IFAC

INTERNATIONAL FEDERATION
OF AUTOMATIC CONTROL

WARSZAWA 1969

Problems on Automation of Computer and System Design

Fourth Congress of the International Federation of Automatic Control
Warszawa 16–21 June 1969

SURVEY PAPER

51

NOT

Organized by
Naczelna Organizacja Techniczna w Polsce

PROBLEMS ON AUTOMATION OF COMPUTER AND SYSTEM DESIGN

V.M. Glushkov, Yu.V. Kapitonova, A.A. Letichevsky

Achievements of cybernetics made it possible to construct automated control systems. An important role in these systems is played by general-purpose computers. Designers of systems center their attention at problems of constructing computers.

Sophistication in computer structure and rapid changes in technology, together with solving problems of design proper put forward problems of automating the computer design.

Investigators were interested in solving the automation problem of designing computers long ago. Historically its solution was associated with an interesting example of using intellectual potentialities of computers when solving noncomputational problems arising during creative activity of an engineer rather than with acceleration of designing computers.

A number of works devoted to description of methods, algorithms and programs are known to solve some problems of the computer synthesis.

On analysing those works one can arrive at two conclusions. The first, positive - the problem solution of automation of designing such complex objects as computers is feasible even now, since all prerequisites for this are available. The second, negative - it is impossible to obtain a positive practical effect by solving individual problems of synthesizing computer schemes. Just because of that the system approach has become prevalent in the works on the design automation recently. Its essence consists in studying and solving by means of computers the entire complex of questions

and problems arising in design. This leads to the need for constructing automated systems of designing computers. In the system approach a large totality of means ensuring the maximum employment of a computer in the design process is developed instead of discrete algorithms, programs or devices. These means involve in the first place the hardware providing a speedy and convenient designer-computer exchange of information, as well as the developed software ensuring a speedy and convenient arrangement of complex programs (from standard programs), for solving particular problems of design, and allowing a quick introduction of changes into the project being worked out, and having a means for choosing the optimal version of the project from the designer's standpoint.

As an illustration of such completed and operating systems the IBM-360 technical design system developed by the USA IBM Corporation, and the Small system of digital automata synthesis worked out by the Institute of Cybernetics AS Ukr.SSR can be pointed out.

The creation of automated systems for designing computers is connected with overcoming a number of difficulties and solving complex problems.

The first difficulty is due to unstability and complexity of methods used in designing computers. It is typical nowadays to introduce new properties into the structure of computers and systems, namely: introduction the time-sharing mode of operation into a computer, complication of an input language, use of the input language interpretation principle, introduction of rich facilities of intercourse with users, etc.

The second set of questions is connected with the construc-

tion of specialized languages for formal description of a computer in the design process.

The third difficulty consists in the development of a special mathematical tool for solving problems of formal transformations, and for optimizing a project when the design process is underway.

The fourth complex of questions is associated with the general organization of data files, development of special means for the user-system intercourse, specialized systems of programming automation inclusive.

Design Technique

The object which any engineer aims at in designing implies attainment of computer circuits with a minimum money consuming hardware and a maximum speed. But the algorithm of achieving this object for the entire computer being designed is extremely complex. Therefore the general design problem is meant to be a sequence of problems.

From the designer standpoint, information on a computer to be constructed consists of at least two parts. The first part involves the algorithm of function of a computer under design, and the second - its structure. The execution algorithm specifies the correspondence between sets of values of input and output signals. The structure states a computer representation as a composition of other simpler devices.

The design process can be described as follows. Having an idea of the computer execution algorithm, an engineer designs its structure in a certain degree of detail. Analysing the obtained structure he either returns to the execution algorithm and introduces changes in it, or, using the structure and the execution algorithm of structure components he obtains the structure with a new degree

of detail.

In view of this, the design process of a modern computer consists of the following main stages : System, logical and technical.

During the system stage of design, a number of memory units and other devices of a computer under construction is determined, flows of requests in the computer calculation process are studied, a calculation process organization is chosen so as to ensure the maximum working capacity of devices and of the entire computer, basic algorithms are compiled characterizing operation of computer individual devices. As a result of the system stage of design a general computer block diagram and execution algorithms of individual devices are obtained.

During the logical stage of design, the functional circuits of devices are obtained by the execution algorithms of individual devices and the entire computer, and by the system of elements chosen. During this stage the problems are solved of obtaining the computer structure in more detail, whose components are elements of the selected system. The principal problems of this stage are problems of coding states of devices, construction of combination circuits controlling the switching of device states, and various optimization problems of structures obtained.

The contents of the third stage of design - technical - are problems of arrangement of obtained functional circuits in the structure. Wiring diagrams and documents for manufacturers are made during the technical stage of design. The main problems of this stage are various problems of layout, run, construction of wiring, and preparation of documents.

From the viewpoint of developing automated design systems all problems and accordingly algorithms of the technique can be

divided into three classes.

The first class contains algorithms-translators by means of which a natural change-over from one representation of a designed computer to the other is effected. The second class is composed of algorithms-transformers with the aid of which deep optimizing transformations on the level of one representation are carried out. The third class of algorithms represents estimation and simulation algorithms used for forecasting and evaluating characteristics of various representations, as well as for simulating the representations with the purpose of obtaining such characteristics.

The computer design automated system developed at the Institute of Cybernetics, Academy of Sciences, Ukr. S.S.R., is provided with the following basic design algorithms composing the technique : Construction of the general block diagram of a computer to be designed on the basis of its simulation as a queueing system; construction of the block diagram of a control device and an operational unit of the computer by the execution algorithm; construction of computer functional (logical) circuits on the potential elementary base; construction of wiring tables of layout of the functional circuits in structures; stating of documents for the project.

Languages for Declaration of Data in Design Process

Languages are needed for declaring a computer being designed on various stages of design. A sufficient number of languages for declaring the execution algorithms and structure of devices are currently available. On the system stage of design, the SOL (System Oriented Language) may be used for representing the computer as a queueing system. On the logical stage - the language of Boolean functions, on the technical stage - the language of wiring tables can be employed. No special language is necessary in the manual

design. For the description of circuits an engineer usually utilizes any means known to him.

When building automated design systems the question of languages for declaring data in a system becomes of great importance, since this question is connected, on the one hand, with the convenience of recording data on a device for a designer and, on the other hand, with the complexity of processing these data in the system. A language should reflect special properties of a given stage of design and be adapted for executing the design algorithms. The main problem connected with languages consists in creating algorithms - translators which allow to translate records from one language into the other. It is just algorithms - translators which condition the complexity of data processing in a system.

One more set of questions forms problems of an internal representation and storage of data on a computer under design in a system. The point is that data on a project grow when the design is in progress, and may acquire a rather large volume. For instance, the documentation of the MIR-I computer exceeded the volume of the computer itself many times over. Taking into account a comparatively small volume of memory of a medium-scale computers for which the automated design systems are developed, storage questions and, what is especially important - data changes in a project - become very significant. A difficult problem is presented by questions of organizing and coding the internal representation of data. The internal representation should be so chosen as to simplify the algorithm of transferring to the internal representation of data and their processing necessitated by needs for design. Several languages for describing a computer to be designed were developed at the

Institute of Cybernetics in Kiev.

During the system stage of design the SLANG language is used for declaring a computer model as a special-purpose queueing system. This language has special means for declaring flows of demands of various processors which are delt with in designing, as well as means for simulating the system time and for constructing the distribution of guantities obtained as a result of simulation. Appendix 1 of this paper represents, for instance, the declaration of a communication system model of off-line devices with immediate-access memory blocks.

During the logical stage of design the ALOS [2] language is employed and the language for declaring a computer structure. The ALOS language is intended to declare a computer operation algorithm or, to be more exact, that part of a computer which consists of a control unit and of an arithmetic unit. The language for declaring a computer structure contains facilities for declaring it as a composition of other simpler devices. Appendix 2 exemplifies the declaration in the ALOS of the execution algorithm of the device and its structure. The technical stage of design deals with the known language of wiring tables.

Optimization in Design

Optimization problems arise during all stages of design. Although the statement of problem of constructing the optimal project in a general form is feasible, the problem solution with the present-day state of mathematical tool used is almost impossible. Therefore, in practice of design the general problem of project global optimization is divided into a sequence of problems allowing to solve optimization problems locally, within one stage.

Let us briefly outline problems connected with finding

criteria and ways of optimization. This is associated, first of all, with the possibility of accomplishing formal transformations over expressions in some algebra with preserving certain invariants but with altering some parameters of expressions. Obviously, in its turn, if on a certain stage a design problem is difficult and a complete optimization is impossible, a partial optimization is done, or a computer under design is simulated and man-simulating computer joint operation is organized permitting to quickly evaluate a number of versions, to effect a formal transformation of the model and to obtain better project features on the given stage.

It is known that the purpose of a computer as a device for processing discrete information is to transform input sequences of discrete signals, say, letters, into output sequences. A computer maps a set of input sequences of signals into a set of output sequences of signals. This is a function analog. There exist, ordinarily, certain construction restrictions imposed on the function realization. A mapping taken at random is far from being easily realized in a computer immediately during one step. It is necessary to divide this mapping in order to have a composition of more minute mappings. Selecting by some method more minute mappings as separate elements we can obtain different compositions.

During the system stage of design the invariant is a mapping. This means that the mapping should not change. In transforming on this stage the mapping realization is subject to alteration, i.e. the way of its decomposition into elementary constituent parts. There are many such ways.

During the next stage the invariant is not only the mapping itself (stated by a computer) but also the way of its decomposition into constituent parts. On this level the designer deals with

automata. The main theoretical base in solving problems of this stage is the automata theory. An automaton has a set of internal states. These states change at discrete time instants. There exists a system of output signals. The next state of automata - a function of the previous state and of an input signal, an output signal - a function of the same arguments. These functions are said to be transition and output functions. They are very objects for transformations on this stage of design.

In further design the partial automata having a single inner state are treated. In this case the invariant is an output function of a certain automaton. Formal transformations change the method of realizing these functions. Finally, during the technical stage of design on the level of hardware described simply by systems of numbers - coordinates (describing position of a point on a plane - position of nodes of corresponding circuits) the invariants are: topology (graph) of connection of nodes first and, second - distance between some nodes. Transformations change coordinates of nodes. One of criteria of efficiency may be the attainment of the minimum length of all coupling wires. This criterium depends on a type of elements chosen.

Optimization questions are complex on all the stages. In particular, on the stage of technical design. They reduce to mathematical problems of finding a maximum and minimum of functions of a very large number of arguments and oftentimes are not confined simply to problems of linear programming. Parameters are numbered in many tens or hundreds. Therefore the solution of such problems is difficult and requires rather extensive programs.

Note that the tool for formal transformations on certain levels has so far been developed not good enough. For instance, no formal transformations are immediately performed on the system

stage of design. In part the problem of formal transformations is solved within the theory of large-scale systems. We cannot manage to use known regularities of the queueing theory, since the general problem cannot be solved by analytical methods and goes beyond the scope of the classical queueing theory. The greatest difficulty consists in that there is a feedback of a device for processing data with an input flow of requests. The flow of requests in ordinary classical queueing problems is not dependent on servicing system. In our case, however, characteristics of the flow of requests are changed depending on the state of the data processing system. In addition, in servicing, much more complex transformations of requests of the input flow are formed or simply included. The flows proper are non-uniform and requests bring a large number of parameters which may influence a process of the subsequent treatment. Meanwhile the design automation system can cover a number of discrete problems allowing a theoretical solution. For this purpose provision is made in the language to have a means for declaring standard flows in queueing systems, and corresponding mathematical transformations.

As to the logical stage of design, there was essentially no, until recently, approach to solving complex problems of design automation. Languages have come into being but they almost have not had formal transformations. The Yu.I. Yanov's paper has probably been the first in this direction. He has constructed the theory of formal transformations of schemes of programs. At present this paper is fairly popular among those who are concerned with optimization problems. However the paper has solved the problem of transforming the algorithm scheme only, and not the problem of transforming the algorithm itself.

The difficulty of the last problem consists in that at the very outset of the theory of algorithms the algorithmic unsolvability of the problem of establishing the equivalence between two algorithms by means of formal transformations was proved. This circumstance scared away investigators from construction of such formal systems for transforming algorithms. Lately it became possible to solve the problem of building the mathematical tool for transforming a special kind of algorithms - microprograms which may be applied in many practically important cases.

We shall now discuss in brief the basic scientific concepts utilized in this theory. Any algorithm essentially is a method of stating the transformation of a set. To construct transformations, conditions are taken specified over this set. Using a set of transformations (operators) we can build a system of operations (similarly to algebra) - multiplication of operators, α -disjunction and α -iteration.

Let M be a set of states of automaton $\mathcal{B}$, $a \in M$, $\mathfrak{A}$ - set of operators acting on set M, $\mathfrak{P}$ - set of conditions determined over set M. Operations of α -disjunction and α -iteration are determined as follows: α -disjunction of operators $\mathcal{P}$ and Q is operator R which transforms element $a \in M$ into $\mathcal{P}(a)$ if condition $\alpha(a)$ holds and operator $\mathcal{P}$ is applicable to a, and into $Q(a)$ if condition $\alpha(a)$ is false and Q is applicable to a. In the rest of cases R is undetermined.

Operation of α -iteration is determined as follows: α -iteration of operator $\mathcal{P}$ is operator R which transforms element $a \in M$ into element $\mathcal{P}^n(a)$ if conditions $\alpha(a)$, $\alpha(\mathcal{P}(a))$, ..., $\alpha(\mathcal{P}^{n-1}(a))$ are not satisfied and $\alpha(\mathcal{P}^n(a))$ is satisfied, otherwise R is undetermined.

Such operations have already been employed earlier in various logical languages. The new (in comparison with classical) operation of multiplying the operator by the logical condition is introduced. The result of multiplying operator φ by condition α is new condition β such that $\beta(a)$ holds iff $\alpha(\varphi(a))$ holds.

Selected in a set of operators are some operators named elementary ones or microoperations. Moreover some logical conditions are fixed. These operators and conditions generate microprograms algebra with the aid of which one can state algorithms realized by units of a computer being designed. The distinction from the classical theory of algorithms consists in that the algorithms are considered to be elements of some algebra and relations of a certain algebra may be used to transform the algorithms. Relations in the algebra of microprograms are much more complex and their number is much greater.

Appendix 3 of this paper illustrates the formal transformation of a microprogram for multiplying two integers in a certain algebra.

The experience of using the idea of transforming algorithms as expressions in microprogram algebras testifies that this means is rather effective and allows to formalize the transformation of structure of a computer to be designed on the whole during the logical stage.

Organization of Automated Design System

The following problems are to be solved in organizing an automated design system:

- Automatic storage of information on the project during the all stages of design.

- Organization of a joint designer-system operation.

- Availability of means to introduce changes in the project and system.

We shall now briefly describe the organization of the automated design system which is being developed at the Institute of Cybernetics in Kiev. This system has means for:

- Declaring information of a computer under design during various design stages.

- Solving all design problems during the logical and technical stages of design.

- Solving a number of optimization problems in the process of design.

- Completing documents on a computer being designed.

- Declaring and including into a system new algorithms for transforming information of a computer under design.

- Ensuring an immediate introduction of changes into the project.

- Providing a convenient user-system intercourse.

From the user's viewpoint the system is imagined to be consisting of the following components: Main information file or a data file on a computer to be designed, standard block file or standard file, standard program file, direction file, and system control.

In the process of intercourse with the system the user should know languages for declaring information on a computer under design, the language of intercourse with the system, and the design methods used in the system.

Data File

In the system the data file is intended for storing information on a computer being designed. The following languages are taken in

the system for declaring the object of design: Language "Algorithm" (ALOS) containing facilities required to declare execution algorithms of devices, language "Structure" serving to declare structures of devices, and language "Construction" allowing to declare arrangements of devices.

The data file consists of objects each of which has a heading. Objects may be simple and composite. The heading contains object name, type and constituent parts if the object is composite. The heading of a simple object has its name and type.

A simple object is a declaration of construction in languages suitable for representing information on a computer to be designed.

There corresponds to each object in the data file the data file address indicating the place in the data file from which the object heading is originated.

The correspondence between the data file address and the object name is the contents of the data file table.

Project information inserted into the system, or obtained in the process of designing, arrives at the data file via a special program - data file monitor which also controls information delivery from the data file.

Standard File

In the file system of standard blocks the standard file is intended to store information on standard devices used in the process of design. In particular, the standard file stores information on a system of components employed to design a computer.

The declaration of standard devices obey the same laws as the declaration of a device under design. Information in the standard file is organized similarly to the case of the data file. A distinctive feature of the standard file with respect to the data file

is constancy in the process of designing.

The standard file has the standard file table and any access to the standard file is gained through its monitor.

Standard Program File

The program file in the system is aimed at storing declarations of programs to design and maintain the system.

Use is made in the system of three levels of the language in which programs are declared. The first level - instruction language of a computer which the system is realised on. The second level - basic language of an autocode type for a certain computer. The third level - operator part of the language of the user-system intercourse. The user may utilize any of these languages and has to know the third level without fail.

All the programs in the program file fall into three parts according to the languages. The first part of the program file contains declarations of programs in computer codes. A declaration of each program begins with the program heading containing program name, type and formal parameters.

Access to standard programs of the first part of the program file in the intercourse language is completed in the form of simple instructions.

The second part of the program file contains declarations of programs in the basic language.

The third part of the program file provides declarations of complex instructions of the intercourse language, and represents a table for deciphering complex instructions through simple ones.

Access to programs of the second part of the program file is gained in the intercourse language via the instruction of making access to the programming automation system the declaration of

which is stored in the first part of the program file.

Access to programs of the third part of the program file is gained through a system manager.

All loading to and extracting from the program file are effected via the program file monitor through use of the program file table.

Instruction File

The instruction file is used to store sequences of particular external instructions with the aid of which the user will conduct the design process in the system.

Access to the instruction file is gained via the system manager.

A declaration of each instruction in the instruction file consists of an instruction and a list of actual parameters of the instruction.

System Control

The system control is employed to execute user's instructions in designing.

The system control consists of the system manager and service information for the system manager.

The instruction file is source information for the system manager.

A time step of the system operation consists of executing one external instruction from the instruction file.

The execution of an external instruction by the manager consists of two parts : Preparation for execution an instruction and its execution itself.

When preparing the execution of a simple instruction the system manager operates as an interpative system which uses

auxiliary information and represents a given external instruction as a sequence of internal instructions.

Programs of the first part of the program file correspond to internal instructions.

The execution of the external instruction consists in execution of each internal instruction of a given sequence.

To execute the internal instruction the manager should call to the system operating board the program corresponding to a given internal instruction and transfer the control to it.

Computer Representation of Information

Information on the device to be designed arrives at the system in the form of texts in languages specially intended for the purpose ("Algorithm" (ALOS) and "Structure") and is written in the data file ~~as a list~~ as a list structure. Its character and coding has been developed. Information in the standard block file is represented similarly to the case of the data file.

The general organization of files has been worked out based on the three-level division of memory (drums, tapes) and questions using the principle of page and book organization have been considered to exchange and distribute the computer memory in the system.

All the decisions on organization, representation and coding of information on data and on the system have been made taking into account features of the computer M-220.

Appendix I

Model of System Communicating Autonomous Devices with Immediate-Access Memory Blocks

With the use of the SLANG-language a declaration is given of operation of the system consisting of file BANK [k] of k blocks of immediate-access memory and of M autonomous devices. Each device sends inquiries with its particular period PERIOD[M] to the immediate-access memory, the inquiries being answered through common buses BUS. Quantities TB, TWR and TO characterize operation time of the common buses, time of making access to memory and memory cycle time.

The process REGHAN describes generation of inquiries of autonomous devices. At first, in the cycle between tag FN: and operator GOTO FN , M initial inquiries are formed: one from each autonomous device, and one from memory blocks (randomly) selected with the aid of CHOICE(I,k). Then, in the cycle between tag E1: and operator NEW ... TO F1 , the devices generate inquiries to other memory blocks, and go through (each with its period) numbers of blocks in the direction of a modulo k increase by 1. The generation of an inquiry is mapped by operator NEW BANKS(BN,PR) the actual parameters of which mean the number of interrogated block and the number of priority of an interrogating device, respectively.

Quantity COUNT is a counter of the general number of inquiries to the memory, and the simulation is ended when it reaches value BOUND (multiplied by k) stated at the input.

Process BANKS describes answering the inquiries to the memory through the common buses and memory blocks.

Model of System Communicating Autonomous Devices with Immediate-Access Memory Blocks

```
BEGIN  INTEGER K,M,TP,TWR,TQ,COUNT,BOUND,I
FACILITY BANK(K),BUS;
STATISTIC BANK;
INTEGER PERIOD(M);
TABLE STAB(M)(0 STEP 5 UNTIL 50);

INPUT (K,M,TP,TWR,TQ,BOUND,PERIOD);


PROCESS RECCHAN;
BEGIN  INTEGER BN,PR;
F:=M;COUNT:=0;
BN:PR:=RANDINT(1,K,I),
F:=F-1;
 IF F=0 THEN  NEW RECCHAN TO FN;
F1: NEW BANKS(BN,PR) TO Q;
 WAIT PERIOD[PR];
BN:= IF BN=K THEN BN+1 ELSE 1;

IF COUNT=K*BOUND THEN  STOP ;
 COUNT:=COUNT+1;
 GOTO F1
 END ;


 PROCESS BANKS(B,L); INTEGER B,L;
 BEGIN  INTEGER T;
 CANCEL ;
Q:T:=TIME;PRIORITY:=L;
M1: WAITUNTIL BUS NOTBUSY ;
 IF BANK[B] BUSY  THEN
 BEGIN  WAITUNTIL BANK[B] NOTBUSY ;
 GOTO M1
 END ;
 IF BUS BUSY  THEN  GOTO M1;
 SEIZE BUS;
 SEIZE BANK[B];
 WAIT TP;
 RELEASE BUS;
 WAIT TWR;
 TABULATE TIME-T-.5 IN STAB[L];
 WAIT TQ;
 RELEASE BANK[B];
 CANCEL ;
 END
 END
```

As soon as an inquiry arrives, an instant is marked of the beginning of waiting for an answer. Upon completion of answering (i.e. access time to memory), an increment of answering time is calculated in comparison with a least wait which equals 55 units.

The value of this increment is entered the table formed for each device (ATAB[L], where L=1,2,...,M).

Before an inquiry is loaded into the buses and memory block (by operators seize BUS ; seize BANK [B]) the analysis is performed whether or not the buses and interrogated memory block are occupied with the priority number of the interrogating device taken account of. This analysis is performed through use of operators wait until and if .. then on the section between tag M1 : and operator goto M1 where all waiting inquiries circulate.

Each inquiry satisfying the conditions occupies the common buses for time TB (specified by operator wait TB) and the memory block for time TB+TWR+TQ, the time of answering the inquiry satisfying the conditions being TB+TWR.

Appendix 2

Example of Declaration of Algorithm for Device Operation in ALOS-Language

```
D e v i c e A
    variables
        input x ; y ;
        internal r; s; t;
        output z ; q ;
    functions
        f(a,b) ; g(a) ; A(a) ; B(a) ; C(a,b) .
```

```
subprogram S(a,b)
S1: if B(a) then a:=x else go to S2
    if C(a,b) then b:=f(r,x) else (a:=g(s); go to S1)
S2: T(y)
end of subprogram S
subprogram T
T1: if B(a) then (t:=f(r,s); stop) else r:=g(a)
                 (q:=g(a); go to T1)
end of subprogram T
begin
P1: if A(y) then (r:=x; S(s,t)) else
                 (r:=y; go to P3).
P2: T(x); go to P4.
P3: if B(s) then (s:=y; T(r); go to P1) else
                 (s:=x; S(t,r); go to P2).
P4: if C(r,s) then (z:=f(x,y); go to P1) else
                   (q:=g(r); go to P2)
end.
```

Example of Declaration of Device A Structure

```
D e v i c e  A
    variables
        input  x; y.
        output z; q.
    components
    P; variables input  y; r; w; ω.
                output  u; v.
```

S; variables input $r; t; v; w'$
output $v'; \bar{u}; w$.
T; variables input $x; y; r; v'; v$.
output $\bar{u}; \omega; w'$.
r; variables input $x; y; u; \bar{u}; \bar{u}'$.
output r.
s; variables input $x; y; u; \bar{u}$.
output s.
t; variables input $x; r; s; \bar{u}'; \bar{u}$.
output t.
Q; variables input , $y; r; \bar{u}'; u$.
output q.
z; variables input $x; y; u; s$.
output z.

Commentary

Device A the operation algorithm of which is declared in the ALOS-language is represented in a form of an 8-component structure.

P, S, T - control blocks
r, s, t - operation blocks
Q, z - functional blocks

The language employed for declaring the structures has a tool for the explicit recording of connections among the components. In the given example this is made implicitly through notation of input and output variables of the components.

Appendix 3

Example of Processings of Microprograms

Let $(x_1, \ldots, x_u)$ be a set of binary registers the states of which are rational numbers written in the binary scale of notation.

Consider some microoperations and conditions used in recording the algorithm of a computer being designed

$$l_i : x_i := 2x_i \quad (i=1,2,\dots,n)$$
$$r_i : x_i := \frac{1}{2}x_i \quad (i=1,2,\dots,n)$$
$$s_{ij} : x_j := x_i + x_j \quad (i,j=1,2,\dots,n)$$
$$p_i : x_i := x_i + 1 \quad (i=1,2,\dots,n)$$
$$p_i^{-1} : x_i := x_i - 1 \quad (i=1,2,\dots,n)$$
$$0_i : x_i := 0 \quad (i=1,2,\dots,n)$$

e – identical transformation

$$\alpha_i = \begin{cases} \text{true}, & x_i = 0 \\ \text{false}, & x_i \neq 0 \end{cases} \quad (i=1,2,\dots,n)$$

$$\beta_i = \begin{cases} \text{true}, & x_i = 2k, \ k \text{ – integer} \\ \text{false} & \text{otherwise} \end{cases} \quad (i=1,2,\dots,n)$$

We shall examplify relations

$$l_i s_{ij} = s_{ij}^2 l_i \quad (i=1,2,\dots,n)$$
$$z_i 0_i = 0_i \ ; \ (z_i : x_i := f(x_i), \ i=1,2,\dots,n)$$
$$r_i p_i^{-1} = p_i^{-1} r_i \quad (i=1,2,\dots,n)$$
$$l_i \alpha_i = r_i \alpha_i = \alpha_i \quad (i=1,2,\dots,n)$$
$$y_i z_j = z_j y_i \quad (y_i : x_i := f(x_i); \ z_j : x_j := f(x_j); \ i \neq j)$$

$$\{(\nu p_i^{-1} z)^2\}\alpha_i = \beta_i \text{ where } \mu = \alpha_i \vee p_i^{-1}\alpha_i \text{ and } y\alpha_i = z\alpha_i = \alpha_i \quad (i=1,2,\dots,n)$$

$$\overset{\mu}{\{\mathcal{P}\}} = (\underset{\nu}{e} \vee \mathcal{P})\{\mathcal{P}^2\} \text{ where } \mathcal{P} \in \mathfrak{A}, \ \mu \in \mathfrak{B}; \ \nu = \{\mathcal{P}^2\}\mu$$

$$\{\mathcal{P}\}\underset{\mu}{Z} = Z\{\mathcal{Q}\} \text{ where } \mathcal{P}, \mathcal{Q}, Z \in \mathfrak{A}, \ \mathcal{P}Z = Z\mathcal{Q}^{\mu \vee \mathcal{P}\mu} \text{ and } Z\mu = \mu$$

Transformation of the multiplication algorithm of two integers. Let x_1, x_2, x_3 be integers. The initial record of this algorithm corresponds to the definition of the multiplication algorithm as a consecutive addition. It turns out to be possible to transform this record into such one which corresponds to the multiplication algorithm usually realized in a computer.

$$Q_6 : \; x_2 := x_1 \times x_3 \; ; \; x_1 := 0 \; ; \; x_3 := 0$$

$$Q = Q_2 \{ s_{12} P_3^{-1} \} Q_1 Q_3 =$$

$$= Q_2 \{ s_{12} P_3^{-1} \}_{\alpha_3} \ell_1 \Gamma_3 Q_1 Q_3 =$$

$$= Q_2 (e_V s_{12} P_3^{-1})_{\beta_3} \{ s_{12}^2 P_3^{-2} \}_{\alpha_3} \ell_1 \Gamma_3 Q_1 Q_3 =$$

$$= Q_2 (e_V s_{12} P_3^{-1})_{\beta_3} \ell_1 \Gamma_3 \{ s_{12} P_3^{-1} \}_{\alpha_3} Q_1 Q_3 =$$

$$= Q_2 ((e_V s_{12} P_3^{-1})_{\beta_3} \ell_1 \Gamma_3)^K \{ s_{12} P_3^{-1} \}_{\alpha_3} Q_1 Q_3 =$$

$$= Q_2 \{ (e_V s_{12} P_3^{-1})_{\beta_3} \ell_1 \Gamma_3 \}_{\alpha_3} Q_1 Q_3$$

References

1. Kalynichenko, L.A. (1967): SLANG– Experimental Programming Language Faced Description and Simulation of Computers and Systems, Col. "Teoriya Avtomatov", No.1, Kiev.

2. Glushkov, V.M., Letichevsky, A.A. (1967): Language for Describing Algorithmic Structures of Computers and Systems, Col. "Teoriya Avtomatov", No.2, Kiev.

3. Glushkov, V.M. (1967): Prospects of Computer Design Automaton, "Vestnik Akademii Nauk SSSR, No.4, Moscow.

4. Glushkov, V.M., Kapitonova, Yu.V., Letichevsky (1967): Toward Automation of Computer Design, Journ. "Kibernetica", No.5.