Automatic Control & Computer Sciences
Vol. 22, No. 6, pp. 59-67 (1988).

HARDWARE IMPLEMENTATION OF PROTOCOLS FOR A FAULT-TOLERANT SELF-SYNCHRONOUS RING CHANNEL

V. I. Varshavskii, V. Ya. Volodarskii, V. B. Marakhovskii,
L. Ya. Rozenblyum, Yu. S. Tatarinov, and A. V. Yakovlev

Avtomatika i Vychislitel'naya Tekhnika,
Vol. 22, No. 6, pp. 60-69, 1988

UDC 681.327:519.713

The principles is design procedures for a channel adapter circuit in a system interface of a ring LAN are discussed. Methods of self-synchronous implementation of a communication protocol are described, including a concept of direct transmission of the protocol into the control circuit operating independently of delays of communication channel elements and lines.

An information exchange protocol, which is a set of rules of joint operation of modules in a network architecture, can also be used as the initial specification of the behavior of adapters which link the modules to the information channel. In [1] a concept of direct translation of physical layer protocol into the interface adapter circuit was suggested. The concept employed signal graphs as the formal language of protocol specification. It has been mentioned, however, that the method is equally applicable to other, more important model classes which describe not only asynchronous and parallel operations, but also the dependence of process dynamics on the values of the data received or generated. In [2] a structural organization was described of a channel with a ring architecture (Fig. 2 in [2]) and communication protocols which provided fault-tolerant self-synchronous interface at the physical layer of a local area network. In this paper we describe the principles and procedures of construction of a channel adapter (CA) circuit implementing these protocols in a way invariant with respect to delays of elements and lines of the communication channel.

1. PROTOCOL AUTOMATON AND CONTROL AND RECOVERY AUTOMATON

Generally, CA is a circuit which, besides a set of protocol functions including capturing the channel and transmitting and receiving messages, is responsible for malfunction, detection, localization, and self-recovery so as to maintain the channel at a desired fault-tolerance level. It is convenient to divide the CA structure into two substructures (Fig. 1): protocol automaton (PA) and control and recovery automaton (CRA). The PA captures the channel and transmits and receives messages operating as the driver and the receiver. PA also forms malfunction signals of both message buffers (FIFO) and a number of auxiliary indication signals for CRA.

CRA diagnoses and localizes malfunctions on buses linking CA with the communication channel and identifies CA malfunctions; it also switches and reswitches the communication lines of the SB (source data bus) and RB (receiver data bus) with internal signal lines of the buses transmitting a half-byte in a self-synchronized code (SCC) such as an optimal equilibrium code (OEC) C_6^3 DIB (1:6) and DPV (1:6), and the respective acknowledgments SIV and SPV; in doing so, if necessary, it makes use of backup lines RI and RP, which are included in SB and RB (the functions of all links of the CA have been described in [1]).

In a sense, CRA is a PA operating environment, which insulates PA from all "bread-and-butter" problems that may spring up from unreliable channel surroundings (the "transparency" principle). This methodology is the hardware analogue of an approach described in [3], based on the idea of an autonomy of atomic actions of programming processes and a localization of all recovery functions in so-called exception handlers.

In the present paper we will discuss only the PA design. The development of CRA has

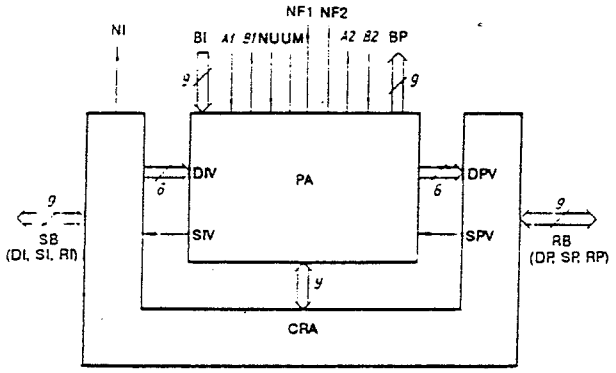

Fig. 1. Flowchart of ring channel adapter.

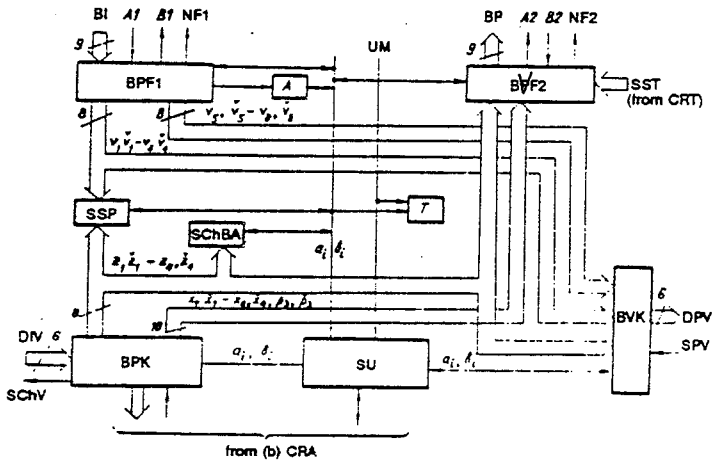

Fig. 2. Flowchart of protocol automaton.

certain specific features associated with the design of internal protocols for exchange of diagnostic and recovery information and is the subject for a separate paper.

A discussion of PA circuit synthesis must cover essential aspects, including development of the operating structure for PA and the choice of the control principle for operating structure elements; determination of the types of main operations and the choice of an adequate language for specification of the asynchronous control of PA structure elements on the basis of communication protocols specified by [1]; and development of methods and means to support translation of a process description into a PA control circuit.

2. THE OPERATING STRUCTURE OF A PROTOCOL AUTOMATON

The structure of a PA consists of the following blocks (see Fig. 2): block receiving information from the channel (BPK), block receiving information from FIFO1 (BPF1), block sending information into FIFO2 (BVF2), block sending information into the channel (BVK); set of auxiliary blocks: an arbitrator (A), counter of address bytes (SChBA), priority comparison circuit (SSP), and "I'm the driver" feature trigger (T); and control circuit (SU).

All the blocks, except for SU, belong to the operating component of PA and are started from SU by special command signals of request a_i; after the appropriate operation is completed, they respond by the appropriate response signals b_i. In this way a self-synchronous ("question-answer") protocol is maintained at the level of PA-internal interblock communications.

Several blocks in the operating component are implemented on the basis of existing

aperiodic circuit-engineering concepts. For example, the "SSK ORK C_6^3 - paraphase positional code" converter, included in this system in BPK, has been described in [4].

For lack of space we cannot describe in detail the structure of each block; we will merely outline their functions.

BPK receives from the input channel a half-byte in SSK ROK C_6^3 and converts it into a paraphase code of internal representation of half-byte (16 code sets) of information and into unitary indication signals for each of the four markers, M1-M4 (altogether, ORK C_6^3 provides 20 working ensembles); this is done for forming the data byte and sending it into BVF2 and BVK (it is sent into BVK by half-bytes), sending the half-byte of the priority number into the comparison circuit, sending the address length half-bytes into SChBA counter, and executing half-byte shift operation after receiving an address or the last word according to the protocols of [1].

BPF1 receives from FIFO1 and stores temporarily a message byte, produces pairs of half-bytes into BVK and priority numbers into SSP, increments priority in the "receiver-competitor" mode (according to capture protocol [1] the model which has received a request for message transmission after the current arbitration section and has not received access as the channel driver is allowed by one-half by the next arbitration session to increment its priority), sends the channel capture request signal into the arbitration unit, indicates error in case of reception from FIFO1 and sends indication signal NF1 to Q-bus interface controller (QK).

BVF2 sends into FIFO2 an information byte formed of two half-bytes stored in BVK, or a byte of the status word received from AKV, or the last-word whose ninth bit (B9) equal to 1; it also detects transmission errors in FIFO2 and sends indication signals NF2 to QK.

BVK transmits into the output channel sequentially information half-bytes formed from the byte received from FIFO1, or half-byte from BPK, or one of the markers M1-M4, with the coding of half-bytes and markers in ORK C_6^{3-}.

Auxiliary blocks A, SSP, and T perform ancillary functions. The arbitrator performs local arbitration between the demand signal received from SU and formed on the basis of a request marker received from the channel capture request feature formed in BPF1 as a result of reception of the first byte of a message placed "at the exit" of FIO1. The arbitrator contains a fixing trigger of the feature RZ indicating the presence of a capture demand; it is used by SU at various steps in the control process.

SSP compares priorities: the "own" priority number of the module N_c formed in BPF1 and the priority number N received from the channel arriving from BPK; as a result, predicates $N_c \geq N$ and $N_c \leq N$ are formed to determine the choice of the control process path in SU.

The trigger of the "I'm the driver" feature is used to store the tag indicating successful capture of the channel by the driver.

Besides the above functions of the blocks we should mention the function of storing in BPK the feature of selecting the receiver RBB (according to its address). The error FIFO signal formation mechanism in BPF1 and BPF2 is based on the concept of detecting the time at which the critical value of the circuit time-out (the scale delay) is exceeded during the formation of the current signal of request A1 (from FIFO1) and response V2 (from FIFO2).

3. THE PRINCIPLE OF TWO-LAYER CONTROL OF OPERATION BLOCKS IN THE PROTOCOL AUTOMATON

In the PA circuit the organization of the structure controlling the operating modules is closely connected with several design and functional parameters, such as operation speed, equipment size, length of wire connections, etc. Two control levels are combined in PA to obtain the desired values: one centralized and one decentralized. We proceed from the notion that a protocol description translated into a circuit is in face the specification of the upper (centralized) control level which coordinates the above blocks by executing the commands of the PA's general control circuit. This assumes that the control circuit executes the entire protocol algorithm. On the other hand, an analysis of protocol functions executed by BPK, such as address reception and reading, reception of a pair of consecutive half-bytes from the input channel and sending either a data byte

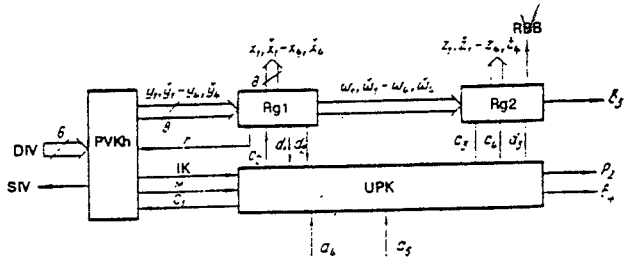

Fig. 3. Flowchart of the unit receiving information from the channel.

into FIFO2 or a pair of half-bytes into BVK (possibly with a shift, if it is an address or the last word), indicates that for effective operation of BPK constructed on the basis of an asynchronous buffer with two registers and an input converter SSK C_6^3, converting data to a paraphase internal code, it is desirable to have the control inside BPK to be decentralized and made independent of the general SU. Partially, it is organized according to the "data stream" principle. BPF1 is a nine-bit register; the four lowest bits consitute the priority counter. The other two blocks BVF2 and BVK do not store (buffer) half-bytes or bytes and perform multiplexing of information flows. The immediate control of the registers is done by local control modules which produce, upon receiving commands a_i from SU, local control signals C_j. Local indication circuits (in accordance with the self-synchronous coordination principle [4]) which place working and separation sets on information buses (outputs of registers or multiplexers) generate indication signals d_k. These signals cause local control modules to produce responses b_l which serve as acknowledgments of individual protocol operation.

The organization of the lower control level can be illustrated by the BPK structure in Fig. 3; it combines the principle of distributed control (coordinated interaction of the first register Rg1 with input set converter PVKh) and centralized control (by signals from the circuit controlling data reception from UPK channel). The coordination of the former type is executed on the following pairs of information and acknowledgment signals: DIV, SIV, (y_1, y_2), and r, which implement a standard question-answer protocol. The centralized control is carried out in response to the command C1-C4 produced from UPK into PVKh, Rg1, and Rg2, respectively. The semantics of C_j is fairly transparent: C1 allows PVKh to perform the reception of the next set SSK C_6^3 from DIV channel and convert it either into informational paraphase code of half-bytes (IK) or into unitary marker code (M); C2 permits reception of information in Rg1 from PVKh; and C3 and C4 require writing a half-byte into Rg2 from Rg1 without a shift or with a shift, respectively (according to protocol of reception of an address or the least word, respectively).

The commands C_j produced by UPK are a result of transformation of central control commands a_4 and a_6 arriving from SU. In particular, a_4 specifies the command "receive from the channel an information half-byte or a marker"; a_6 specifies the command "receive from the channel either address half-byte or last work half-byte with a shift toward higher bits or a marker." BPK responds to these commands in two possible ways: by a response signal b_3 indicating that it "has received a new half-byte of address, or last word, or priority number, or data half-byte," or the response signal b_4 indicating that it "has received a token." In addition, BPK forms predicate signals R2 (byte readiness feature) and RVB (receiver selection feature) which are used by SU to select the direction for the protocol algorithm.

4. DEFINITION OF TYPICAL OPERATIONS OF A PROTOCOL ALGORITHM

Proceeding to the actual design of a central control circuit (SU), the initial functional protocol specifications described in [1] in the form of marked Petri nets should first be converted to a representation describing exactly the control process by the block of operational structures selected in terms of a certain finite set of typical operations on such blocks.

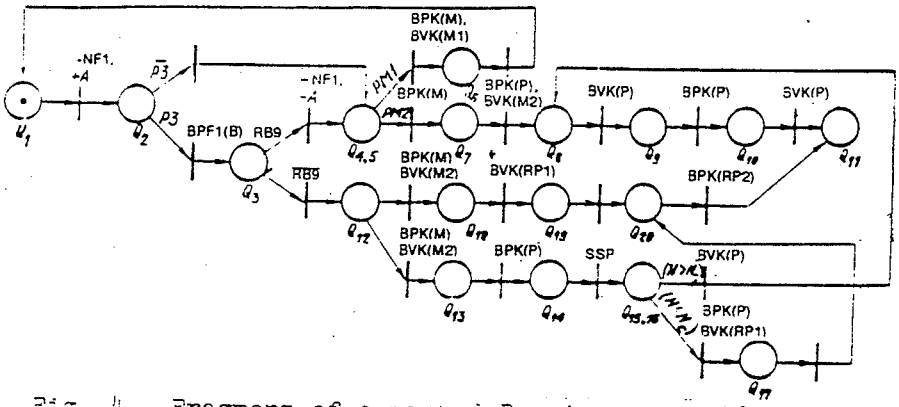

Fig. 4. Fragment of a marked Petri net specifying the algorithm of the control circuit operation.

Typical operations are to be understood as formal actions representing those functions of operation blocks described informally in section 2. A list of typical operations performed on blocks of PA operation structure is given below.

For BPK:

BPK(M) - reception of a token (one of four M1-M4) from the channel;

BPK(B) - reception of information half-byte from the channel;

BPK(S) - reception of information half-byte with a shift toward higher bits;

BPK(S, M) - execution of either BPK(S) or BPK(M) depending on the value of the code C_6^3 received;

BPK(P, M) - execution of either BPK(P) or BPK(M) depending on the value of the code C_6^3 received.

For BVK:

BVK(M_i) - transmission into channel of the token M_i (i = 1-4);

BVK(P) - forwarding into channel the half-byte received from channel;

BVK(P_i) - forwarding into channel half-byte isolated in the byte received from FIFO1 (if i = 1, the lower half-byte is produced; if i = 2, the higher half-byte is produced);

BVK(Ø) - transmission of the zero half-byte into the channel (to form the last word).

For BPF1:

BPF1(B) - reception of information byte from FIFO1;

BPF(+) - incrementing the priority level by one-half;

BPF1(NF1) and BPF1(NF1) - setting and clearing error signal of FIFO1 NF1, respectively.

For BVF2:

BVF2(SS) - transmission of status word into FIFO2;

BVF2(BI) - transmission of information byte into FIFO2;

BVF2(KF) - transmission of last word byte into FIFO2.

For the other blocks:

SSP - starting priority comparison circuit;

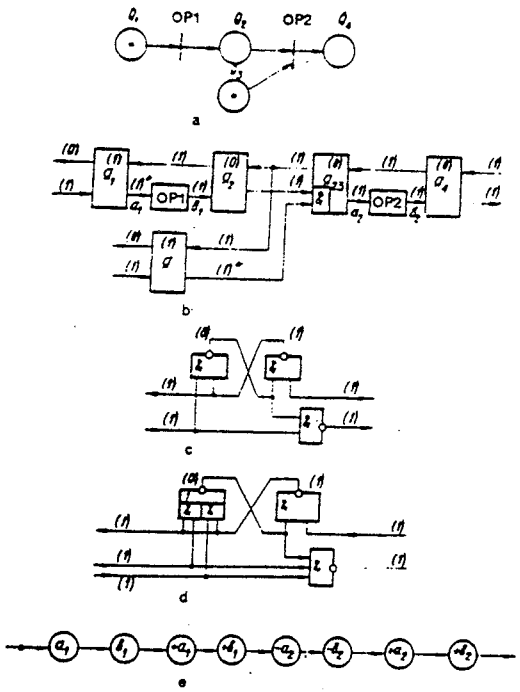

Fig. 5. An illustration of the principle of translation of control into a circuit on the basis of a phase-distributed implementation: a) a fragment of assignment represented by a marked Petri net; b) a fragment of the corresponding control circuit consisting of asynchronous distribution cells; c, d) implementations of two cells; e) signal graph specifying the sequence of the change of control signal types by operations OP1 and OP2.

+A - interrogation of arbitrator (local arbitration) and setting RZ (feature indicating the presence of a request for channel capture);

-A - clearing RZ to 0;

+T - setting feature-trigger "I am the driver" to 1;

-T - clearing the trigger T to 0.

The following predicates features are used to select the direction of the protocol algorithm during the course of control by SU: RM_i, flat indicating placement of token M_i at BPK input; RD, flag of a request for channel capture (set during arbitration if a request is present); R2, flag of presence of two half-bytes in the registers Rg1 and Rg2 of BPK (generated by BPK to coordinate the mode of reception of a whole information byte and its transmission into FIFO2); RVB, flag of selection of the receiver (after addressing the receivers, the device operates in the mode of selected receiver or an unselected receiver) generated by BPK; RB9, flag indicating that the ninth bit of the byte received from FIFO1 is on (set y BPF1); RT, flat "I am the drivier" (set by the operation +T).

Once the set of operations executed by the operating blocks is specified, the formal language of control description is chosen; the action symbols in the language represent the operations to be executed, and the condition symbols give the values of the features generated in course of execution of the operations. The language should provide a possibility of formal transition from a specification of a parallel algorithm directly to a control circuit; as much as possible, this should occur by a fragment-by-fragment translation (process of a linear complexity), which ensures automatic validity by construction, i.e., obtaining a control circuit whose behavior is independent of delays in its components. Among possible choices of notation are marked Petri nets (MPN) and parallel asynchronous graph charts (PAGC).

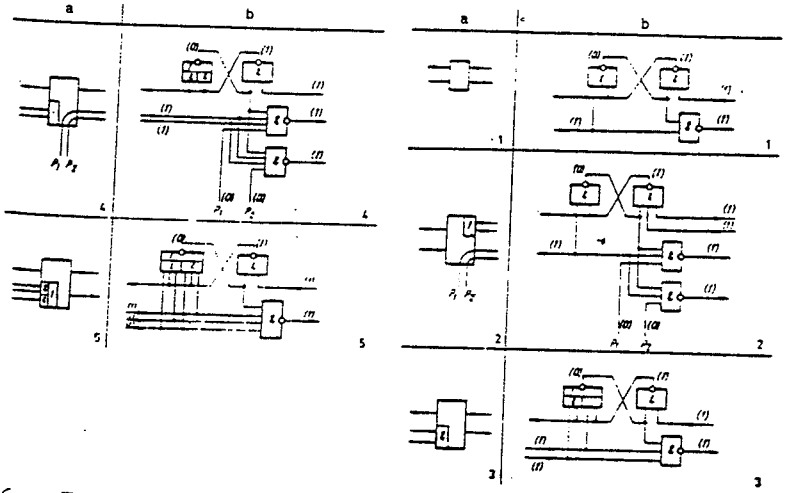

Fig. 6. Examples of types of elements of DAC library: a) cell type; b) cell circuit.

In this paper preference is given to MPN notation used earlier in [1]; an illustration is given in Fig. 4, which represents a component of an implementation of channel capture protocol. Transitions (event nodes) of MPN are interpreted in terms of typical control operations. The arcs emanating from conditions (position nodes) in MPN can be interpreted in terms of the features listed above. This MPN, in essence, describes in detail the transitions incident to zero position in MPN, as shown in Fig. 3 in [1].

5. TRANSLATION OF PROTOCOL AUTOMATON CONTROL ALGORITHM INTO A SELF-SYNCHRONOUS CONTROL CIRCUIT

The mechanism translating MPN (see Fig. 4) into a self-synchronous control circuit generating command signals A_i and receiving acknowledgment signals b_i is based on a further elaboration of the algorithm, specifically on an application of a signal interpretation: 1) MPN positions are interpreted in terms of memory elements storing the MPN state; 2) arcs with possible predicates are interpreted in terms of feature signals; and 3) transitions denoted by operations of the operational structure blocks are interpreted in terms of request signals a_i and acknowledgment signals b_i.

A method modeling Petri nets as circuits on the basis of conditions described in [4] can be used for this purpose. It belongs to the group of methods of so-called phase-distributed implementation of parallel control algorithms; they are based on a convention that passing of a token through MPN transition is equivalent to executing both operation phases of the block controlled. The principle is illustrated by Fig. 5; the initial control specification by MPN is shown in Fig. 5a, and the corresponding circuit modeling the phase-distributed implementation is diagrammed in Fig. 5b. It comprises distributive asynchronous cells (DAC) Ya_i corresponding to the positions Q_i of MPN, and sometimes additional cells (Ya_{23}) for implementation of intermediate control logic. The implementation of cells of two types used in this example - Ya_i, i = 1-4 (the first type), and Ya_{23} (the second type) is illustrated by Fig. 5c and d, respectively. The initial state of the circuit of Fig. 5b is given in brackets. The symbol (1)* marks excited signal states, i.e., the states about to be changed.

The specific feature of phase-distributed implementation is the fact that the sequence in the execution of the operations OP1 and OP2 on respective blocks is performed in such a way that the ordering of the changes in request signals and response signals a_1, b_1 and a_2, b_2 is organized according to the signal graph of Fig. 5b. As a result, the block OP1 performs both phases: the working phase (transitions $-a_1$, $-b_1$) and blanking phase (transitions $+a_1$, $+b_1$) before the block OP2 is activated.

In order to organize the above signal control discipline, each operation marking a MPN transition (see Fig. 4) is assinged to start signal a_1 and several response signals b_i.

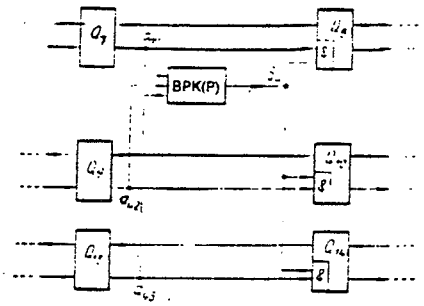

Fig. 7. Example of organization of a repeat call circuit.

For example, the operations of BPK are associated with the following combinations of control signals: BPK(M), a_4, b_4; BPK(P), a_4, b_3; BPK(S), a_6, b_3; BPK(P, M), a_4, (b_3, b_4); BPK(S, M), a_6 (b_3, b_4).

After completing the signal interpretation of MPN operator transitions, a library of DAC must be formed to be used subsequently for construction of control circuits. At the "ruptures" of direct links between several DAC, the respective control signals (a_1, b_1) are generated to mark the respective transitions between MPN precisions.

Figure 6 shows examples of selected types of DAC library elements which provide a circuit support to the generation of a desired "object code," resulting from translation of "initial" code specified by an MPN. Column a gives the notation of the cell type with its incident links (the arrows indicate the direction of signal transmissions through the communication lines: the right-pointing arrows mark direct links between DAC and the direction of links in MPN; the left-pointing arrows specify the feedback links between DAC and serve for implementation of a phase-distributed circuit operation discipline. Column b gives the corresponding circuits.

The translation procedure can be carried out in two stages. In the first stage, the "object code" is generated from DAC set in accordance with the limited set of rules defining the semantic equivalence of DAC for individual local MPN components. For example, a condition node (without branching) corresponds to cell of type 1; a node with a branching condition (predicates on outgoing arcs) corresponds to a cell of type 2; a node of a synchronizing transition type with several input position-conditions is represented by a cell of type 3, etc.

The second state is the optimization of the "object code." Since the object code generated according to the above rules can be redundant in terms of the DAC used, a deeper semantic analysis of MCN fragment is desirable (fragment context analysis); it allows modeling by a single DAC cell a certain set of typical MPN nodes rather than just one node. Such combination of functions involves the possible use of AND, OR, AND/OR, and OR/AND logic on direct and reverse DAC inputs. It results in a saving DAC memory elements, which is a key to raising SU operation speed. In particular, context analysis can be based on separate transformation rules; for example, DAC of type 4 models a sequential connection of transition-synchronizer node and position-condition with branching; DAC of type 5 represents a combination of synchronizer and position-assembly as modeled by exclusive OR (note that only safe MPN are modeled).

An important aspect in MPN modeling in terms of DAC is arranging feedbacks between cells. For example, the cell representing the position Q_2 in Fig. 4 will initiate, depending on conditions P3 and $P\overline{3}$, either the operation BPF1(B), and then the cell corresponding to Q_3 or the cell corresponding to $Q_{4,5}$. The logical function of feedback for the cell Q_2 is, therefore, logical OR. Other feedback functions are also possible.

A control MPN can contain identically marked transitions representing a multiple use of the same operation. In other words, the SU behavior should provide a possibility of repeatedly calling an operation from different points in the algorithm. A repeated call circuit is used; it is illustrated for BPK block and BPK(P) operation with start signal a_4 and response signal b_3 in Fig. 7. The diagram illustrates the method of starting BPK(P)

from the points in the control algorithm corresponding to the transitions of MPM of Fig. 4 situated between pairs of positions Q_7 and Q_8, Q_9 and Q_{10}, Q_{13} and Q_{14}, etc. With AND logic at the inputs of the cells Q_9, Q_{10}, and Q_{14} it is possible to ensure that the control tag moves only on one of the three branches.

In conclusion, we should note that the complexity of the resulting control circuit is a linear function of the dimension of the protocol algorithm; it amounts approximatel to 200 NAND gates and AND/NOR gates with the fan-in coefficient not greater than 4.

We have not discussed the construction principles and the structural organization of control and recovery automaton. In view of the specifics of these aspects and the fact that the development of error detection and recovery algorithms constitutes a separate subject of study, the authors plan to consider them in a separate publication.

REFERENCES

1. V. I. Varshavskii, V. B. Marakhovskii, L. Ya. Rozenblyum, Yu. S. Tatarinov, and A. V. Yakovlev, "Transition from physical level protocol specification to its circuit implementation," AVT*, no. 5, pp. 82-88, 1987.

2. V. I. Varshavskii, V. Ya. Volodarskii, V. B. Marakhovskii, L. Ya. Rozenblyum, Yu. S. Tatarinov, and A. V. Yakovlev, "Structural organization and protocols of information exchange of a fault-tolerant self-synchronous ring channel," AVT*, no. 4, pp. 48-55, 1988.

3. R. H. Campbell and B. Randell, "Error recovery in asynchronous systems," IEEE Transact. on Software Engineering, vol. SE-12, no. 8, pp. 811-826, 1986.

4. Automatic Control of Asynchronous Processes in a Computer and Discrete Systems [in Russian], Nauka, Moscow, 1986.

12 February 1988

*Automatic Control and Computer Sciences. Available from Allerton Press, Inc., 150 Fifth Ave., New York, N. Y. 10011; (212) 924-3950.