUT specification file – for the LST data set, the GOUT analysis file - for the LSG or OUTPUT.000 data set. Examples of SET commands for a batch file in REXX language, in which the name of the formalized task is assigned to the variable NAME, are given below:

```
'SET DD: SYSIN =' name'.dat '
'SET DD: DBT =' uipcad.dbt '
'SET DD: SYSOUT =' name'.lsp '
'SET DD: GOUT =' name.lsg '.
```

Similarly, you can associate file names and datasets for C++ using the Visual Age tools and translators for the VM/ESA and OS/390 operating systems. For compatibility with other translators for the C++ language, the getenv() environment variable read function is used. Dynamic linking of file names and data sets before the task execution step is important for compatibility of client and server versions of the CODOS software package. At the workstation, after

registration and settings, information about the name of the formalized task and the drawing frame are in the COD.INI file. On servers, tasks for execution arrive at arbitrary points in time, and parameters are transferred on the command line or by setting environment variables. Therefore, in C++, the environment variables CODLOC are checked for a disk symbol with the installed CODOS system, CODE for the name of the formalized task and FRAME for the name of the frame (A0 – A4) of type SCH. Examples of command execution for automated analysis of a formalized task are given below. Selecting the *Execute* item and the *Modeling* sub-item in the main menu, execute the fexec.lx editor command, which, depending on the operating system and the selected language, executes an OS/2 command file of the CMD type or a Windows command file of the BAT type in accordance with table 4.4. For example, for C++ in Windows, modeling is done with the fcwf.bat command, and for OS/2, the simulation is fcof.cmd. For JAVA, modeling in the OS/2 environment is performed by the fjof.cmd command, and in the Windows environment, simulation is performed by the fjwf.bat command. All batch files of types LX, BAT, and CMD are located in the subdirectory X:\CODOS\BIN. Examples of the generated fcwf.bat, fjwf.bat, and fpwf.bat commands for Windows are shown below:

```
 REM C++: fcwf.bat,% 1-FT name,% 2-type library (LIB or DLL)
 icc  /qautoimported /Ss /G4 /Q /W2 /C  %1.cpp
 IF "%2"=="DLL" goto dll
 goto lib
 :DLL
 ilink %1.obj /pm:vio /noe  icwf.lib icwbits.lib icwintv.lib
 goto end
 :LIB
 ilink %1.obj /pm:vio /noe  lcwf.lib lcwbits.lib lcwintv.lib
 :end
 SET DD:CODE=%1
 SET DD:Lang=%3
 %1.exe
REM JAVA: fjwf.bat,% 1-FT name,% 2-drive name
SET CLASSPATH=c:\java\lib;%2:\codos\java\ljof.zip;%2:\fa\jav
IF EXIST %1.lsg del %1.lsg
IF EXIST %1.msg del %1.msg
if exist %1.java goto begin
if exist %1.jav copy %1.jav %1.java
:begin
javac %1.java >%1.err
echo java begin....
SET user=%1,%2,,,%3,
if exist %1.class java -Duser.name=%user% %1 >%1.msg
goto end
:delete
```

```
if exist %1.class del %1.class
if exist %1.java del %1.java
:end@echo off

REM PL / 1: fpwf.bat,% 1-FT name,% 2-name of the disk,% 3-type library
pli %1.pli   ( m s  default(linkage(system)) langlvl(saa2))
IF "%2"=="DLL" goto dll
goto lib
:DLL
ilink %1.obj  /nologo ipwf.lib
goto end
:LIB
ilink %1.obj  /nologo lpwf.lib
:end
SET DD:SYSIN=%1.dat
SET DD:DBT=%3codos\bin\uipcad.dbt
SET DD:GOUT=%1.lsg
SET DD:SYSOUT=%1.lsp
SET DD:SYSPRINT=%1.msg
SET DD:Lang=%4
%1.exe
```

The JDK JAVA development kit is included with Windows, Linux, and OS/2 and is installed during the installation of the operating system. The following statements are generated in the CONFIG.SYS OS/2 configuration file:

```
LIBPATH=X:\JAVA\DLL
SET PATH=X:\JAVA\BIN;
SET INCLUDE=X:\JAVA\INCLUDE; X:\JAVA\INCLUDE\OS2;
SET LIB=X:\JAVA\LIB;
SET CLASSPATH=X:\JAVA\LIB\CLASSES.ZIP,
```

where X is the name of the drive.

In the environment Windows JDK JAVA development kit is installed additionally in the JAVA subdirectory and in the AUTOEXEC.BAT file, or the following statements must be included in the environment variables:

```
SET PATH=C:\JAVA\BIN;
SET INCLUDE=C:\JAVA\INCLUDE; C:\JAVA\INCLUDE\WIN32;
SET LIB=C:\JAVA\LIB;
SET CLASSPATH=C:\JAVA\LIB\CLASSES.ZIP;
```

An example of the batch file fcof.cmd in REXX for executing the Federal Law in C++ in the OS/2 environment is shown below, where the variable name is the name of the Federal Law:

```
/* fcof.cmd */
    parse arg names
    name = word(names,1);  libsys = word(names,2);
    DN=VALUE('CODLOC',,'OS2ENVIRONMENT')
    'IF EXIST 'name'.lsg del 'name'.lsg'
'icc /Ss /G4 /Q /W2 /C ' name'.cpp'
if libsys='DLL' then    'ilink' name'.obj icof.lib icobits.lib icointv.lib'
 else    'ilink' name'.obj lcof.lib lcobits.lib lcointv.lib'
'SET DD:CODE='name
name
```

In the examples, OS/2 is installed on drive Q, and Windows or MSDOS установлена on drive C (H). If you are installing on other drives, you must replace the drive name.

When installing CAD UI using Visual Age tools in Windows and OS/2 operating systems, questions may arise. For some of them in table 6.3 answers are given.

Table 6.3

Solving issues when installing UI CAD

Question	Answers for operating systems	
	WINDOWS	OS/2
Additional CODOS menu does not load when loading editor	Check environment variable settings	
	in the autoexec.bat file: set LPATH, set PLILPATH4	in the config.sys file: set LPATH, set LPATH2
Files from the INCLUDE and INCPLI subdirectories are not included in the FT	Check settings in configuration files	
	in the autoexec.bat file: set Include	in the config.sys file: set Include
PCAD does not load in DOS session	The presence of the PCADDRV.SYS file in the root directory of the drive on which PCAD is installed	
	PCAD only works in full screen mode	
	The presence of paths for the OS/2 operating system in the config.sys and autooecec.bat file for the DOS session, and for Windows – only in autoexec.bat	
Access to the subdirectory X:\FA\CPP on a disk other than the specified X:	Check settings in configuration files	
	in the autoexec.bat file: set codloc	in the Config.sys file: set codloc
The JDK is installed in the X:\JAVA directory, and formalized tasks in the Java language are not performed	The paths to the main components of the JDK are not installed	

The OS/2 CONFIG.SYS configuration file is supplemented by specifying paths to batch files and modules in the SET PATH = D:\CODOS\BIN line, to include files in the SET INCLUDE = D:\CODOS\INCLUDE and D:\CODOS\INCPLI line, to dynamic libraries in the line LIBPATH =

D:\CODOS\LIB. To use the COD CAD system with various operating systems, it is recommended to install it on a disk accessible to all systems, for example D. There is also an FA catalog with test and working examples, with component libraries and drawing frames for industrial CAD systems.

6.3. Network services and formalized tasks in the Internet environment

The fifth version of the software and methodological complex UI CAD is designed to teach design methods and the basics of computer CAD. The fifth version was implemented using standard Internet tools: a www server and browsers (Mozilla, Netscape Communicator or Microsoft Internet Explorer) for IBM PC or web browsers for network mobile computers [153].

The software and methodological complex UI CAD COD (PMK) consists of the freely distributed Apache Web server [31, 153] (www.apache.com) for various platforms, CGI (Common Gateway Interface) software, interface, methodological and training software and formalized assignments. The CGI-interface software is implemented on the basis of the REXX and PHP languages available for various operating systems [27, 31]: OS/2, Windows, LINUX, VM/ESA, zVM. All of these operating systems provide the use of multiprocessor computing systems [4, 27, 31, 51, 53, 119, 137].

PMK is available on the Internet at addresses http://e.sfu-kras.ru, http://study.sfu-kras.ru and on an optical disk (CD-ROM) depending on the user's capabilities and network bandwidth. Using PMK on a network server allows you to use the methodological and software without installing it at the user. To be able to view the simulation results, you need to install the program for viewing digital and analog signals UISGO.EXE for OS / 2 and USGW.EXE for Windows, which is called by the type of the lsg file.

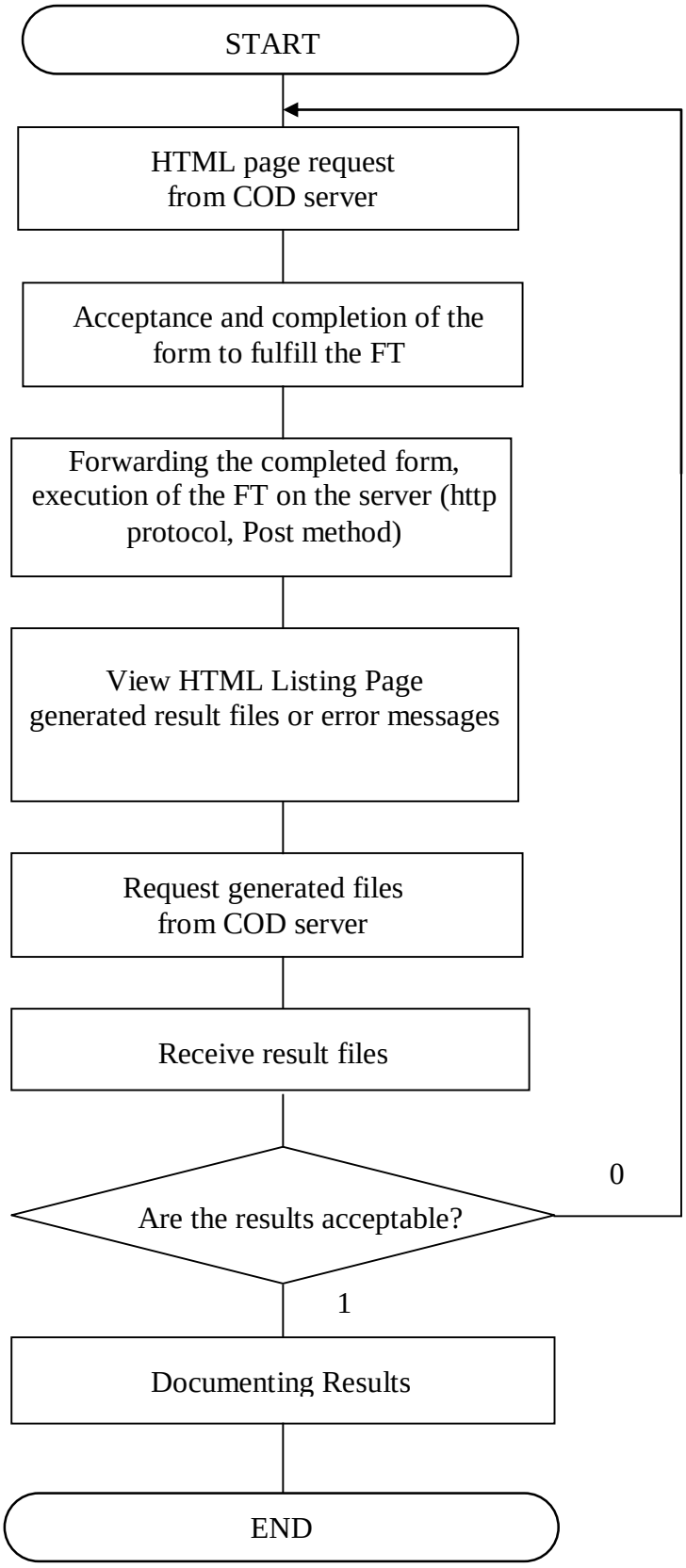

Fig. 6.1. The block diagram of the route for performing formalized tasks in various operating systems in the Internet environment

The teaching materials are presented in Russian and English languages by the course program “Design Methods and CAD Computer”, designed for two semesters, and control questions for each of the semesters.

The focus of the fall semester is on modeling computing devices and systems, methods for constructing micromodels and macromodels of components, and comparing expected and actual results. The complexity of the design problems solved at the PMK belongs to the second level [27, 103], which corresponds to the set of design decisions.

In the process of automated analysis, an automatic assessment of resources and the calculation of performance criteria are performed, which facilitates the structural optimization of computing devices and systems.

In the next semester, the main focus is on specific industrial CAD systems and the automatic conversion of a high-level description of many options in one formalized task into many schemes for a particular CAD system. To use foreign CAD systems, Eagle, PCAD 2001-2006, ORCAD, CADdy need libraries of domestic components. Component libraries can be created on the server automatically if there is information support in the form of tables.

The contents of the optical disk and www-server are supplemented by educational versions of CAD electronic equipment (PCAD, ORCAD, CADdy) and temporary versions of Visual Age tools for performing tasks, development and development of the complex for various operating environments (Windows, OS/2, LINUX). The software-methodical complex (PMC) consists of software in the COD and CODOS catalogs, test cases and component libraries in the FA catalog. The content of the CD-ROM and www-server ensures the implementation of laboratory, term papers and dissertations in local and remote versions on a computer network and contributes to the individualization of training.

The execution route of formalized tasks in various operating systems in the environment of the program for viewing web pages is shown in fig. 6.1. A formalized task (FT) must be generated by the user and sent to the server, or you can execute the FT from a variety of examples available on the server. The first time you access the application server, it is recommended that you complete the example project to verify the health of the entire system.

Having connected to the server on which the PMC COD V.5 is located, the user gets acquainted with the manuals and finds an HTML page for entering data into the formalized task execution form.

The name of the Federal Law is filled in and the required results are entered. The results can be a file with the data of analysis and resource estimation, a diagram in the format of a specific CAD system, for example PCAD, or a batch file for executing electronic equipment in one of the CAD systems (PCAD, ORCAD, ALTIUM, EAGLE, CADdy). After filling in the required results and confirming readiness, the HTML page is sent to the server, where the Federal Law is executed.

As a result, the user receives a list of generated files or error messages. After receiving the generated files, the validity of the results is checked. If valid,

the results are saved; otherwise, it is necessary to repeat the synthesis or correct the FT.

Manuals and teaching materials are best placed on a network server, and formalized tasks are performed on a variety of application servers in various operating systems. A user can work in the same operating system with a network viewer, and run it in various environments [22–26, 47, 50, 53]. Various network devices can be connected to the application server to compare the simulation results and the experiment (fig. 6.2). An analog-digital device from National Instruments USB DAQ and Lab View 8.2–8.5 software was used [29].

Consider a formalized task for modeling and working with real usb LSI module NI6009 in the file fp04ninp.cpp:

```
#include <uicpp.h>
void main() // fp04ninp
  {
int noutdac1=2; // output number dac1
int noutdac2=3; // output number dac2
int mninadc[8]={4,0,0,0,0,0,0,0,0}; // array of ADC data circuit numbers
int mndio1[8],
mndio2[4]={5,6,7,8}; /* correspondence of circuit and pin numbers
        mndio2[0],
        mndio2[1] - control inputs of the multiplexer
        mndio2[2] - strobe
        mndio2[3] - control signal of the current generator
        */
 int mode; // ADC operation mode: 10126 - diff., 10083 - non-diff.
 #include <txtpar>
INIT: / * Set initial values * /
 nvarmax=1; ns=16; ntmax=20; nas=4; lres=-1; deltp=500; deltn=500; // ns
 mode=10083;
 #include <txtdcl>
INPUT: /* Generation of external influences */
  SignalA(2,2*cos(xt/3+1.0)+2,ntmin,ntmax); // analog signal 1
  SignalA(3,2*sin(xt/3+1.0)+2,ntmin,ntmax);  // analog signal 2
UNIT: /* Device Description */
   ADCUSB("NI6009",1,noutdac1,noutdac2,mninadc,mndio1,mndio2,mode);
   MKM("M686",1);
   APrint (5,12,ntmin,ntmax,2,4);
    #include <w>
}
```

Connecting analog-to-digital devices with digital and analogue inputs/outputs via USB or a network interface provides a high exchange rate while maintaining a guarantee for hardware and computer systems. Similarly, microcomputer-based network devices are connected to the programmable

technological equipment with access through network programs for viewing information Mozilla Firefox, Google Chrome. Therefore, production personnel can control the process through a high-speed network, regardless of the location of the facilities. This is especially important for distributed objects, such as energy.

The menu of a working multi-platform application server is shown in fig. 6.3. The main menu items for performing formalized tasks correspond to the local and remote menu options in the Lpex editor or Firefox, Chrome browser.

Microcomputer-based network devices are specialized application servers with a network address, management, and access using a network viewer. Developed for different platforms, programs for viewing time diagrams, circuits, and three-dimensional models with the display of signals and temperature will find application in such systems.

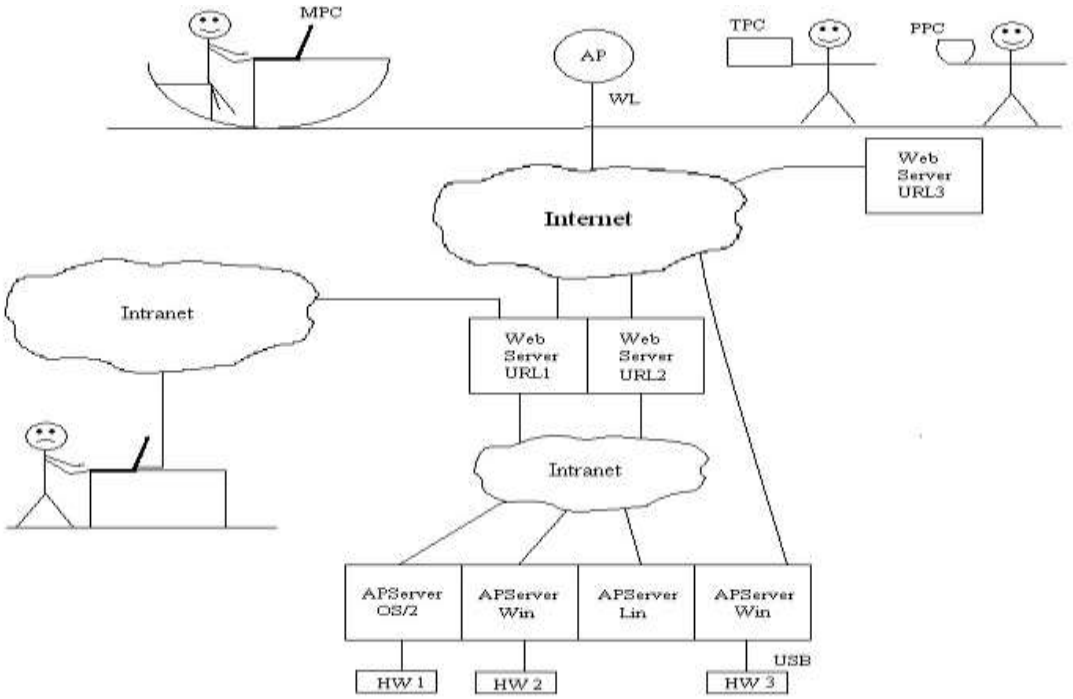

Fig. 6.2. A fragment of a network with web servers, APServer application servers, fixed and mobile clients. Various HWi hardware systems are connected to application servers

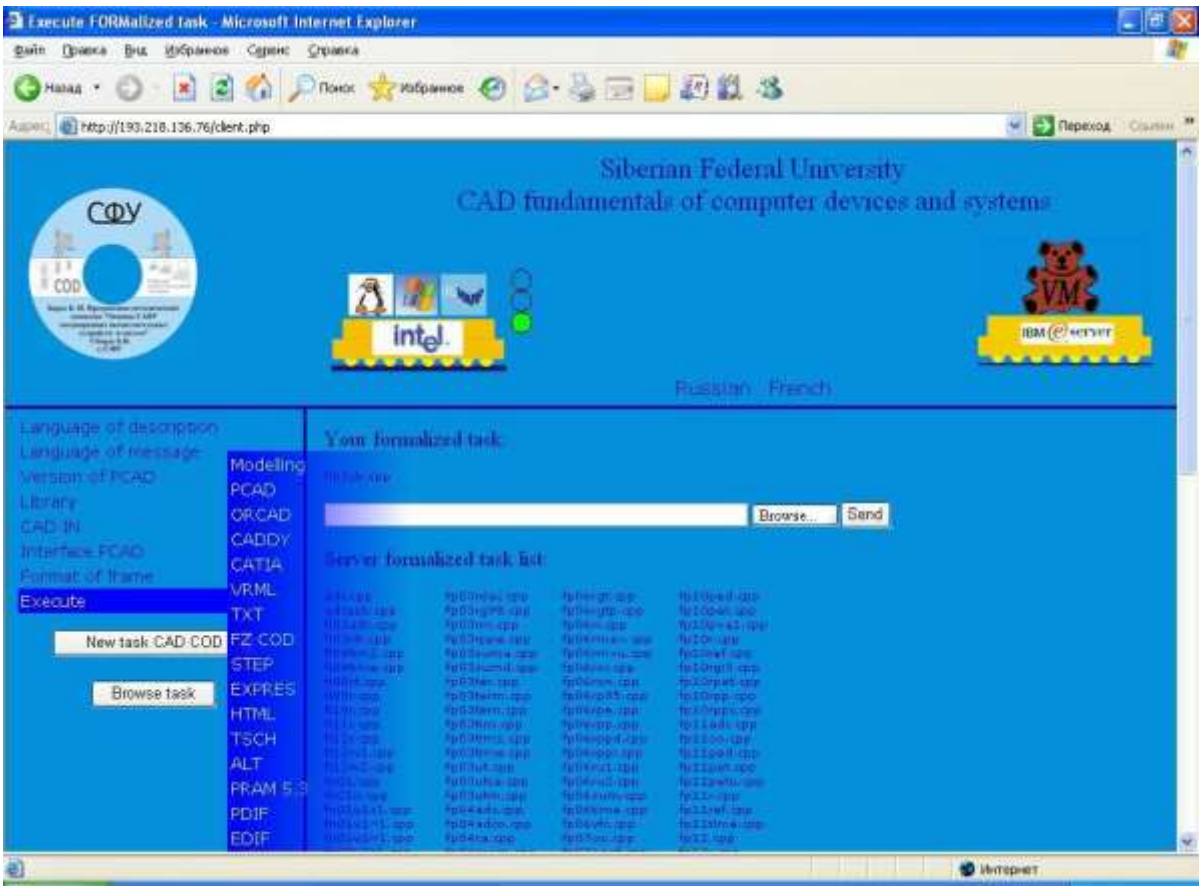

Fig. 6.3. Menu of a working multi-platform application server

CONCLUSION

The work performed by the author on automation methods for modeling and designing heterogeneous computing systems yielded the following results: a methodology was created and the problem of automation of a multivariate object was solved component modeling and design at different levels of abstraction of the creation of objects and developed an open modular developed multilevel software and methodological complex. In the design process, a decision table is formed, and the quality of the product depends on the number of effective iterations of creating the object.

The main scientific and practical results are given below.

1. **The problem of creating an open modular developed multilevel software and methodological complex** that allows one to obtain the necessary representations of the projected object, resources for creating options, performance criteria for making decisions and obtaining descriptions for the next design stage from one formalized description of the set of options for the object is obtained.

2. **Created representations of models of many technical solutions of computing systems at various levels of abstraction in high-level languages in various syntactic environments and algorithms for their construction for various applications.** A new combination in one description of computing systems at various levels of abstraction and algorithms for their transformation in various syntactic environments reduce the volume of descriptions, allow for conceptual unity and reduce the possibility of environmental errors.

3. **A methodology and algorithms for the operation of generalized multifunctional component models with a common interface have been created.** New multifunctional models of components with a common interface are associated with common data for various models and applications, and allow you to get many applications from one description of an object. Multifunctional component models with a common interface significantly reduce the complexity of design, especially in the initial stages.

4. **Created algorithms for automatic resource estimation tools, comparing the expected and actual results of analog-to-digital subsystems for structural optimization.** The transition to the second level of complexity of design tasks is impossible without software support for decision making. Decision support software includes information support, resource assessment programs, and key performance criteria. Along with traditional parameters, information support is supplemented by the energy and thermal characteristics of the components, allowing to evaluate the temperature of the components and the energy of solving a problem or service.

5. **Models and implementation algorithms for modular interfaces with circuitry and design engineering systems have been created**. A single representation of a variant of an object is transformed by a modular complex into representations perceived by a wide variety of specialized and complex systems of circuitry and design engineering. Tabular, command and text versions of

interface models have been built. The table interface is distinguished by the highest speed, and the command effect is clear and the learning effect. Of particular note is the automatic creation of formalized tasks in various syntactic environments.

6. Created a methodology for designing and using local and network service options programmatically methodological complex for various platforms. Programmatically the methodological complex was created at various stages of the development of hardware and software. An open modular principle with a single representation of an object variant in various syntactic environments made it possible to develop services of a virtual machine system with the capabilities of servers, workstations and mobile clients. The main results are transmitted in text form with interpretation on the client side in a network program for viewing results, which minimizes traffic.

Practical value and implementation of work results.

The main scientific works (see table A2.1) were performed by the author on topics that are important for the Russian Federation and the Krasnoyarsk Territory in accordance with industry plans, acts of testing, implementation and use of the results. The results were used in the educational process at various stages of the development of information technology with the recommendation of teaching aids and the program-methodological complex by the educational-methodical department and the Ministry of Education of Russia.

An automatic multivariate analysis of the operation of computing systems with an assessment of resources and performance criteria allows the user to solve design problems of the second complexity level and focus on creative synthesis and decision-making procedures, form the necessary representations of the designed object, resources for creating options, performance criteria for making decisions and obtain descriptions for the next stage of design.

BIBLIOGRAPHY

1. Actual problems of modeling in automation systems of circuit design / ed. A. L. Stempkovsky. – M.: Nauka, 2003. – 430 p.

2. Artamonov, E. I. Structural design of systems / E. I. Artamonov // Information technologies in design and production. – 2008. – No. 2. – P. 3–10.

3. Artamonov, E. I. Interactive systems. Synthesis of structures / E. I. Artamonov. – M.: Insvyazizdat, 2010. – 185 p.

4. Aleksandrov, A. Spirals of hardware virtualization. SOA for networks / A. Aleksandrov // Open Systems. – 2007. – No. 3. – P. 13–37.

5. Alekseev, A. V. Programming in the dialog processing subsystem of the EU CBM / A. V. Alekseev. – M.: Radio and communications, 1990. – 256 p.

6. Analog and Digital Integrated Circuits / ed. S. V. Yakubovsky. – M.: Sov. Radio, 1979. – 336 p.

7. Arais, E. A. Modeling of heterogeneous circuits and computer systems / E. A. Arais, V. M. Dmitriev. – M.: Radio and communications, 1982. – 160 p.

8. Armstrong, J. R. Modeling of digital systems in the VHDL language: tran. from English / J. R. Armstrong. – M.: Mir, 1992. – 175 p.

9. General principles of design organization in KASPI / B. A. Babayan, O. K. Gushchin, G. G. Ryabov, A. M. Stepanov. – M.: ITMiVT, 1975. – 32 p.

10. Bashmakov, A. I. Intelligent Information Technology: textbook allowance / A. I. Bashmakov, I. A. Bashmakov. – M.: Publishing House of MSTU im N.E. Bauman, 2005. – 304 p.

11. Balashov, E. P. Design of information management systems / E. P. Balashov, D. V. Puzankov. – M.: Radio and communications, 1987. – 256 p.

12. Borde, B. I. Multilevel structural optimization of heterogeneous computing systems / B. I. Borde // Vestn. Krasnoyarsk. State Una., Ser. Physics and mathematics. – 2006. – Vol. 7. – P. 155–161.

13. Borde, B. I. Modeling of heterogeneous computing systems / B. I. Borde // Vestn. Krasnoyarsk. State Unthat Ser. Physics and mathematics. – 2005. – Issue 4. – P. 197–202.

14. Borde, B. I. Multifunctional Models of Components of Inhomogeneous Computing Systems / B. I. Borde // Computational Technologies. – 2005. – T. 10. Special. issue – P. 69–77.

15. Borde, B. I. Digital–to–analog converters with equalized weighted discharge coefficients / B. I. Borde // Problems of Radioelectronics. Ser. Electronic computing technology. – 1971. – Vol. 4. – P. 29–36.

16. Borde, B. I. Passive adder of a continuous conductive structure for a digital–to–analog converter / B. I. Borde // Problems of Radioelectronics. Ser. Electronic computing technology. – 1971. – Vol. 11. – P. 125–128.

17. Borde, B. I. Structures of programmable multifunctional analog–to–digital converters / B. I. Borde // Problems of Radioelectronics. Ser. Electronic computing technology. – 1978. – Vol. 2. – P. 83–88.

18. Borde, B. I. Digital–to–analog converters with active generators of discharge currents / B. I. Borde // Problems of Radioelectronics. Ser. Electronic computing technology. – 1978. – Vol. 4. – P. 87–97.

19. Borde, B. I. Transistor switching elements with control circuits on tunnel diodes / B. I. Borde // Avtometriya. – 1966. – No. 4. – P. 69–77.

20. Borde, B. I. Dynamic display of data on segment matrices / B. I. Borde, E. G. Petrikeev // Devices and Control Systems. – 1976. – No. 1. – P. 50–51.

21. Borde, B. I. Optoelectronic control of MNOP–memory matrices / B. I. Borde, V. L. Kuznetsov // Instruments and control systems. – 1979. – No. 9. – P. 32–33.

22. Borde, B. I. Syntactically controlled translation of a test task into a control program for a microcomputer / B. I. Borde, V. L. Kuznetsov // Programming. – 1980. – No. 4. – P. 61–63.

23. Automatic processing of the results of studies of the distribution of thermopower in semiconductor minerals / B. I. Borde, A. S. Guryevich, V. I. Krasnikov, V. G. Romanov, V. G. Cherepanov // Avtometriya. – 1977. – No. 4. – P. 25–30.

24. Borde, B. I. Data processing system for physical research of natural minerals based on microcomputers / B. I. Borde, V. G. Cherepanov // Control systems and machines. – 1979. – No. 3. – P. 9398.

25. Borde, B. I. Automation of the collection and processing of geological and mineralogical data in the exploration of ore deposits / B. I. Borde [et al.] // Geophysical equipment. – 1989. – Issue. 90. – P. 98–104.

26. Borde, B. I. Automation of data processing during tensile testing / B. I. Borde, T. A. Kuklina // Factory Laboratory. – 1983. – No. 8. – P. 88–89.

27. Borde, B. I. Program and methodological complex “Fundamentals of CAD of heterogeneous computing devices and systems” / B. I. Borde, Krasnoyarsk: KSTU, 1999. – CD–ROM (Russian, English). State number registration of STC INFORMREGISTR 0329900090.

28. Borde, B. I. Program and methodological complex “Fundamentals of CAD of heterogeneous computing devices and systems” / B. I. Borde, Krasnoyarsk: KSTU, 2001. – CD–ROM (Russian, English). State number registration of STC INFORMREGISTR 0320000143.

29. Borde, B. I. Program and methodological complex “Fundamentals of CAD of heterogeneous computing devices and systems” / B. I. Borde, Krasnoyarsk: KSTU. – 2006. – CD–ROM (Russian, English). State number registration of STC INFORMREGISTR 0320702238.

30. Borde, B. I. Fundamentals of CAD of computing devices / B. I. Borde – Krasnoyarsk: KrPI, 1987. – 63 p.

31. Borde, B. I. Fundamentals of CAD of computing devices and systems: / B. I. Borde. – Krasnoyarsk, KSU, 1989. – 176 p.

32. Borde, B. I. Fundamentals of CAD of heterogeneous computing devices and systems / B. I. Borde. Krasnoyarsk: – KSTU, 1996. – 248 p.

33. Borde, B. I. Fundamentals of CAD of heterogeneous computing devices and systems: textbook. allowance / B.I. Borde. – 2nd ed., revised. and add. – Krasnoyarsk. – CPI KSTU, 2001. – 350 p.

34. Borde, B. I. Calculation of optimal modes and accuracy of transistor switches for digital–to–analog converters / B. I. Borde // Automatic control and methods of electrical measurements: tr. IV All–Union. conf. / RIO SB AS USSR. – Novosibirsk, 1964. – T. 1. – P. 119–127.

35. Borde, B. I. Information performance of digital–to–analog converters with transistor switches / B. I. Borde // Automatic control and methods of electrical measurements. – Novosibirsk: Nauka, 1965. – P. 130–135.

36. Borde, B. I. Statistical analysis of ways to reduce the error of digital–to–analog converters with transistor switches / B. I. Borde // Automation systems for scientific experiments: tr. All–Union. conf. IAE SB AS USSR. – Novosibirsk: Nauka, 1970. – P. 358–361.

37. Borde, B. I. Statistical characteristics of the error in the operation of digital–to–analog converters / B. I. Borde // Automatic control and methods of electrical measurements: tr. VIII All–Union. conf. – Novosibirsk: Nauka, 1971. – P. 183–191.

38. Borde, B. I. Analysis of ways to reduce the error of the operation of transistor switches / B. I. Borde // Automation systems for scientific experiments: tr. All–Union. conf. – Novosibirsk: Nauka, 1971. – P. 154–159.

39. Borde, B. I. Design of a multilevel system for modeling heterogeneous computing systems using R–technology / B. I. Borde // R–programming technology and tools for its instrumental support. – Kiev: EC AN USSR, 1982. – P. 61–63.

40. Borde, B. I. Designing a multilevel system for choosing the structure and modeling of heterogeneous computing systems / B. I. Borde // Programming R–technology. – Kiev: IK AN SSSR, 1983. Part 2. – P. 24 – 26.

41. Borde, B. I. Integrated and specialized CAD in engineering education / B. I. Borde // Higher school on the path of reform: Sat. Vseros. conf. – Krasnoyarsk: Publishing house Klaretianum, 1998. – P. 171–172.

42. Borde, B. I. Development of structures of analog–digital subsystems of test automation / B. I. Borde // Automation of engineering research and experiment / ed. I. M. Wittenberg. – M.: MD NTP, 1978. – P. 56–61.

43. Borde, B. I. Evaluation of the effectiveness of distributed analog–to–digital subsystems / B. I. Borde // Improving the efficiency and quality of products based on the development of distributed control systems and the use of microprocessor devices. – M., 1978. – P. 80–81.

44. Borde, B. I. Optimization of distribution of subsystems of transformation and signal processing / B. I. Borde // Automation of Scientific Research. – Krasnoyarsk: Siberian Branch of the USSR Academy of Sciences IF, 1980. – P. 56–63.

45. Borde, B. I. Development of Computing and Communication Systems for Engineering Education / B. I. Borde // Improving the Quality of Continuing

Professional Education: materials of All–Russian. scientific method. conf. – Krasnoyarsk: KSTU, 2005. – P. 243–245.

46. Borde, B. I. Development of network educational services for multilevel engineering education / B. I. Borde // Improving the quality management system for training specialists: materials of the All–Russian. scientific method. conf. from the international participation. – Krasnoyarsk: KSTU, 2004. – P. 166–167.

47. Borde, B. I. Automation of the collection and processing of geological and mineralogical data in the exploration of ore deposits / B. I. Borde [et al.] // International Geological Congress: abstract. doc. – M.: Science, 1984. – P. 361–363.

48. Implementation of web services for CAD electronic equipment / B. I. Borde, E. V. Khabarov, C. I. Osit, D. A. Podkamenny // Problems of informatization of the region (PIR–2003). – Krasnoyarsk: KSTU, 2003. – P. 118–119.

49. Borde, B. I. Education–Research CAD of analog–digital computer units and system / B. I. Borde // Proceedings of First International Conference on Distance learning and new tehnologies in education (ICDED94). – 1994. – P. 277–278.

50. Borde, B. I. Modeling and Presentation local Network with mobile users / B. I. Borde // 14 IST SUMMIT. – Dresden, 2005.

51. A. S. No. 902242. Digital–to–analog converter / B. I. Borde // IPOTZ No. 4. – 1982.

52. Borde, B. I. Model of a relay system for controlling an inertial object / B. I. Borde // Krasnoyarsk: KrPI, 1969. – 15 p.

53. A. S. 367541. Galvanic separation of analog signals / B. I. Borde // IPOPS No. 8. – 1973.

54. Borde, B. I. Research and Industrial CAD in engineering education / B. I. Borde // International conference of Engineering Education ICEE95. – M., 1995. – P. 240.

55. Borde, B. I. Automation of analysis and presentation of the set of results of heterogeneous computing systems / B. I. Borde // Sat. doc. conf. CAD / CAM / PDM–2007. – M.: IPU RAS, 2007. – P. 42–45.

56. Borde, B. I. Automation of the formation of images of objects of computing systems. / B. I. Borde // Models and image processing methods. MMOI–2007: Materials of the All–Russian. scientific conf. – Krasnoyarsk, 2007. – P. 112–117.

57. Borde, B. I. Program and methodological complex for multivariate design of heterogeneous computing systems / B. I. Borde. – Sat. doc. conf. CAD / CAM / PDM–2008. M.: IPU RAS, 2008. – P. 23–26.

58. Bukhteev, A. Methods and tools for designing systems on a chip / A. Bukhteev // CHIP NEWS. – 2003. – No. 4. – P. 4–14.

59. Venger, O. V. Change of the LSI design route during the transition to nanotechnology / O. V. Venger, A. V. Zhmurin, D. A. Rybin // Information Technologies. – 2005. – №. 5. – App. – P. 2–4.

60. Vermishev, Yu. Kh. Methods of automatic search for solutions in the design of complex technical systems / Yu. Kh. Vermishev. – M.: Radio and communications, 1982. – 152 p.

61. Vermishev, Yu. Kh. Fundamentals of design automation / Yu. Kh. Vermishev. – M.: Radio and communications, 1988. – 280 p.

62. Vermishev, Yu. Kh. Management of the development of a complex object / Yu. Kh. Vermishev // Information Technologies in Design and Production. – 2004. – No. 2. – P. 4–12.

63. Vermishev, Yu. Kh. The concept of synthesis of technical solutions / Yu. Kh. Vermishev // Information Technologies in Design and Production. – 2007. – No. 1. – P. 49–55.

64. Gagarin, A. P. On the need to form a national base for system design of infocommunication systems and technologies / A. P. Gagarin, E. I. Shulgin // Information Technologies in Design and Production. – 2007. – No. 1. – P. 3–9.

65. Gitis, E. I. Analog–to–Digital Converters / E. I. Gitis, E. A. Piskulov. – M.: Energoizdat, 1981. – 360 p.

66. Glushkov, V. M. Computer–aided design of computing machines / V. M. Glushkov, Yu. V. Kapitonova, A. A. Letichevsky. – Kiev: Science. Dumka, 1975. – 231 p.

67. Gorban, A. N. Neuroinformatics / A. N. Gorban [et al.]. – Novosibirsk: Nauka, 1998. – 296 p.

68. Gribov, V. V. Development of the ontological approach to automate the development of user interfaces with dynamical data / V. V. Gribov // Information Technologies. 2010. – No. 10. – P. 54–58.

69. Gurevich, I. M. Laws of informatics – the basis of research and design of complex systems / I. M. Gurevich // Information Technologies. – 2003. – App. No. 11. – 24 p.

70. Gusakov, A. A. Construction system engineering / A. A. Gusakov. – M.: Stroyizdat, 1993. – 368 p.

71. Dobronets, B. S. Interval mathematics: textbook allowance / B. S. Dobronets. – Krasnoyarsk: KSU, 2004. – 216 p.

72. Druzhinin, V. V. Problems of Systemology / V. V. Druzhinin, D. S. Kontorov. M.: Radio and communications, 1976. – 276 p.

73. Druzhinin, V. V. System engineering / V. V. Druzhinin, D. S. Kontorov. – M.: Radio and communications, 1985. – 200 p.

74. Evgenev, G. B. Systemology of Engineering Knowledge / G. B. Evgenev. – M.: Publishing House of MSTU im N.E. Bauman, 2001. – 376 p.

75. Evreinov, E. V. Homogeneous computing systems, structures and environments / E. V. Evreinov. – M.: Radio and communications, 1981. – 208 p.

76. Evreinov, E. V. Homogeneous computing systems / E. V. Evreinov, V. G. Khoroshevsky. – Novosibirsk: Nauka, 1978. – 319 p.

77. Elshin, Yu. M. GRIF–4, Information and software complex for expanding CAD functional P–CAD 200X. M.: PAO NPO << DIAMOND >> 2017. – 496 p.

78. Elshin, Yu. M. Automated workplaces in the design of REA / Yu. M. Elshin. – M.: Radio and communications, 1983. – 128 p.

79. Elshin, Yu. M. Electronic industry: a step towards CALS–technologies – ODB++ standard / Yu. M. Elshin. – M.: Rodnik Soft, EDA Express No. 7. – 2003. – P. 16–24.

80. Zaripov, R. Kh. Machine search for options in modeling the creative process / R. Kh. Zaripov. – M.: Nauka, 1983. – 232 p.

81. Engineering support for the life cycle of electronic tools: monograph / V. V. Goldin, V. G. Zhuravsky, A. V. Sarafanov, Yu. N. Kofanov. – M.: Radio and communications, 2002. – 379 p.

82. Karman, T. Mathematical methods in engineering / T. Karman. M. Bio, – tran. from Eng. M. G Shestopal. – M.: OGIZ, 1948 – 424 p.

83. Kalyaev, A. V. Multiprocessor systems with programmable architecture / A. V. Kalyaev. – M.: Radio and communications, 1984. – 240 p.

84. Colombet, E. A. Microelectronic means for processing analog signals / E. A. Colombet. – M.: Radio and communications, 1991. – 376 p.

85. Korobkov, B. P. Models of structural adaptation in the processes of managing complex objects / B. P. Korobkov, L. A. Rastrigin. – Riga: Zinatne, 1977. – P. 3–21.

86. Korolev, L. N. Computer structures and their mathematical support / L. N. Korolev. – M.: Nauka, 1974. – 256 p.

87. Koryachko, V. P. Theoretical Foundations of CAD / V. P. Koryachko, V. M. Kureichik, I. P. Norenkov. – M.: Energoizdat, 1987. – 400 p.

88. Kuzin, E. S. Knowledge Representation and Solving Information–Complex Problems in Computer Systems / E. S. Kuzin // Information Technologies. Adj. – 2004. – No. 4. – 32 p.

89. Lazarev, I. A. Compositional design of complex aggregative systems / I. A. Lazarev. – M.: Radio and communications, 1986.– 312 p.

90. Lapko, A. V. Nonparametric information processing systems / A. V. Lapko, S. V. Chentsov. – M.: Nauka, 2000. – 350 p.

91. Lebedev, S. A. High–Speed Universal Computing Machines / S. A. Lebedev // Tr. conf. “Ways of development of Soviet mathematical engineering and instrumentation”. – M.: VINITI, 1956. – P. 31–43.

92. Mayorov, S. A. Structure of electronic computers / S. A. Mayorov, G. N. Novikov. – L.: Mechanical Engineering, 1979. – 384 p.

93. Mesarovich, M. General theory of systems: mathematical foundations / M. Mesarovich, J. Takahara. – M.: Mir, 1978. – 311 p.

94. Guidelines for using the electrical properties of ore minerals to study and evaluate endogenous deposits. – L., 1983. – 90 p.

95. Moiseev, N. N. Mathematical problems of system analysis / N. N. Moiseev. – M.: Nauka, 1981. – 488 p.

96. Methods of optimization / N. N. Moiseev [et al.]. – M.: Science, 1978. – 352 p.

97. Narignani, A. S. Introduction to Uncertainty / A. S. Narignani // Information Technologies. Adj. – 2007. – No. 4. – 32 p.

98. Netrebenko, K. A. Digital voltage dividers / K. A. Netrebenko. – M.: Energy, 1970. – 224 p.

99. Nefedov, A. V. Integrated circuits and their foreign analogues: ref. 11 t. – T. 5 / A.V. Nefedov. – M.: IP Radio Soft, 2000. – 608 p.

100. Neumann, J. Theory of self–reproducing automata / J. Neumann. – M.: WORLD, 1971. – 382 p.

101. Norenkov, I. P. Systems of computer–aided design of electronic computing equipment / I. P. Norenkov, V. B. Manichev. – M.: Higher school, 1983. – 272 p.

102. Norenkov, I. P. Development of computer–aided design systems: textbook for universities / I. P. Norenkov. – M.: MSTU im. N.E. Bauman, 1994. – 207 p.

103. Norenkov, I. P. Fundamentals of Computer Aided Design: textbook for universities / I. P. Norenkov. – M.: MSTU im. N. E. Bauman, 2000. – 360 p

104. Norenkov, I. P. Information support of high technology products / I. P. Norenkov, P. K. Kuzmik. – M.: MSTU im. N. E. Bauman, 2002. – 320 p.

105. Norenkov, I. P. Computer Science at a Technical University / I. P. Norenkov, A. M. Zimin. – M.: MSTU im. N. E. Bauman, 2004. – 352 p.

106. Norenkov, I. P. Genetic algorithms for finding solutions in ontological knowledge bases / I.P. Norenkov // Information Technologies. – 2010. – No. 9. – P. 20–24.

107. Norenkov, I. P. Telecommunication Technologies and Networks / I. P. Norenkov, V. A. Trudonoshin. – M.: MSTU im. N. E. Bauman, 2000. – 248 p.

108. Ovchinnikov, V. A. Algorithmization of combinatorial–optimization problems in the design of computers and systems / V. A. Ovchinnikov. – M.: MSTU im. N. E. Bauman, 2001. – 288 p.

109. Fundamentals of the theory of computing systems / ed. S. A. Mayorova. – M.: Higher school, 1978. – 408 p.

110. Fundamentals of computer–aided design: textbook / ed. A. P. Karpenko. – M: INFRA M, 2015. – 329p

111. Petrov, G. M. Trends in the Development of Structures of Analog–Digital Computing Systems / G. M. Petrov, Yu. A. Shubin // Radioelectronics Issues. Ser. Electronic computing technology. – 1974. – Vol. 2. – P. 111–128.

112. Polovinkin, A. I. Fundamentals of engineering creativity / A. I. Polovinkin. – M.: Mechanical Engineering, 1988. – 368 p.

113. Polyakov, A. K. Modeling of digital equipment based on the PL / 1 language / A. K. Polyakov. – M.: MPEI, 1978. – 68 p.

114. Polyakov, A. K. Computer simulation on a computer / A. K. Polyakov. – M.: MPEI, 1981. – 105 p.

115. Polyakov, A. K. Languages VHDL and VERILOG in the design of digital equipment / A. K. Polyakov. – M.: Solon–Press, 2003. – 230 p.

116. Potapov, Y. Review of CAD of printed circuit boards / Y. Potapov // CHIP NEWS. – 2003. – No. 4. – P. 36–39.

117. Potemkin, I. S. Automation of the synthesis of functional circuits / I. S. Potemkin. – M.: Energoizdat, 1981. – 88 p.

118. Information Converters in Analog–Digital Computing Devices and Systems / ed. G. M. Petrova. – M.: Mechanical Engineering, 1973. – 369 p.

119. Programming, debugging and solving problems on a single series of computers. Language PL / 1 / ed. I. A. Kudryashov. – L.: Energoatomizdat, 1989. – 280 p.

120. Razevig, V. D. P–CAD System 8.5–8.7: User Manual / V. D. Razevig. – M.: Solon–R, 1999. – 720 p.

121. Razevig, V. D. PCC Design System ACCEL EDA 15 (P–CAD 2000) / V. D. Razevig. – M.: Solon–R, 2000. – 416 p.

122. Razevig, V. D. ORCAD Digital Device Design System / V. D. Razevig. – M.: Solon–R, 2000. – 160 p.

123. Melekhin V. F., Pavlovsky E. G. Computing machines, systems and networks / V. F. Melekhin, E. G. Pavlovsky. – M.: Academy, 2006.

124. Ruban, A. I. Methods of optimization: textbook allowance / A. I. Ruban. – Krasnoyarsk: Research Institute of IPU, 2001. – 528 p.

125. Ryabov, G. G. Choice of functional diagrams of computer nodes for integral execution / G. G. Ryabov, V. G. Sokolov, V. S. Chunaev. – M.: ITMiVT, 1969. – 83 p.

126. Ryabov, G. G. Intellectual objects – a concept from computer technology / G. G. Ryabov, V. V. Suvorov // Information Technologies. – 2004. – No. 6. – P. 9–16.

127. Ryabov, G. G. Integrated fundamental research of intelligence: the path to the creation of computer technologies of new generations / G. G. Ryabov, V. V. Suvorov // Computational methods and programming. – 2004. – No. 2. – P. 206–209.

128. Computer–aided design systems in electronics: ref. / ed. I. P. Norenkova. – M.: Radio and communications, 1986. – 368 p.

129. Sabunin, A. G. Altium Designer / A. G. Sabunin // New solutions in the design of electronic devices. – M.: Solon–Press, 2010. – 432 p.

130. Sovetov, B. Ya. Information Technology / B. Ya. Sovetov. – M.: Higher School., 1994. – 368 p.

131. Solodovnikov, V. V. The principle of complexity in control theory / V. V. Solodovnikov, V. F. Biryukov, V. I. Tumarkin. – M.: Nauka, 1977. – 344 p.

132. Stempkovsky, A. L. System for the development of macro–models of analog and digital–analog nodes for CAD BIS / A. L. Stempkovsky, Yu. B. Egorov, A. A. Lyalinsky // Information Technologies. – 2000. – No. 2. – P. 31–33.

133. Sudov, E. V. Integrated Information Support for the Life Cycle of Engineering Products / E. V. Sudov. – M.: MVM, 2003. – 264 p.

134. Suchkov, D. I. Adaptation of CAD / CAD / PCAD to domestic technological equipment / D. I. Suchkov. – Obninsk: Prism, 1993. – 460 p.

135. Trudonoshin, V. A. Modeling of electromechanical systems using the PA9 software : methodical complex / V. A. Trudonoshin, I. V. Trudonoshin // Information Technologies. – 2006. – No. 4. – P. 10–13.

136. The creation of VRML–worlds: tran. from Eng. / E. Tittel, C. Sanders, S. Charlie, P. Wolf. – M.: Publishing House BHV group, 1997. – 320 p.

137. Toporkov, V. V. Choice of target architecture and resource allocation strategies of computing systems / V. V. Toporkov // Information Technology. 2004. – Adj. – 32 p.

138. Toporkov, V. V. Automation of joint design of hardware and software systems: a behavioral approach / V. V. Toporkov // Materials of the II Intern. conf. CAD / CAM / PDM–2002: T. 1. – M.: IPU RAS, 2002. – P. 26–37.

139. Educational research system of computer–aided design of electronic circuits: textbook allowance / ed. V. I. Anisimov. – L.: Leningrad State University, 1989. – 256 p.

140. Ushakov, V. B. Modeling facilities and ways of their development / V. B. Ushakov // Ways of development of Soviet mathematical engineering and instrumentation: tr. conf. – M.: VINITI, 1956. – P. 82–132.

141. Ushakov, V. B. Structural and functional capabilities of problem–oriented analog–digital computing systems of the third generation / V. B. Ushakov, I. M. Vitenberg, G. M. Petrov // PISU. – 1979. – No. 5. – P. 5–8.

142. Fayzulaev, B. N. Ultimate performance and basic laws of development of logical LSI computers / B. N. Fayzulaev // Microelectronics and semiconductor devices. – 1984. – Vol. 8. – S. 5–15.

143. Fedorov, I. B. Organization of training of specialists in the field of computer science, engineering and technology, reflected in Russian and foreign educational documents / I. B. Fedorov, I. P. Norenkov, S. V. Korshunov // Information Technologies. – 2006. – No. 9. – P. 73–77.

144. Fleishman, B. S. Fundamentals of Systemology / B. S. Fleishman. – M.: Radio and communications, 1982. – 368 p.

145. Frolov, A. V. Operating System IBM OS / 2 Warp / A. V. Frolov, G. V. Frolov. – M.: Dialog–Mifi, 1996. – 272 p. (System Programmer Library, T. 20).

146. Fu, K. Structural methods in pattern recognition / K. Fu. – M.: Mir, 1977. – 318 p.

147. Halstead, M. Kh. The Beginning of the Science of Programs / M. Kh. Halsted. – M.: Finance and statistics, 1981. – 128 p.

148. Tsapenko, M. P. Measuring Information Systems / M. P. Tsapenko. – M.: Energy, 1974. – 319 p.

149. Zirlin, A. M. Optimization Methods in Irreversible Thermodynamics and Microeconomics / A. M. Zirlin. – M.: Fizmatlit, 2003. – 416 p.

150. Tsibulsky, G. M. Multi–agent approach to image analysis / G. M. Tsibulsky. – Novosibirsk: Nauka, 2005. – 188 p.

151. Tsypkin, Ya. Z. Fundamentals of the theory of learning systems / Ya. Z. Tsypkin. – M.: Nauka, 1970. – 252 p.

152. Chekmarev, A. Visual Design Tools for JAVA / A. Chekmarev. – SPb.: BHV, 1998. – 400 p.

153. Shaposhnikov I. V. Webmaster Reference. XML / I. V. Shaposhnikov. – SPb.: BHV, 2001. – 304 p.

154. Sheremetov, L. B. Virtual educational environments / L. B. Sheremetov, V. L. Uskov // App. Information Technology. – 2002. – No. 5. – 24 p.

155. Shokin, Yu. I. Interval computing / Yu. I. Shokin. – Novosibirsk: Nauka, 1981. – 112 p.

156. Schreider, A. A. Systems and models / A. A. Schreider, A. A. Sharov. – M.: Radio and communications, 1982. – 152 p.

157. Stager, V. V. Digital Communication Systems. Theory, calculation and optimization / V. V. Stager. – M.: Radio and communications, 1993. – 312 p.

158. Ashby, W. R. Introduction to Cybernetics / W. R. Ashby. – M.: IIL, 1959.

159. Yanbykh, G. F. Methods of analysis and synthesis of computer networks / G. F. Yanbykh, B. Ya. Ettinger. – M.: Energy, 1980. – 96 p.

160. CADdy: Experience using technology. CAD and graphics. Specialist.issue – M.: Computer Press, 2000. – 112 p.

Used Internet Resources

I1. www.autodesk.com
I2. www.fsf.org
I3. www.cypress.com
I4. www.docinfo.ru
I5. www.eclipse.org
I6. www.ibm.com
I7. www.elibrary.ru
I8. www.intel.com
I9. www.intergraph.com
I10. www.inventionmachine.com
I11. www.iso.org
I12. www.java–source.net
I13. www.mentor.com
I14. www.microsoft.com
I15. www.cad.onshape.com
I16. www.mozilla.com
I17. www.nanocad.ru
I18. www.ni.com
I19. www.web3d.org
I20. bim–association.ru
I21. www.oracle.com

I22. www.parallelgraphics.com
I23. www.protege.stanford.edu
I24.www.tadviser.ru/index.php
I25. www.vmware.com
I26. www.w3c.org
I27.https://www.lektorium.tv/mooc2/26296
I28. rustandards.ru/docs/themes/13SP. 2016. “Automation Systems”.
I29. docs.cntd.ru/document/1200146763. Joint venture development “Information modeling in construction. Terms of use in reuse projects and their binding”.
I30. allgosts.ru/35/240/gost_r_57310–2016 Development of the joint venture “Information modeling in construction. Rules for developing project plans implemented using information modeling technology”.
I31. Professional standard for training specialists in the creation and modification of integration solutions, code 06.041 of the Ministry of Labor of the Russian Federation in 2017.

APPLICATIONS

Supplement 1

Brief Glossary

Analysis – design procedure for obtaining information about the properties or behavior of designed objects.

Data – information in a form suitable for automatic processing.

Basic data – entities, properties and relationships.

Integrated resources – many components for application protocols.

Interface – a set of tools and rules that ensure the interaction of parts of a computer system or software complex.

Command interface – automatic conversion of a high-level description of a design object into a sequence of commands of a specific low-level CAD system.

Table interface – automatic conversion of a high-level description of an object into the internal representation of an object of a specific low-level CAD system.

Performance criterion – measure of effectiveness or rule of preference for the effectiveness of a variant.

Design route – the sequence of design operations and procedures that implement the design stage of many objects.

Model functional – mathematical model reflecting the physical or informational state of the designed object and the process of changing states. When the internal states of individual elements of an object are reflected in the model, it is called a micromodel; otherwise, it is called a macromodel.

Physical contradiction – mismatch of storage media or channels.

Technical contradiction – improving one requirement for an object worsens another.

Design operation – part of the design procedure.

Design procedure – part of the design process, ending with a design decision and consisting of many design operations.

CAD (Computer Aided Design) – computer-aided design system.

High level CAD – serves to automate mainly algorithmic and functional-logical design stages, using high-level languages.

Low level CAD – serves to automate the final stages of the design process of specific objects (design and partially functional-logical).

Synthesis – design procedure for obtaining a description of the projected object in any language.

Object structure – component composition of the object and the relationship between components.

Design phase – part of the design process, ending with a design decision.

AAM (Application Activity Model) – functional model.

AIM (Application Interpreted Model) – model presented in the STEP standard (ISO 10303) in Express.

APIO (Application Programming Interface Overview) – application programming interface.

AP (Application Protocol) – application protocol in STEP, an information model of an object, consisting of entities and attributes, in Express.

ARM (Application Requirements Model) – data model that uses presentation and application terminology.

BIM (Building Information Model) **–** information model of a building or object. ISO 29481-1: 2016. ISO 29481-1: 2010. **GOST R 57310-2016 (ISO 29481-1: 2010) Guide for the delivery of information. Methodology and format**

CALS (Continued Acquisition of Lifecycle Support) – forms and methods of representing data on objects at all stages of the life cycle.

EDIF (Electronic Design Interchange Format) – data exchange format for the design of electronic equipment.

EXPRESS – modeling language, which has two forms of representation – textual, actually EXPRESS and graphical EXPRESS-G.

LPEX (Live Parsing Extensible editor) – multifunctional advanced editor that supports many programming languages and descriptions of technical solutions. The editor is part of IBM's Visual Age (VA) tools.

ONTOLOGY – formalized specification of conceptualization.

Abstraction Operation (L2) – saving a subset of the properties of objects.

Concretization Operation (L2) – expansion of many properties of objects.

Conceptualization – a description of the many concepts (concepts) of the subject area, knowledge about them and the relationships between them.

RDF – XML-based resource description format. RDFS is an RDF scheme. W3C Standards.

OWL – language for representing ontologies on the network. W3C Standards.

OWL-S – is a language for representing services on a network. It allows you to discover a service, call a service, create a composite service, and monitor the execution of the service. W3C Standards. Industrial automated systems and integration. Lifecycle data integration for processing plants. Part 1. GOST R ISO 15926-1-2008 (ISO 15926 1-11). The practical implementation of the network ontology language (OWL).

ST-EXPRESS – software for creating and displaying models in EXPRESS; allow you to create EXPRESS-G diagrams using the EXPRESS scheme.

STEP (Standard for the Exchange of Product model data) – the international standard GOST R ISO 10303 (ISO 10303) for describing the data models of industrial products and design objects at all stages of the life cycle.

SOC (System on Chip) – on-chip computing system.

SOAP (Single Object Accesses Protocol) – object access protocol.

SCA (Service Component Architecture) – service component architecture.

UML (Unified Modeling Language) – a unified modeling language.

Visual Age (VA) from IBM – a toolkit for creating software in various programming languages (VA C++, VA PLI, VA JAVA).

VRML (Virtual Reality Modeling Language) – virtual reality model description language ISO / IEC 14772-1.

XML (Extensible Markup Language) – advanced markup language.

X3D (Extensible 3D) – extensible 3D language **ISO / IEC 19775**.

WebGL – 3D format graphics supported by network viewers without additional software modules.

Supplement 2

Acts of use of work

Table A2.1

The list of works included in the coordination plans of the State Committee of the USSR (Ministry of Higher Education), the State Committee for Higher Education and Science of the RSFSR and line ministries carried out under the supervision of the author, and acts of use

Room	Document Number	Title of the document	Report title	Work results
1	2	3	4	5
1	Problem 080.250-D1 approved. SM of the USSR	Ministry of non-ferrous metallurgy of the USSR Thematic map of the Ministry of color 18-69-032	Report on the topic 59 “A device for calculating and analog-to-digital conversion of the reduced voltage and resistance of the electrolyze”, KPI, 1971, 180 p.	P1, P2 Test Acts and Implementation Effect at the Aluminum Smelters in 1970-1980 years
2	Order of the Ministry of Higher Education № 595 of 06/27/1971 p. 13	On the development of research in universities in the field of optimization and automation of research	Development and research of video encoding device “KOD4” Multichannel digital-to-analog converter with memory device “KOD5”	Acts of testing and effect of 12/19/1974, 12/13/1975 Acts of testing and effect of 04/16/1976 KB MRZ Moscow
3	Rosminvuz order № 394 of 09/17/1976	Design Automation. P. 09.07.03. Development and research of automation system for processing test results	Development and research of a system for automating the processing of test results Research report. KPI, Krasnoyarsk, № GR 78082237, Inv. № B996696, M: VNITICENTER, 1981, 123 p. (92 p.)	Act on the implementation of research from 03.03.1981
4	The template of the Ministry of Geology of the USSR dated 12/26/1977 XII-020 313-3	Research and development of the analog-to-digital data processing system for physical research of natural minerals “KOD8”.	Research report System for processing data from research on natural minerals based on the KPI microcomputer, Krasnoyarsk, № GR 75055070, Inv. № B625323, M.: VNITITSENTR, 1977. 277p. (60 p.)	Act on the implementation of the research work “KOD8” from 11/15/1982

Continuation of the table A2.1

5	Decree of the Council of Ministers of the USSR for the military-industrial complex № 35 of February 7, 1978.	Rosminvuz order 29ss dated 03/23/1978	An analog-digital subsystem for interfacing a computer complex with an object. Research report. KPI, Krasnoyarsk, GR81051088, Inv. № 0281.3014017, Moscow: VNITITSENTR, 1981, 235 p. (30 p.)	Act on the implementation of research from 4.11.1981
6	Mingeo Order № 409 of 09/09/1983	Rosminvuz order 131 chipboard dated 03/12/1979	The analog-digital system for processing the data of research on natural minerals "KOD8M" Report on research. KPI, № GR01814004783, Inv. № 02830028481, M.: VNITICENTER, 1983, 130 p. (23p.)	Act on the implementation of the research work “KOD8M” dated 11/15/1982
7	Mingeo Order № 212 of 04/25/1988	About acceptance tests of the automated system “KOD10”	Analog-digital terminal system for processing natural minerals research data “KOD10” Research report. KPI, Inv. № GR0288000956260062129, M.: VNITITSENTR, 1988. 109 p. (50p.)	Act on the implementation of research from 04/19/1989
8	The order of the Ministry of Higher Education № 1211 of December 29, 1978, Clause 2.1.1. The order of Rosmin-university 31 of 02.15.1983 p.3.182	Development of theoretical and algorithmic foundations for the automation of the design of analog-to-digital subsystems	Research work “Fundamentals of design automation of analog-to-digital computing systems.” Implemented “Basic software of the first phase of UI CAD”	The Act on the introduction of research into the educational process of the KPI from 01.04.1986

Continuation of the table A2.1

9	Order of the Ministry of Higher Education № 195 dated 03/16/1987	About development in 1986-1990 years of research on the creation of UI CAD and their subsystems in universities.	Clause 3.2.17. To develop a design automation subsystem for analog-to-digital computing systems.	UI CAD, study guide published in 1989 g. and with the stamp of the Ministry of Education of the Russian Federation in 1996 and 2001.
10	Grants of the Regional Science Foundation 2000, project "Integratio" 2001 A0020/2.1.		Textbook "Fundamentals of CAD of heterogeneous computing systems" with the stamp of the Ministry of Education in 2001	PMK on the CD "Fundamentals of CAD of heterogeneous computing systems" in 2001, 2006
11	EU SITE Project	IST mobile Summit	Modeling systems with mobile components	Dresden 2005
12	Order of SFU, project 001/3 2007 year.	Innovative educational programs (UMKD)	PMK on CD and UMKD "Modeling heterogeneous computing systems"	The program of the course, lectures, laboratory and term papers. Acts of use of SFU 38-1810.10 dated 05/14/2008

Supplement 3

Basic events in computer science and SibFU

Table A2.2

Fundamental events in computer science and Siberian Federal University

Events in KPI KSTU SFU	Storage and transmission of information	Events in the development of computer technology	Stages (years)
Building KPI / KPI transferred buildings on Mira Ave. 49 and Lenin 70. In the fall semester, classes were held at the afternoon and evening faculties. B. I. Borde was admitted to the KPI on September 15, 1956.	Conference << Ways of development of Soviet mathematical engineering and instrumentation >>	NPO Svetlana released the first transistors. Series production of desktop AVM MH7 with sets of diodes for continuous logic	1956
Creation of a laboratory of computer engineering (VT) as part of the department of higher mathematics. Rector V.N., Borisov and head of department. VM T.I. Vorobeva In 1960, B.I.Borde passed the Acad. S.A. Lebedev at ITMiVT RAS.	Learning the basics of computer technology and programming for all students of the KPI. Report at the KPI conference on a set of components for computers, including transistors in key modes. Continuing education at MPEI in 1959	Creation in the lab. VT KPI mock-ups of transistor computers and the acquisition of several MH7 machines. MH7 machines were designed to solve nonlinear differential equations up to 6th order. And they were very effective.	1957-1960
Accommodation in 1963 in the main building of the KPI 7 educational laboratories. On the 4th floor in the central part of the main building. In 1967, B.I. Borde defended his thesis	By order of the rector of the KPI Borisov V.N. 76 of 02.16.1963 organized an independent discipline of computer technology based on 7 laboratories. By order of the Minister V. Stoletov of June 23, 1969, the Department of Computer Engineering was established at the KPI as part of the RTF	Acquisition of MIR computer for engineering calculations. In 1969. Issue of training manuals on MH7 AVM and MIR computer programming.	1961-1970

Continuation of the table. A2.2

The development of MIR machines was easy, since the input language of MIR machines was Algol-60, translated into Russian ALMIR	Design and commissioning of the MIR computer hall for the independent work of KPI students in 1973. Design, installation and commissioning of the KPI computer center on the basis of ESEVM in 1976, with a staff of 16 units. Nuzhdin V.P. has been appointed the head of the CC. Since 1979, Ph.D. G.M. Tsibulsky, elected Dean of the FIVT and in 2006 Director of IKIT SFU	Acquisition of EC7906 terminal stations in 1976 and then EC7920. The beginning of paperless technology.	1971-1980
Access to international software for IBM 360-370 allowed KPI to enter the international achievements of software systems	The acquisition of personal computers (PC). Access to network services from PC or terminals	Organization of a global network structure with servers	1981-1990
For such systems, CAD systems appeared Computer-aided design systems in various industries	Software and CAD increased engineering productivity	Ubiquitous access to the global network. The beginning of wireless computer networks	1991-2000
CAD students can only be taught on open systems. Therefore, there was an order of the Minister for Educational Research CAD CAD number 195. from 16.03. 1987, Clause 3.2.17 approved the UI CAD CAD NVS COD	Creation of the data center of Siberian Federal University and access to educational content from anywhere in the world. Start mass online courses. Learning Without Borders	Mass production of mobile wireless network access devices	2001-2010
The emergence of cloud CAD, access to which is provided from anywhere in the network. Autodesk InfraWorks example	Unique Date Creation centers in Russia. An example is the data center of Sberbank in Skolkovo, a project that was created in CAD REVIT. Centralization of data storage and processing. It is planned to complete the transition to the new BIM technology in 2024.	Creation of centers and structures for the digital economy. New BIM codes of practice are mandatory from March 1, 2018.	2011-2024