REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE February 1997	3. REPORT TYPE AND DATES COVERED Final Report

4. TITLE AND SUBTITLE

The Self-Synchronous Schemotechnique as a Design Basis for The Fail-Safe Wafer Scale Integration Emergency Computers *RPT #2*

5. FUNDING NUMBERS

F6170896W0290

6. AUTHOR(S)

Prof. Adolf Filin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute of Informatics Problems (IPI RAN)
Ul. Vavilova 30/6
Moscow 117900-GSP-1
Russia

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
PSC 802 BOX 14
FPO 09499-0200

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

SPC 96-4086

11. SUPPLEMENTARY NOTES

Report is in two parts: Intermediate Report No 1 and Concluding Report No 2

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

A

13. ABSTRACT (Maximum 200 words)

This report results from a contract tasking Institute of Informatics Problems (IPI RAN) as follows: The contractor will prepare a detailed technical report to include the analysis of present state trends in the development of computer systems IC base including a description of requirements for Emergency Computer schemotechnics. his report will also include a discussion of the main principles of self-timing as well as analysis and description.

DTIC QUALITY INSPECTED

14. SUBJECT TERMS Computers		15. NUMBER OF PAGES 243
		16. PRICE CODE N/A

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

RUSSIAN ACADEMY OF SCIENCES

INSTITUTE OF INFORMATICS PROBLEMS
(IPI RAN)

SPECIAL CONTRACT PROGRAM

SPC 96-408*φ*

THE SELF-SYNCHRONOUS SCHEMOTECHNIQUE AS A DESIGN BASIS FOR THE FAIL-SAFE WAFER SCALE INTEGRATION EMERGENCY COMPUTERS

CONCLUDING REPORT No 2

WITH EUROPEAN OFFICE OF AEROSPACE
RESEARCH AND DEVELOPMENT (EOARD)

CONTRACT F61708-96-W0290

MOSCOW, RUSSIA

AUGUST, 1997

DTIC COULD NOT GET MISSING

PAGE FROM CONTRIBUTOR

1-88

Principal Investigators

Dr. A. Filin

Dr. J. Stepchenkov

Dr. L. Plekhanov

Dr. J. Rogdestvensky

Dr. J. Diachenko

V. Petrukhin

PREFACE

This document - Part 2 of the Report - contains the results of further investigation of problems presented in the Part 1 and dedicated to using of the self-synchronization in computing systems. Both parts of the Report confirm the fruitfulness of the self-synchronous interaction principle when applied to computers.

The idea of circuits which detect the moment of transition process completion was proposed first by D. E. Muller in early 50-th. But the idea, as it happens often, was ahead of its time. The lack of theoretical base for design of such circuits, appropriate technological support, and social order on such systems slowed down their development for long time. The investigations started in the middle of 50-th in the USA met with raising in early 70-th, decrease in 80-th, and only the crisis with synchronization in VLSI in early 90-th caused again a raise of interest to the self-synchronous or, more exactly, quasi-self-synchronous schemotechnique. Step-by-step, self-synchronous circuits turned to become from scientific exotics to competitive tendency winning their niche in computer design. But even at present time the idea of creation of completely self-synchronous computers did not become widely popular and are promoted by efforts of some groups of specialists who believe in it. The main source of such a situation consists in the fact that nobod\y up to now proved practical attainability of advertisable merits of the self-synchronous schemotechnique even today and its ability to compete with the synchronous schemotechnique generally used in VLSIs.

The lack of specific information about relative characteristics of synchronous (S), asynchronous (A) and self-synchronous (SS) variants of the same circuits implementation did not allow designers to make unprejudiced conclusion on optimal application areas for them. The results of investigations available in this report partially clarify this situation. The first real attempts to make such comparison have been performed within frames of this contract; the results are described in the further chapters.

To give the final answer regarding all profits and expedient application areas for different schemotechniques, it is necessary to make comparative design not only for separate functional components but also for microsystems, perhaps not big but full-valuable.

Apart from the results of S-, A- and SS-schemotechniques comparison, Part 2 of the report includes information on an implemented concept of SS-circuits design-through, used algorithms for their analysis and synthesis, methods and methodology of combinational SS circuits design and also description of demo-programs and slide-show for the CAD system FORCAGE (including a demo-version) and slide-show for the CAD system RONIS. This report includes also a short description of the results of project devoted to development of a highly reliable computer for emergency real-time applications, with the element base of the WSI level relied upon the SS schemotechnique (the HREC project).

Concluding report No 2 contains 176 pages.

CONTENTS

1 SCHEMOTECHNIQUE SPECIFICITY OF S³ CIRCUITS DESIGN

1.1 Methods of building up combinational S³ circuits

At present, it is created the theoretical base of building up self-synchronous (SS) circuits consistent with various model representations, i.e. hypotheses on delay character in logic elements and connection wires. A comparative analysis of the hypotheses is performed in [1.1].

The issues of development of SS circuit design techniques enabling a user to build up arbitrary combinational circuits are reflected in the literature significantly less. Most comprehensively the issues of the SS circuit design are discussed in [1.1, 1.2].

1.1.1 Procedure of SS circuits assignment

The basic items of an assignment procedure for SS circuits are:

- accepting a hypothesis on the delay character and values in elements and wires
- selecting a self-synchronous code for input and output signals representation
- accepting a specific succession discipline for input and output sets
- determining an indication method of the input signals validity and transition process completion, in all circuit elements.

1.1.1.1 Hypothesis on the delay character and values

The following assumptions are taken in the hypothesis on the delay character and values.

1) Delays in elements are of arbitrary values not having top bounds but finite.

2) Delays in wires before branchings are reduced to delays in source (producing a signal) elements and may be of arbitrary values.

3) Delays in wires after branchings are reduced to delays in destination (consuming a signal) elements, with the delay deskews being obligatory less than a minimum delay in the destination elements.

4) The delays in AND expanders, within AND - OR - NOT gates, and OR expanders, within

OR - AND - NOT gates, are accepted equal to zero, with all these delays assumed to be reduced to delays of the output inverters (the *single-cascade* gates requirement in terms of [1.3]).

1.1.1.2 Self-synchronous codes for representation of input and output signals

Without a formal definition of a *self-synchronous code system* (refer to [1.1]), we only note here that, in such a system, a completion moment of each transition $a \rightarrow b$, i.e. replacement of the set a by the set b, can be determined on the fact of appearance of the valid set b itself independently of the transition duration.

Two-phase codes are used in the self-synchronous schemotechnique. At each transition $a \rightarrow b$, one of the sets services as a *spacer*, i.e. information sets delimiter not bringing information as itself. The transition from an information set to a spacer is referred to as *reset* (or *service*) *phase* while the reverse transition — as *work phase*.

Below, only one class of the two-phase codes — *paraphase codes* — is considered briefly. At the paraphase coding, each information bit is represented by two binary variables. In a number of applications, e.g. at the data transfer over communication channels, the paraphase code is more redundant as compared with other two-phase codes (*codes with identifier, optimal equiponderant codes*, etc.). However, the paraphase code is widely applied due to its simplicity and ease of interfacing to conventional nonredundant binary codes.

The class of paraphase codes comprises the following code types:

a) having the *zero-spacer*, — with each variable X_i being represented by three pairs

$$(X_i \ \overline{X}_i) \in \{(0\ 1), (1\ 0), (0\ 0)\};$$

b) having the *one-spacer*, — with each variable X_i being represented by three pairs

$$(X_i \ \overline{X}_i) \in \{(0\ 1), (1\ 0), (1\ 1)\}.$$

Both types are distinguished only by representation of the spacer. This enables the designer to combine both code types resulting in a combined paraphase code with the alternating *dual-spacer*.

A dominant majority of self-synchronous combinational elements represented in [1.1, 1.2] use the paraphase code with the zero-spacer, a few of them apply the one-spacer, and none — the dual-spacer.

1.1.1.3 The succession discipline for input sets

The succession discipline of input and output sets is, at circuit design, a source information that is fixed, for the listed code types, as follows.

The **input zero-spacer discipline** — for inverting (a) and noninverting (b) elements:

a) input information set → output information set → input zero-spacer → output one-spacer → etc.

b) input information set → output information set → input zero-spacer → output zero-spacer → etc.

The **input one-spacer discipline** — for inverting (a) and noninverting (b) elements:

a) input information set → output information set → input one-spacer → output zero-spacer → etc.

b) input information set → output information set → input one-spacer → output one-spacer → etc.

The **input dual-spacer discipline** — for inverting (a) and noninverting (b) elements:

a) input information set → output information set → input zero-spacer → output one-spacer →input information set → output information set → input one-spacer → output zero-spacer → etc.

b) input information set → output information set → input zero-spacer → output zero-spacer →input information set → output information set → input one-spacer → output one-spacer → etc.

The disciplines need some additional information for selecting a variant of building up combinational circuits. This issue is considered in detail in the next subsection.

1.1.2 Indicator implementation variants in combinational elements

1.1.2.1 General consideration

Various implementations of combinational SS circuits are possible as a result of combining two kinds of units:

- a *Function Unit* (FU) executing the required combinational function
- an *Indication Unit* (IU) or *indicator* detecting and indicating the moment of transition process completion in circuit.

A generalized structure of a combinational SS circuit is represented in *Fig. 1.1*. All inputs X_1 $\overline{X}_1,\ldots, X_n$ $\overline{X}_n$, common for both the FU and IU, and outputs Y_1 $\overline{Y}_1,\ldots, Y_n$ $\overline{Y}_n$ are encoded using one of paraphase codes with a spacer considered in the previous paragraph. The output indicator I is responsible for signaling transition processes completion in the circuit.
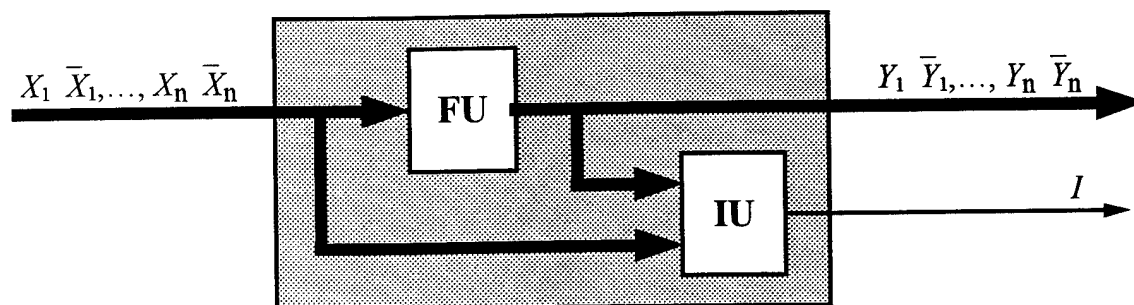
Fig. 1.1   **Generalized Structure of a Combinational SS Circuit**

Fig. 1.2 explains indicator implementation variants as so called *Hysteresis* flip-flops (*H*-ff) [1.1, 1.2] for two single-phase (*a*) and paraphase (*b*) signals or as the Muller's *C-element* [1.4] (*c*). Logically, the *H*-ff and *C*-elements are identical. Their behavior for n single-phase inputs I_n is described by the equation:

$$I_{(t)} = (I_1 I_2 \ldots I_n) \vee I_{(t-1)} (I_1 \vee I_2 \vee \ldots \vee I_n), \qquad (1.1)$$

where:

- $\vee$ — denotes disjunction (logic OR)

- $I_{(t-1)}$ and $I_{(t)}$ — are a previous and next states of the indicators, respectively.

The criterion of transition process completion for the checked single-phase indicators is fulfilment of the condition $I_1 = I_2 = \ldots = I_n = 1$ in one phase, and $I_1 = I_2 = \ldots = I_n = 0$ — in another phase.

The *H*-ff functioning logic, *Fig. 1.2a*, needn't comments.

Fig. 1.2b shows a logic scheme of an *H*-ff for indication of paraphase signals with the one-spacer. For this case, all indicator variables I_i in the equation (1.1) should be replaced by pairs I_i $\overline{I}_i$, $i = 1,\ldots, n$.
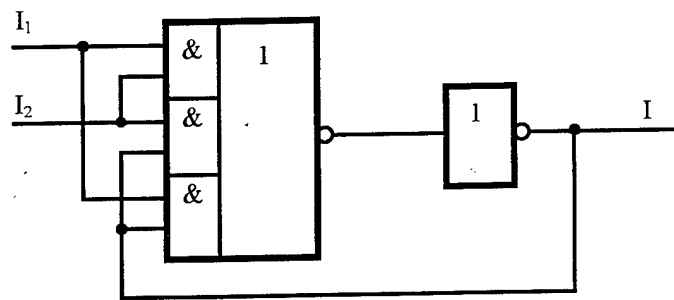
a) Single-phase *H*-flip-flop
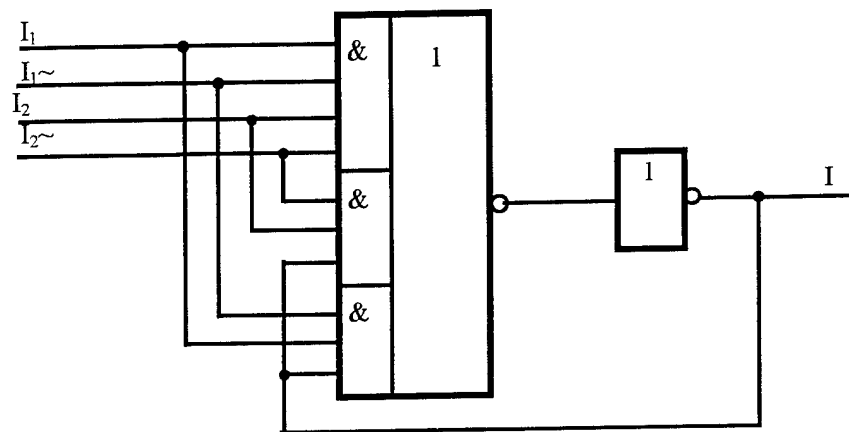
b) Paraphase *H*-flip-flop
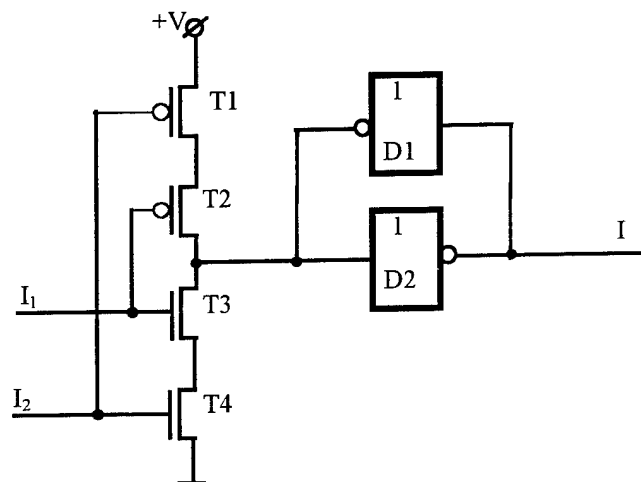
c) CMOS Muller's *C*-element

Fig. 1.2   Indicator Implementation Ways

The *C*-element, *Fig. 1.2c*, functions as follows. In the initial state, the transistors **T3** and **T4** are open, and the HIGH logic level is forced on the **D2** output. The **D1** weak output confirms the LOW logic level on the **D2** input. At transition of only one of input signals to the LOW logic level, the **D2** input occurs in an uncertain state (a *floating* state, with no input voltage source). However, the LOW logic level is retained on the **D2** input due to the **D1** feed-back, and the **D2** output state is not changed. At transition of the second input signal to the LOW logic level, the transistors **T1** and **T2** get open and, being more strong than the **D1** transistors, override them and pull up the **D2** input to the HIGH logic level. Then the formed LOW logic level on the **D2** output is confirmed over the **D1** feed-back. The reverse process, i.e. process completion indication for transition of both input signals to the state $I_1 = I_2 = 1$, is analogous.

Further, only some combinational circuits implementation variants are discussed. Therefore, any recommendations on variant selection are preliminary. More exact recommendations may be derived from a larger number of examples.

In principle, 5 basic methods of building up the strictly self-synchronous (SSS or S^3) elements are possible.

1.1.2.2 First Method of building up combinational S^3 circuits

At the 1st Method of building up combinational S^3 circuits, an FU implements the required function with no respect to transition process completion on all inputs. The transition of the FU from the reset phase to the work phase and vice versa is accomplished in compliance with the diagram represented below.

Reset phase $\underrightarrow{\quad F (X_1\ \overline{X_1},\ldots, X_i\ \overline{X_i}) \mid X_i \neq \overline{X_i},\ i = 1,\ldots, min\quad}$ *Work phase*

$\boxed{Y = \overline{Y}}$ $\underleftarrow{\quad \text{For any } X_i\ \overline{X_i} \mid X_i = \overline{X_i};\ i = 1,\ldots, min\quad}$ $\boxed{Y \neq \overline{Y}}$
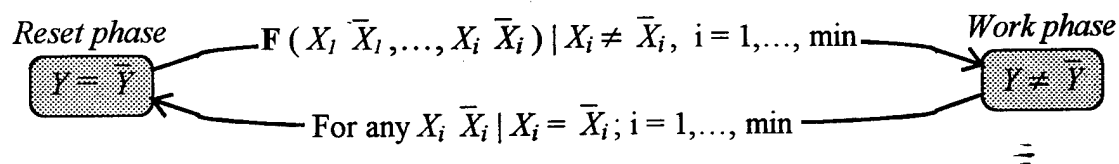
In the diagram, a paraphase element output $Y\ \overline{Y}$ transits from a reset phase (a spacer) to a work phase after transition processes completion of only minimally necessary, for implementation of the function **F**, rather than of all inputs $X_i\ \overline{X_i}$. For example, a single logic TRUE is sufficient on inputs of a multi-input OR gate to force its output to the TRUE state regardless the state of other inputs. For a self-synchronous paraphase 2×2 AND - OR gate, the necessary and sufficient condition of transition to the work phase may happen logic TRUE states on two of its 4 inputs. However, transition to the reset phase of any one of inputs belonging to the minimally necessary set initiates immediately transition of the element output to the reset phase.

As an example, a paraphase 3-input OR gate with the one-spacer on both inputs and output is given in *Fig. 1.3*. Its circuit includes four gates, **D1** ÷ **D4**, and the *C*-element-based indicator **D5** that is responsible for indication of transition processes completion on all inputs and output in both phases. Thus, the indicator implementation requires more transistors and depends on the number of cascades within it. The hardware expenditures for various S^3 circuit implementation variants are summed up in *Table 1.1*[1].

Table 1.1

Hardware Required for Combinational S^3 Circuits

Variant \ Method		Gate type	Number of transistors		
			FU	IU	FU + IU
1	a	2 OR	12	20	32
		3 OR	16	28	44
	b	2 OR + 2 AND	12	28	40
		3 OR + 3 AND	32	40	72
	c	2 OR + 2 AND	12	12	24
2	a	3 OR	28	26	54
	b	3 OR + 3 AND	56	32	88
	c	3 OR + 3 AND	40	32	72
3		3 OR	37	4	41
4		3 OR	62	14	76
5		3 OR	52	4	56

Note. Hereafter, only CMOS S^3 circuits are considered, with the number of transistors in the FUs being twice greater than the number of its inputs (this rule is not applicable to the IUs). The element basis within this subsection comprises only gates AND - NOT, OR - NOT, and OR - AND - NOT, with the number of consecutive transistors not exceeding four in the considered examples.

The 1st Method of building up combinational S^3 circuits is expedient in cases when paraphase signals have many destinations, i.e. are consumed by several receivers. At a constant number of paraphase inputs, increase of the number of gates in a circuit augments its hardware proportionally to the number of outputs. In *Fig. 1.4*, an independent 3-input paraphase AND gate with the one-spacer on input and output signals is added to the 3-input OR gate. The hardware estimations are see in *Table 1.1*.

---

[1] It should be noted that the figures in the table are valuable for comparisons of implementation variants only.
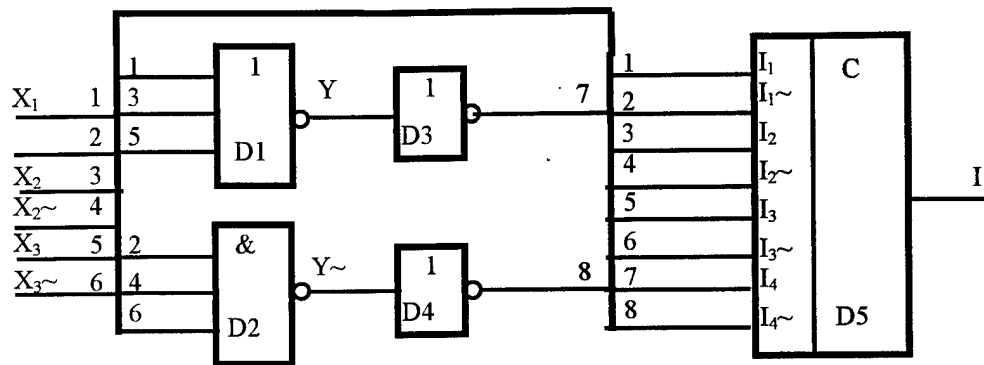
$$I=(I_1 I_1{\sim} ... \ I_4 I_4{\sim})+i(I_1 I_1{\sim}+ ... \ +I_4 I_4{\sim})$$

Fig. 1.3   **Electrical Circuit of 3 OR Gate**
(variant 1a of *Table 1.1*)
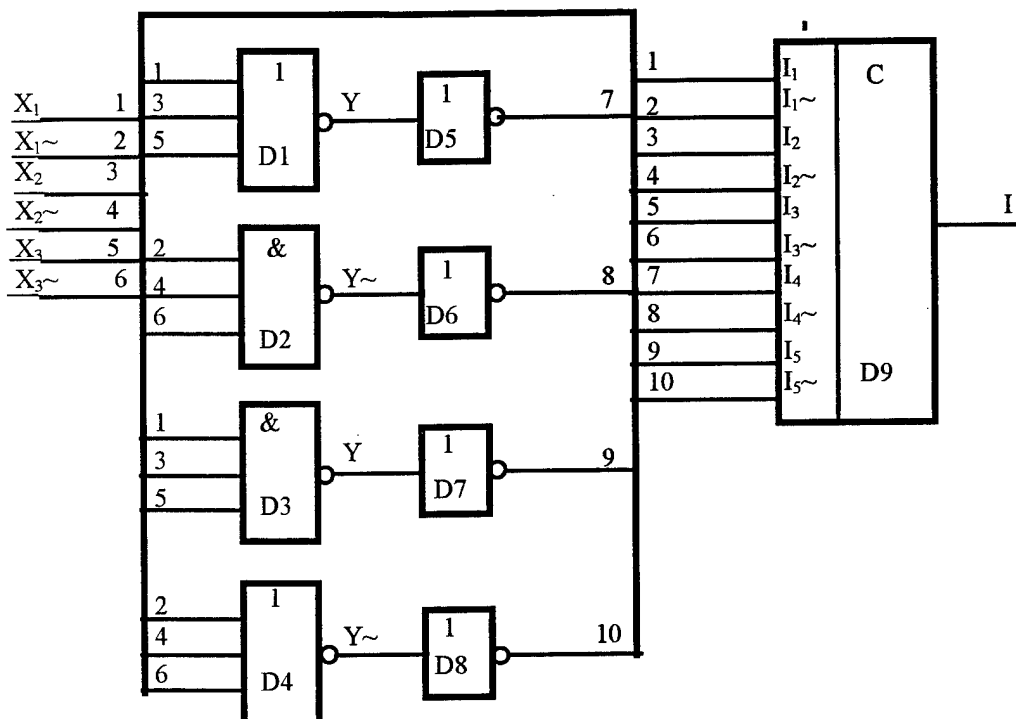
Fig. 1.4   **Electrical Circuit of Combined 3 OR + 3 AND Gate**
(variant 1b of *Table 1.1*)

A large number of inputs makes the IU multi-cascade and reduces the circuit performance.

Circuits built up on the 1st Method may be characterized as *circuits with a centralized indicator*.

The 1st Method is universal and ensures building up correct S^3 circuits. The payment for simplicity of using this formal method is a certain S^3 circuits hardware redundancy. Nevertheless, elements built on this method provide not only information processing but also *indicativity*, i.e. ability to detect transition processes completion in respect to a number of input sets.

For example, appearance on one of inputs of a 2-input OR gate with a spacer of the logic ZERO (X_1 $\overline{X}_1 = 0$ 1) does not force changing its output (Y $\overline{Y} = 1$ 1) until its second input is retained in the reset phase (X_2 $\overline{X}_2 = 1$ 1, i.e. transition process on the second input is not completed). Change of the gate output state (i.e. transition to the work phase) will occur after one of action sets on the second input will have been established (Y $\overline{Y} = 0$ 1 at X_2 $\overline{X}_2 = 0$ 1 and Y $\overline{Y} = 1$ 0 at X_2 $\overline{X}_2 = 1$ 0). Thus, the fact of transition process completion for a number of input signal sets can be detected not only by a direct analysis in the indicator but also by indirect checking outputs of the element that processes the input sets.

The indicator, variants 1a and 1b, accomplishes checking transition processes completion for a number of input signal sets twice. Sometimes, eliminating such a doubling enables an S^3 circuit to become simpler. For example, eliminating the doubling in the 2 OR + 2 AND combined gate, *Fig. 1.5*, reduces the number of transistors required (see variants 1b and 1c in *Table 1.1*). This is an "ideal" case when the FU of two elements provides mutually indicativity of all input signal sets. Checking of only the circuit outputs is sufficient for building up the indicator.
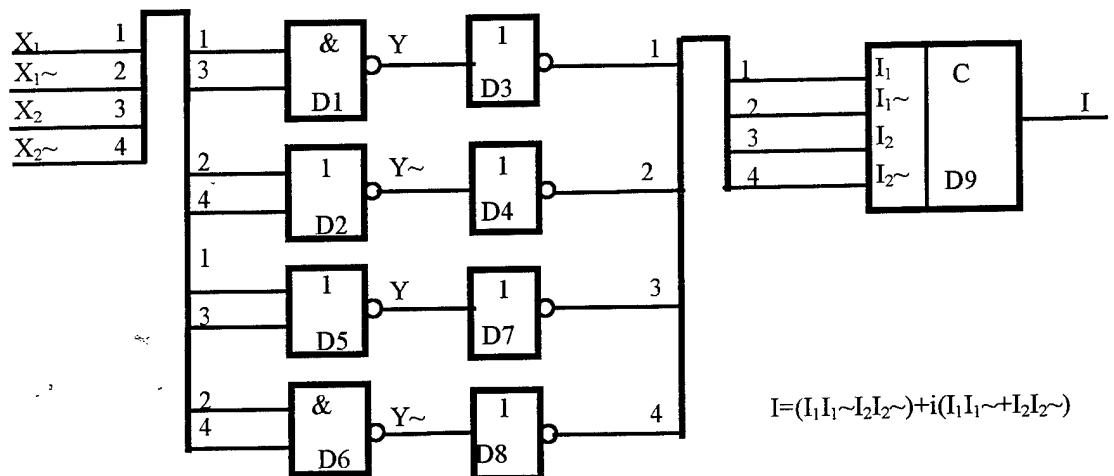
Fig. 1.5   **Electrical Circuit of Combined 2 OR + 2 AND Gate**
(variant 1c of *Table 1.1*)

1.1.2.3 Second Method of building up combinational S^3 circuits

The 2nd Method of building up combinational S^3 circuits supposes implementation of not only a required function itself but also of a portion of the indicator to check transition processes completion in input signals at transition from the reset phase to the work phase while the indicator checks transition processes completion in output signals, in both phases, and in input signals at transition from the work phase to the reset phase only. The 2nd Method is described by the following diagram.

Reset phase $\mathbf{F}(X_1\,\overline{X}_1,\ldots,X_i\,\overline{X}_i)\,|\,X_i \neq \overline{X}_i,\ i=1,\ldots,n$ *Work phase*

$Y = \overline{Y}$

For any $X_i = \overline{X}_i$, $i = 1,\ldots, n$

$Y \neq \overline{Y}$
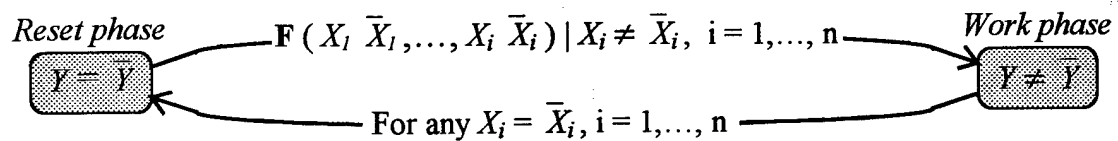
For an element, transition from the reset phase to the work phase is permitted only when all inputs complete transition to the work phase. The reverse transition, from the work phase to the reset phase, takes place when at least one of input signals returns to the reset phase.

Fig. 1.6 represents a modification of the 3-input OR gate on the variant 2a, and *Fig. 1.7 —* of the 3 OR + 3 AND gate on the variant 2b. The S^3 gates 3 OR and 3 AND in *Fig. 1.7* use the same input paraphase signals and, hence, double detecting transition process completion in the input signals at transition to the work phase. From the functional point of view, the detection can be implemented in one S^3 element only that is reflected in *Table 1.1* as variant 2c.

$$I = (I_1 I_1\!\sim \ldots\ L_4 L_4\!\sim) + i(L_4 L_4\!\sim)$$
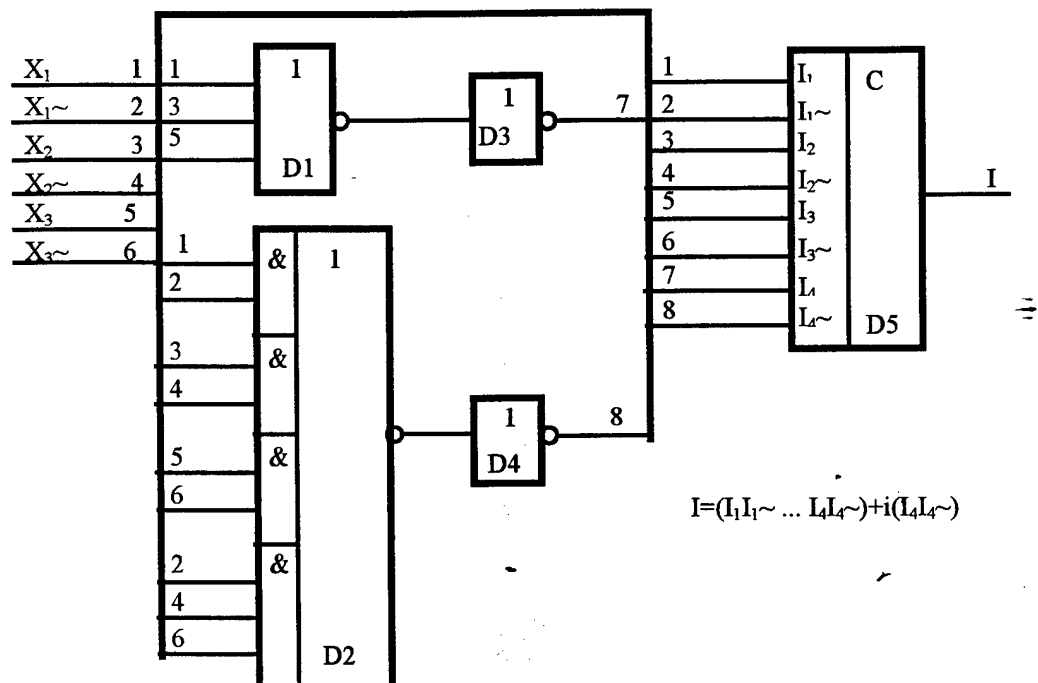
Fig. 1.6 **Electrical Circuit of Combined 3 OR Gate**
(variant 2a of *Table 1.1*)

$$I=(I_1I_1\sim \ldots\ I_5I_5\sim)+i(I_4I_4\sim+I_5I_5\sim)$$
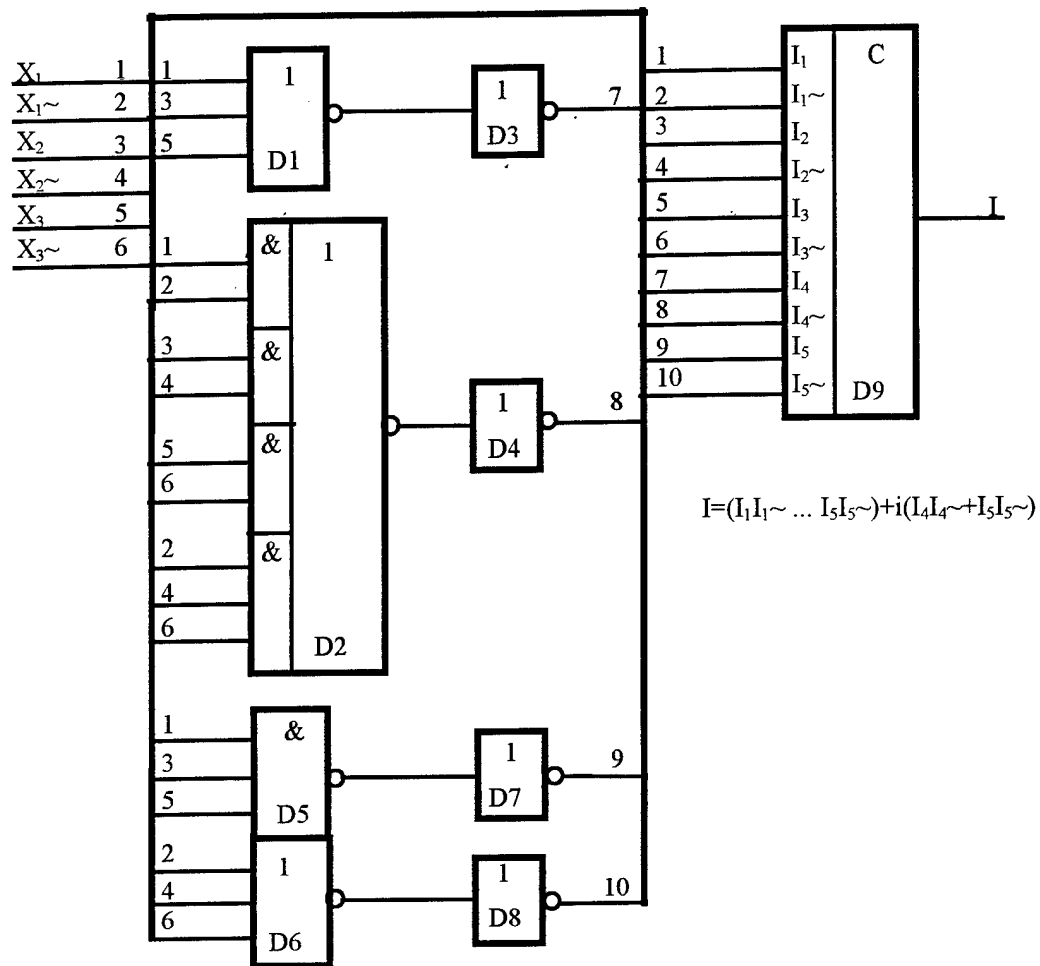
Fig. 1.7   **Electrical Circuit of Combined 2 OR + 2 AND Gate**
(variant 2b of *Table 1.1*)

Circuits built up on the 2nd Method may be characterized as *circuits with a partially centralized indicator*.

Table 1.1 shows 1st and 2nd Methods to be near identical, with respect to the number of transistors required, for the case of "single source — multiple destinations".

1.1.2.4   Third Method of building up combinational S^3 circuits

Theoretically, it is possible the 3rd Method of building up combinational S^3 circuits, with the FU fully responsible for checking both transition processes completions: in input signals at transition from the reset phase to the work phase and in input and output signals at transition from the work phase to the reset phase — in compliance with the following diagram.

Reset phase — $F(X_1\,\overline{X_1},\dots, X_i\,\overline{X_i}) \mid X_i \neq \overline{X_i},\ i = 1,\dots, n$ — *Work phase*

$Y = \overline{Y}$

Simultaneously for all $X_i = \overline{X_i}$

$Y \neq \overline{Y}$
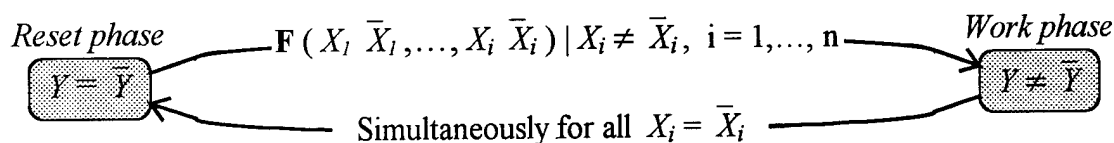
For an element, transition from the work phase to the reset phase is permitted only when all input signals return to the reset phase. By this, a combinational element is transformed to a memory element, with the indicator checking solely element outputs. *Fig. 1.8* illustrates implementation of the 3 OR gate on the 3rd Method, hardware estimates are given in *Table 1.1* as variant 3.
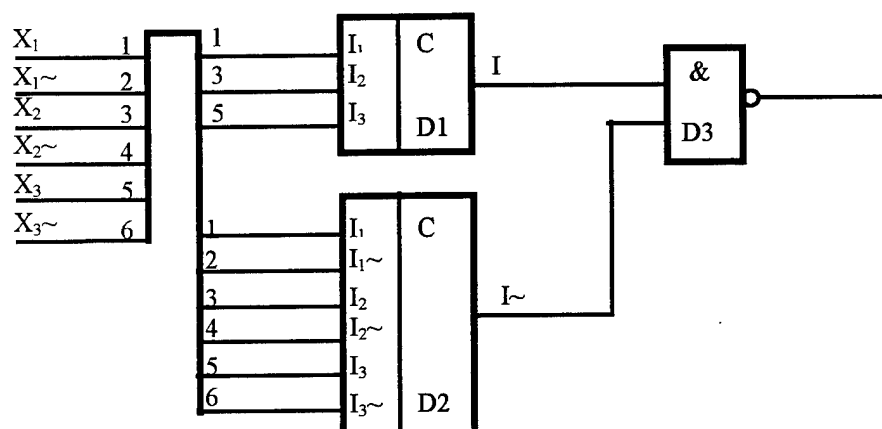
$I=(I_1 I_2 I_3)+i(I_1+I_2+I_3)$

$I\sim=(I_1 I_1{\sim}I_2 I_2{\sim}I_3 I_3{\sim})+i(I_1 I_1{\sim}+I_2 I_2{\sim}+I_3 I_3{\sim})+(I_1{\sim}I_2{\sim}I_3{\sim})$

Fig. 1.8   Electrical Circuit of Combined 2 OR Gate
(variant 3 of *Table 1.1*)

The basic difference of the S^3 circuits on the 3rd Method consists in their ability to detect, at the logical level, not only constant faults (CFs) on element outputs but also some CFs types that result in breaking S^3 circuit operation correctness and can not be detected at implementation on the 1st and 2nd Methods. It should be noted that the total hardware expenditures, the indicator included, are not obligated to increase (compare variants 1a and 3 in *Table 1.1*).

1.1.2.5 Fourth Method of building up combinational S^3 circuits

The three considered above Methods of implementation, in combinational S^3 elements, of the function of transition processes completion indication exhaust principally the formal indication methods reasonable on hardware criterion. Applying the S^3 code with the dual-spacer enables the S^3 circuit checkability level to increase and provides for detecting 100 % of CFs in output elements and 100 % of CFs of following types:

- mutual shortings of paraphase signals
- breaking or shorting to "ground" (Vss) of input signals
- double faults, breakings and shortings, along any paraphase signal.

For any element, transition from the reset phase to the work phase and vice versa complies with the following diagram.

Reset phase $Y = \overline{Y} = 0$ — For all $X_i = \overline{X}_i = 1$ — *Reset phase* $Y = \overline{Y} = 1$

— For all $X_i = \overline{X}_i = 0$ —

For all $X_i = \overline{X}_i = 0$ $\mathbf{F}\,(\,X_1\,\overline{X}_1, \ldots, X_i\,\overline{X}_i\,)\,|\,X_i \neq \overline{X}_i,\ i = 1, \ldots, n$ For all $X_i = \overline{X}_i = 1$

$Y \neq \overline{Y}$
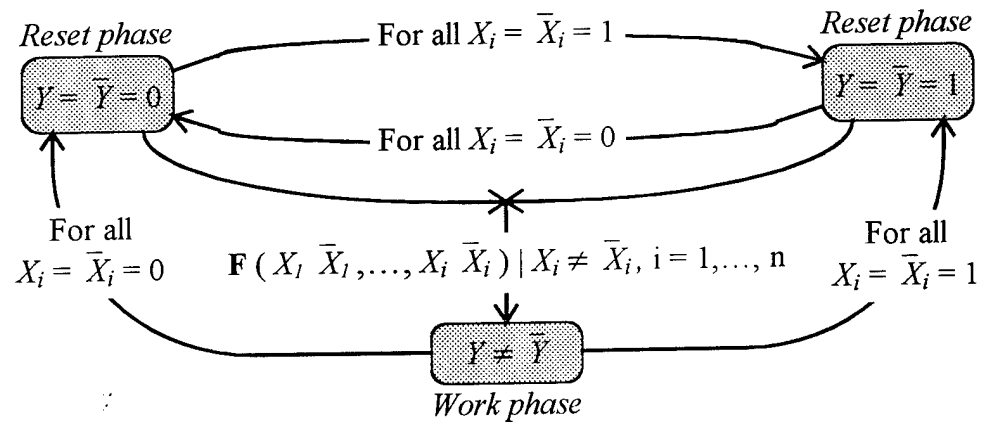
Work phase

Transition of a dual-spacer output $Y\,\overline{Y}$ from the work phase to the reset phase is possible only after all alternating dual-spacer inputs have returned to the reset phase. The input and output signals, in the case under consideration, have in the reset phase the same alternating spacer, e.g. if, in a current state, the input signals have the zero-spacer then the output signal has the zero-spacer as well. Completion of a next work phase will take place only after transition of all input signals to the state with the one-spacer. A next reset phase will be featured by the zero-spacers of the input and output signals, and so on.

Application of the S^3 code with the alternating dual-spacer is followed by increasing the number of transistors in comparison with the variant 1a (for hardware comparison, refer to *Table 1.1*).

Detection of a wider class of CFs is possible with non-logic methods (e.g. concerning schemotechnique, topology, and technology) that is not a matter of consideration here. A

comparative analysis of different methods on criterion of chip area required would allow an investigator to estimate advisability of their application for achieving a desirable CF detection level.

The hardware expenses featuring the dual-spacer implementation of an S^3 element can be considerably decreased if this element is a terminal (output) cascade of an S^3 circuit whose output is not obligated to have the dual-spacer. The variant 5 in *Table 1.1* illustrates the hardware expenditures for such a terminal 3 OR gate. The behavior character of input signals in variants 4 and 5 is identical: the zero-spacer and one-spacer alternate. The variants are distinguished by the behavior of the output signal in the reset phase. On the variant 5 specifically, any reset phase is featured solely by the same spacer type, one-spacer in the given example. Implementation complexity is reduced comparing with the variant 4 (see *Table 1.1*).

The hardware expenditures can be saved also:

• at the logic level, by applying S^3 codes with different spacers on circuit inputs and outputs due to removal of excessive output inverters (e.g. D3 and D4 in *Fig. 1.3*)

• at the topology level, by applying some special elements with specific parameters.

The variants of building up combinational S^3 circuits considered in this subsection relate to single-cascade S^3 elements. In real multi-cascade S^3 circuits, paraphase outputs of some elements are paraphase inputs of others, and, depending on the chosen building method, the indicator has to check some intermediate outputs. Other methods of indicator implementations in multi-cascade S^3 circuits (e.g. an *orthogonal implementation* and *collective responsibility*) are found in [1.2].

1.1.3   Basis of combinational S^3 circuits implementation

In the previous subsection there is considered some specific features of combinational S^3 circuit design with respect to diverse degrees of combining functions of FUs and IUs, with the only implementation basis being regarded — the disjunctive normal form (DNF).

Really, any table-defined Boolean function F_1 (except for zero) of single-phase variables can be represented as a disjunction of elementary conjunctions [1.5]:

$$F_1 (X_1, X_2, ..., X_n) = \vee (X_1^{L1} \wedge X_2^{L2} \wedge ... \wedge X_n^{Ln}), \qquad (1.2)$$

where "$\vee$" designates disjunction of conjunctions $X_1^{L1} \wedge X_2^{L2} \wedge ... \wedge X_n^{Ln}$, which total number can be 2n; each single-phase input variable X_i^{Li} is encountered not more than one time

within each conjunction; $X_i^{Li} = X_i$ if $Li = 1$, and $X_i^{Li} = \overline{X}_i$ if $Li = 0$. The index "1" in the designation F_1 means a function of single-phase variables.

The disjunction in the right part of (1.2) is taken on multitude of argument sets that transform the function F_1 into "1".

For paraphase variables with a spacer, the formula (1.2) may be rewritten in a less strong form:

$$F_2 (X_1 \; \overline{X}_1 ,..., \; X_n \; \overline{X}_n) = \vee (X_1 \; \overline{X}_1 \wedge ... \wedge X_n \; \overline{X}_n) \tag{1.3}$$

The index "2" in F_2 indicates a function of paraphase variables.

In the formula (1.2), any single-phase variable may be included in either direct (X_i) or inverse ($\overline{X}_i$) form. In the formula (1.3), in a general case, any paraphase variable with a spacer also may be included into any conjunction in either direct (X_i) or inverse ($\overline{X}_i$) form and determine, if necessary, behavior of the function F_2 of a paraphase variable $X_i \; \overline{X}_i$ on intermediate sets and the reset set.

For any work sets, an inverse output $\overline{F}_2$ of a paraphase variable $F_2 \; \overline{F}_2$ has a value opposite to the direct output F_2. Applying the de Morgan's rule to (1.3) gives:

$$\overline{F}_2 = \overline{F_2 (X_1 \; \overline{X}_1 ,..., \; X_n \; \overline{X}_n)} = \wedge (\overline{X}_1 X_1 \vee ... \vee \overline{X}_n X_n) . \tag{1.4}$$

The view of (1.4) conforms to the representation of the function $\overline{F}_2$ in the conjunctive normal form (CNF), with the maximal number of disjunctions being equal to 2n.

Definition 1. A two-channel implementation of a paraphase function with a spacer $F_2 \; \overline{F}_2$, in which the direct channel implements the function F_2, and the inverse channel — $\overline{F}_2$, with both functions being represented in the DNF, is named a disjunctive-conjunctive normal implementation (DCN implementation). Analogously, a conjunctive-disjunctive normal implementation (CDN implementation) takes place if a source function $F_2 \; \overline{F}_2$ is represented in the CNF.

Theorem 1. If a source function of single-phase variables $F_1 (X_1,..., \; X_n)$ is given in the minimal DNF then its DCN implementation is also minimal.

Proof. It is proved in [1.5] that if a function $F_2 (X_1 \; \overline{X}_1 ,..., \; X_n \; \overline{X}_n)$ is obtained from a function $F_1 (X_1 ,..., \; X_n)$ in compliance with the stated procedure then the obtained thus function F_2 is minimal of all possible predeterminations of the function F_1. Evident is the proof that the function $\overline{F}_2$ is also minimal.

Fig. 1.9 shows a DCN implementation of an S^3 gate 2 × 2 AND - OR - NOT. The element **D1** implements the direct function, and the element **D2** — the inverse one. The elements **D1** and **D2** are built on the 1st Method (see subsection **1.1.2**), i.e. implement only the FU while all other elements implement the *H*-ff responsible for checking transition processes completion in all input signals and the output signal in both phases.
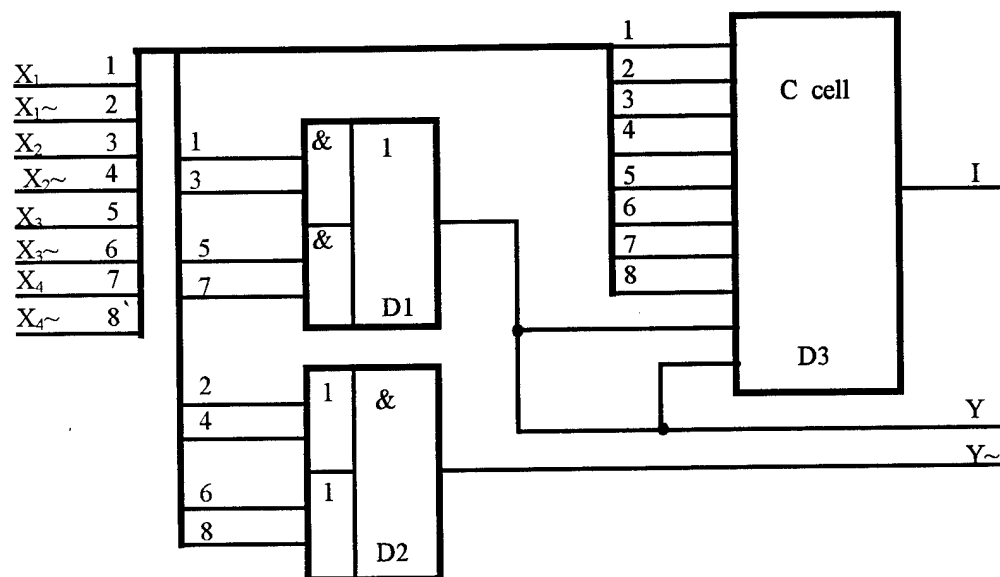
Fig. 1.9   Electrical Circuit of Combined 2× 2 AND - OR - NOT Gate in the Disjunctive -Conjunctive Basis(variant 4 of *Table 1.1*)

Analysis of base element libraries of zeroth and first levels for a number of gate arrays allows us to conclude that most of them are oriented on an efficient implementation of positive single-phase variables by DNF elements. However, implementation of the conjunctive form in the disjunctive basis results in augmented hardware expenditures. For example, implementation of the 2 × 2 AND - OR - NOT gate, *Fig. 1.10.* requires a double quantity of transistors in the inverse channel.
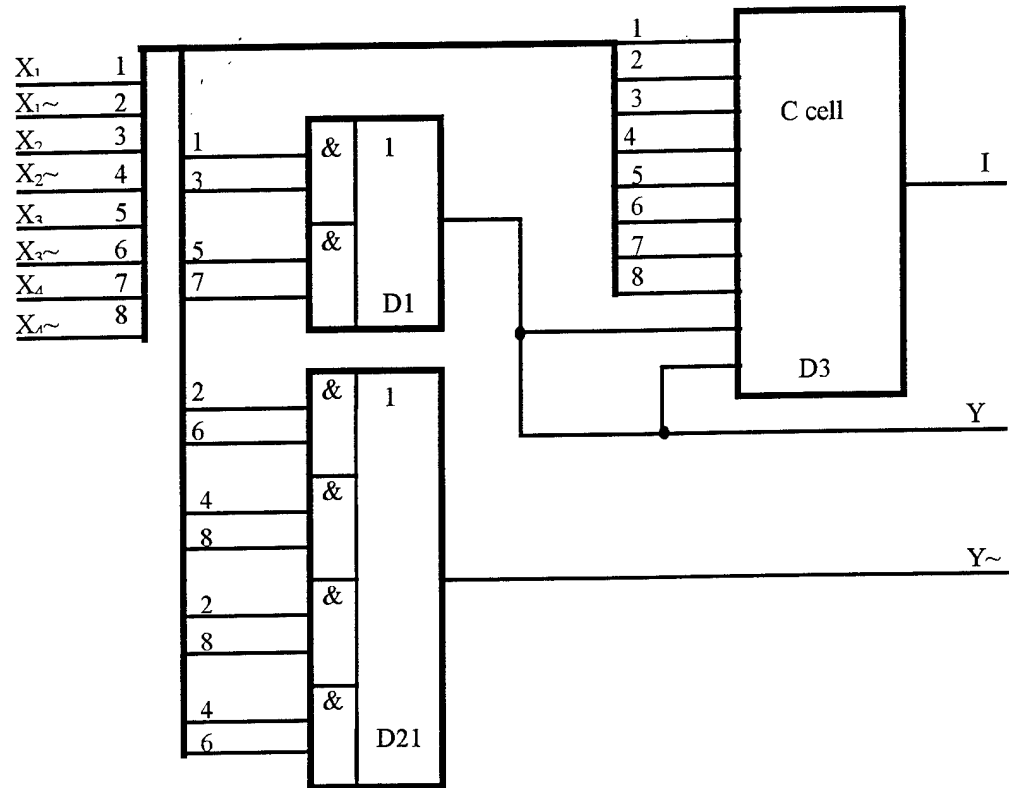
Fig. 1.10 **Electrical Circuit of Combined 2 × 2 AND - OR - NOT Gate
in the Disjunctive Basis (variant 5 of *Table 1.1*)**

1.2. Library elements and base cells for S³ circuits

For acceleration and automation of S³ circuits design, some predesigned fragments are grouped into libraries. There are libraries of typical circuits of functional devices and libraries of base cells (S³ base elements) in transistors.

The available S³ circuits libraries comprise near 50 elements and above 100 cells. This section is concerned with the library elements and cells used for S³ implementations of the test circuits described below.

For all circuits considered in the section, the spacer values (if exist) are given in parentheses.

Notes.

1) *Spacer* is an intermediate operation phase necessary for self-synchronous interactions in a circuit.

2) The following designations are accepted in diagrams of this section:

I — indication signals of a current phase (work or spacer);
C — control signals;
R — reset signals;
S — preset signals;
I^+ — a future value of the signal I.

Encircled in diagrams are weak transistors.

1) The single-clock controlled flip-flop with reset *T1* (*fig. 1.11*)
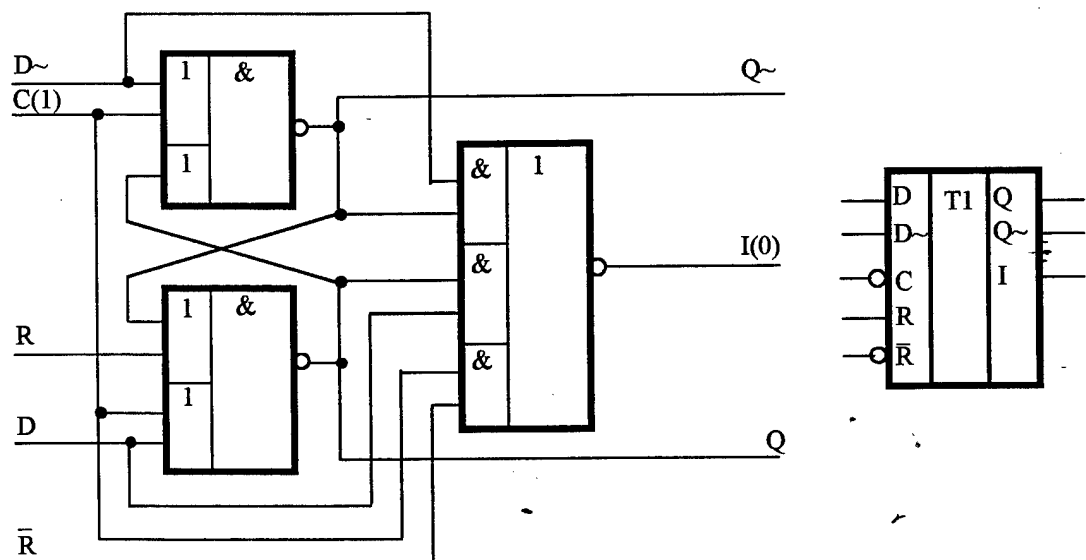
Fig. 1.11. **Single-clock controlled flip-flop *T1* and its symbol**

The flip-flop circuit described in [1.6] is modified with introducing the reset input R that does not reduce its performance in read and write operations.

The flip-flop stores one information bit during the spacer, i.e. one half of the two-phase operation cycle. The value of a stored bit is changed during the work phase. The inputs D and $D\sim$ must be updated only within the spacer ($C = 1$). During the work phase, $C = 0$ enables the flip-flop to transit to a new state, with the signal I becoming equal to 1 as soon as outputs Q and $Q\sim$ correspond to inputs D and $D\sim$.

The signal R resets the flip-flop while the signal $\overline{R}$ forces the indicator I to 1. After termination of the reset signal, the indicator returns to 0. This feature is used in the transformer $S^3\_SR8\text{-}1$ for the resetting free of leakage currents.

2) The two-clock controlled flip-flop with reset *TT1* (*fig. 1.12*).
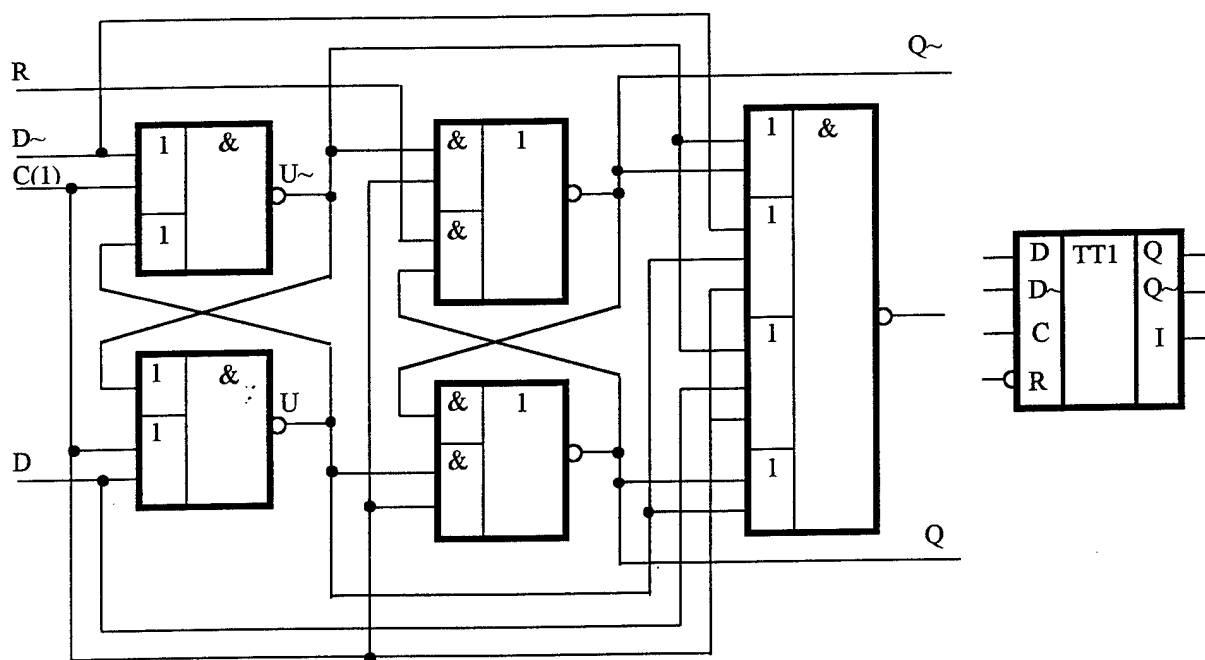
Fig. 1.12. Two-clock controlled flip-flop with reset *TT1* and its symbol

The circuit in [1.6] is modified with adding the reset signal R.

The flip-flop stores one information bit an entire next cycle: during both the spacer and new work phase. In the work phase ($C = 0$), information is written into a first bistable cell. In the spacer ($C = 1$), information is moved from the first bistable cell to the second one. The signals D and $D\sim$ must be updated only at $C = 1$. The indicator I, common for both cells, signals completion of transition processes of a current phase in the flip-flop.

3) The two-clock controlled flip-flop with reset *TT2* (*fig. 1.13*).
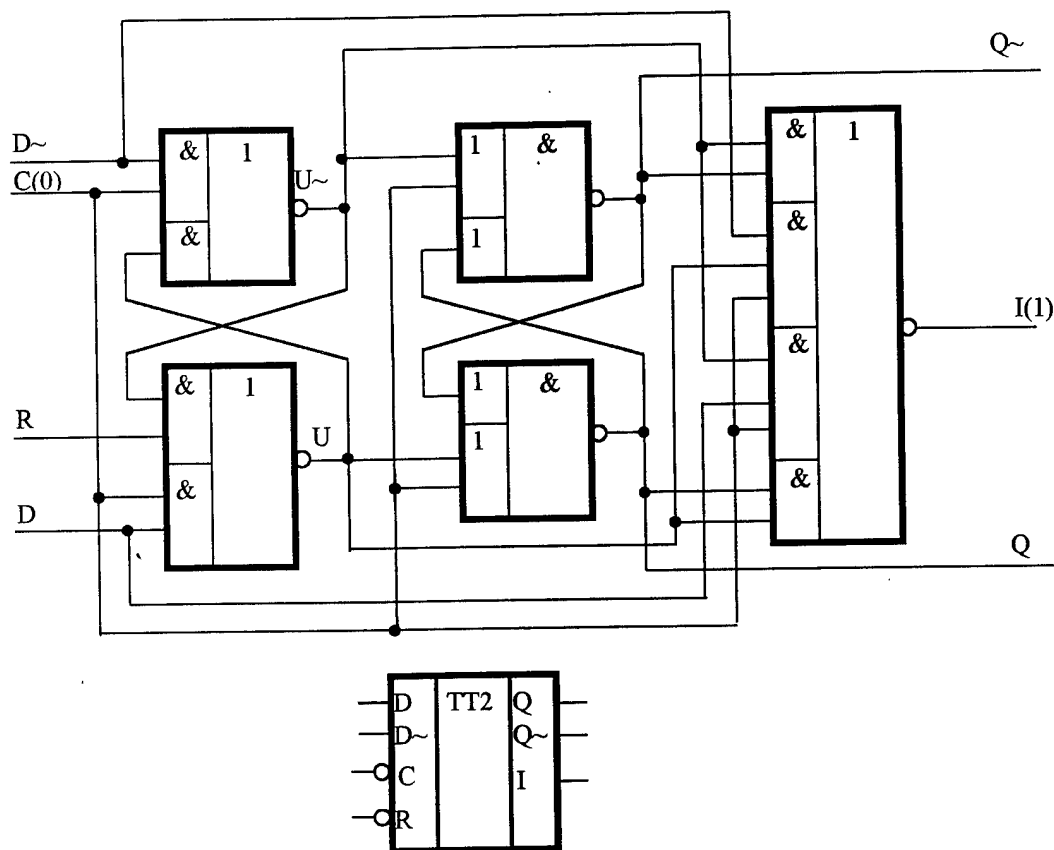
Fig. 1.13. **Two-clock controlled flip-flop with reset *TT2* and its symbol**

This flip-flop differs from the flip-flop *TT1* by the values of the signals *C* and *I* that are opposite to the corresponding values in *TT1* within both phases. There are also some differences in the way of resetting.

4) The *Q*-element (*fig. 1.14*).
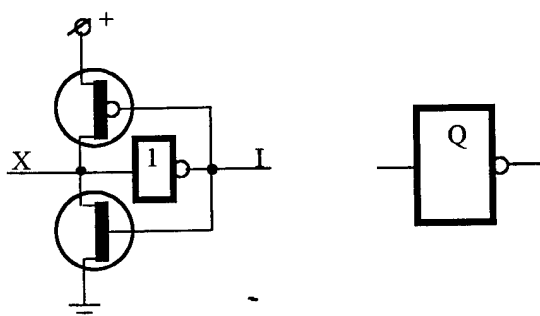
Fig. 1.14. ***Q*-element and its symbol**

The circuit is designed for storing a previous value of the input signal I in the case of high impedance on the input X. The circuit functions as an inverter in all other cases. The element is included into many subsequent circuits.

5) The *C*-elements of synphase signals (*fig. 1.15 ÷ 1.17*)

C-elements proposed by Muller are widely applied in the self-synchronous schemotechnique. They reduce the number of input indication signals, with any the spacer values being permitted. There are several variants of the *C*-elements. All of them function identically.

The output signal I becomes equal to 1 as soon as all indicator inputs X_1, X_2,... are set to 1. Analogously, $I = 0$ only after X_1, X_2,... are set to 0. That is, the signal I preserves its value until all input indicator values become identical.

$$I^+ = X_1 X_2 \vee I\left(X_1 \vee X_2\right)$$
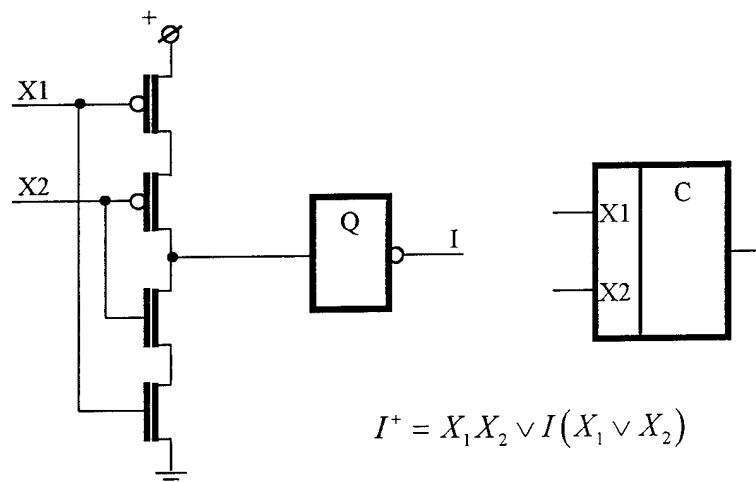
Fig. 1.15. **Two input *C*-element and its symbol**

$$I^+ = X_1 X_2 X_3 \vee I\left(X_1 \vee X_2 \vee X_3\right)$$
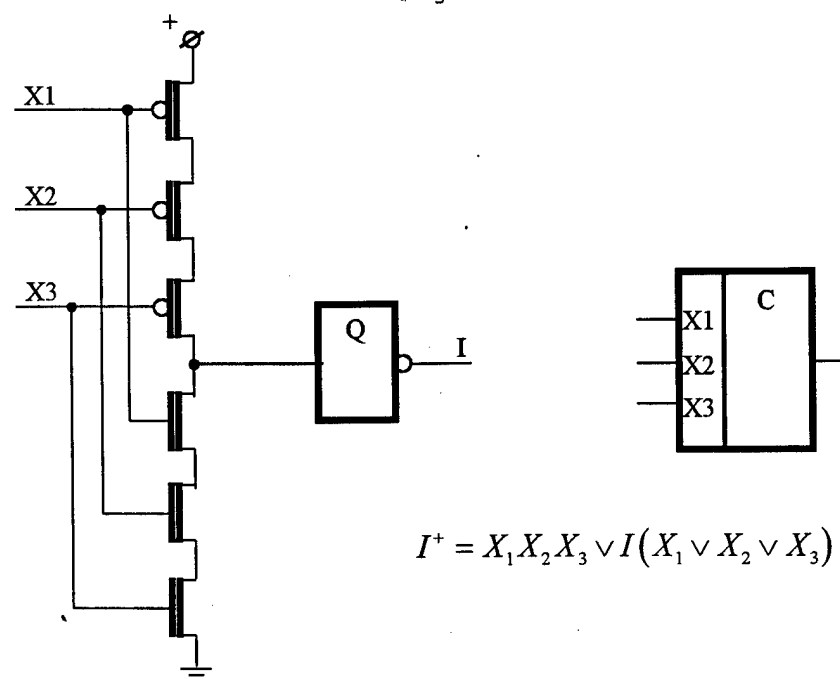
Fig. 1.16. Three input *C*-element and its symbol

$$I^+ = X_1 X_2 X_3 X_4 \vee I\left(X_1 \vee X_2 \vee X_3 \vee X_4\right)$$
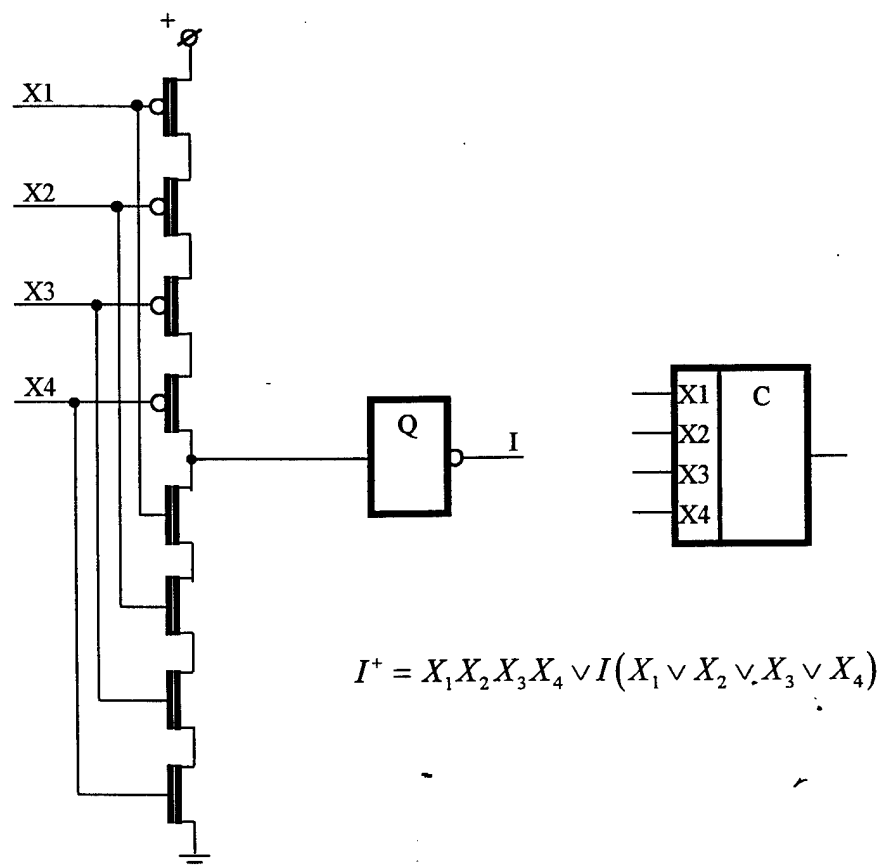
Fig. 1.17. Four input *C*-element and its symbol

6) The *C*-elements with reset (*fig. 1.18 ÷ 1.20*)

$$I^{+} = X_{1}X_{2} \vee I\left(X_{1} \vee X_{2}\right)R$$
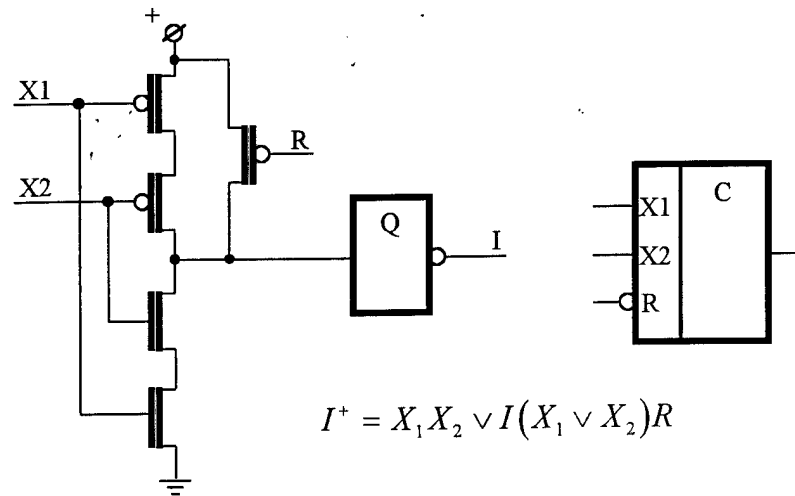
***Fig. 1.18.* Two input *C*-element with reset and its symbol**

Note.   Care should be taken to eliminate current through the element at the moment of resetting with the signal $R = 0$: $X_{1}X_{2} = 0$.

$$I^{+} = X_{1}X_{2}X_{3} \vee I\left(X_{1} \vee X_{2} \vee X_{3}\right)R$$
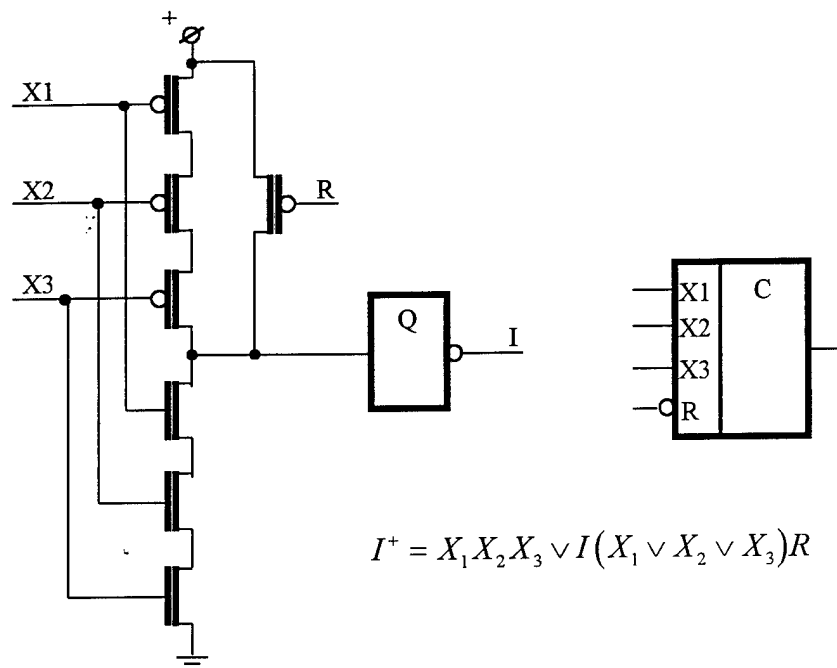
***Fig. 1.19.* Three input *C*-element with reset and its symbol**

Note.   Care should be taken to eliminate current through the element at the moment of resetting with the signal $R = 0$: $X_{1}X_{2}X_{3} = 0$.

$$I^+ = X_1 X_2 \vee I\left(X_1 \vee X_2 R\right)$$
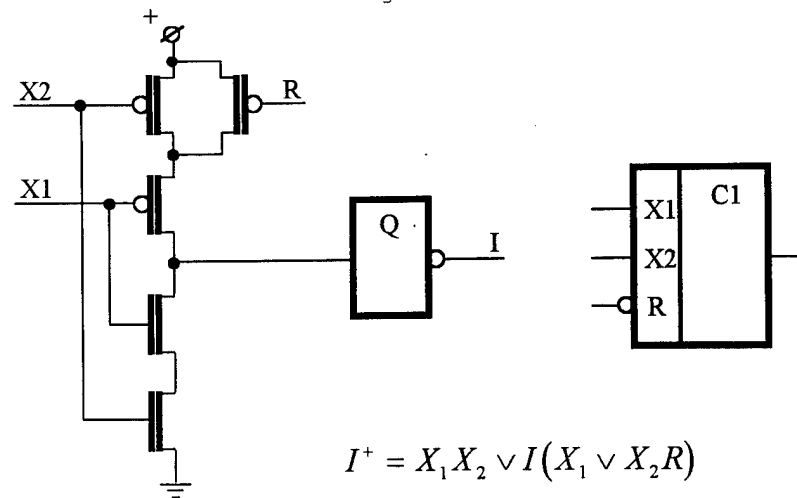
Fig. 1.20. C-element with conditional reset and its symbol

For resetting the output I to 0 with the signal $R = 0$, the condition $X_1 = 0$ must be provided.

7) The C-element of synphase and paraphase signals (*fig. 1.21*).

The element is designed for mutual indication of one synphase and one paraphase signals.

$$I^+ = \left(X_1 \vee X_1^{\sim}\right) X_2 \vee I\left(X_2 \vee \left(X_1 \vee X_1^{\sim}\right) R\right)$$
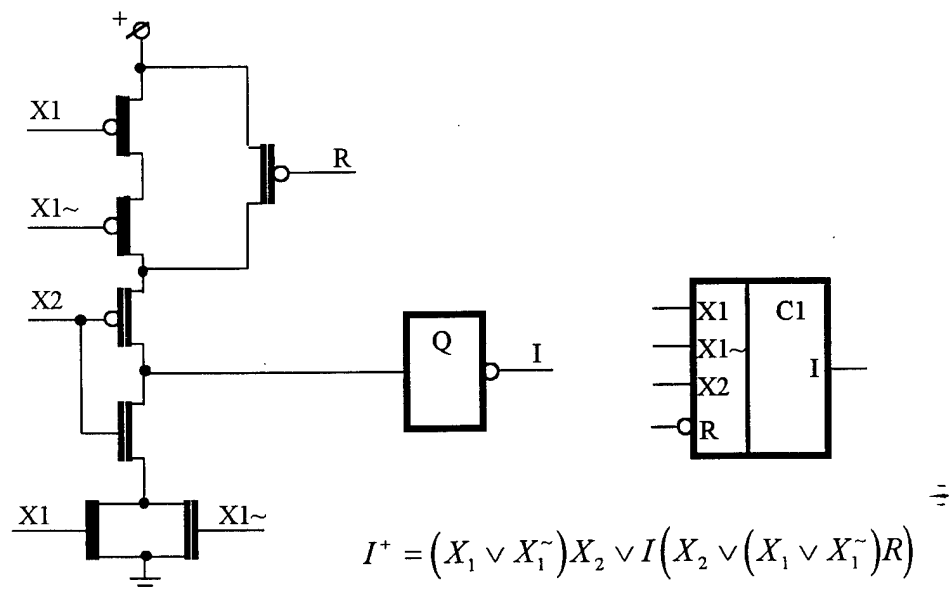
Fig. 1.21. C-element for synphase and paraphase signals and its symbol

The output signal I becomes equal to 1 when the signal X_2 and either X_1 or $X_1{\sim}$ are set to 1. At $R = 1$ the signal I becomes 0 when all inputs X_1, $X_1{\sim}$, and X_2 are set to 0. A previous value of the signal I is preserved in all intermediate states.

The condition $X_2 = 0$ must be met for resetting with the signal $R = 0$.

8) The indicator flip-flop *TC1* (*fig. 1.22*)

The circuit is designed for mutual indication of a bistable cell and one synphase signal.

The circuit operates as follows. In order to reset the output I to 0 (spacer), the inputs of the bistable cell X_1, $X_1\sim$ and the synphase signal X_3 must be set to 0 with no respect to the cell outputs X_2 and $X_2\sim$. The circuit transits to the work phase ($I = 1$) when the signal X_3 becomes equal to 1 and the outputs of the bistable cell X_2 or $X_2\sim$ are in a stable state conforming to the inputs:

$$X_2 = X_1^{\sim} = 1 \text{ or } X_2^{\sim} = X_1 = 1.$$

$$I^+ = \left(X_1 X_2^{\sim} \vee X_1^{\sim} X_2\right)X_3 \vee I\left(\left(X_1 X_2^{\sim} \vee X_1^{\sim} X_2\right)R \vee X_3\right)$$
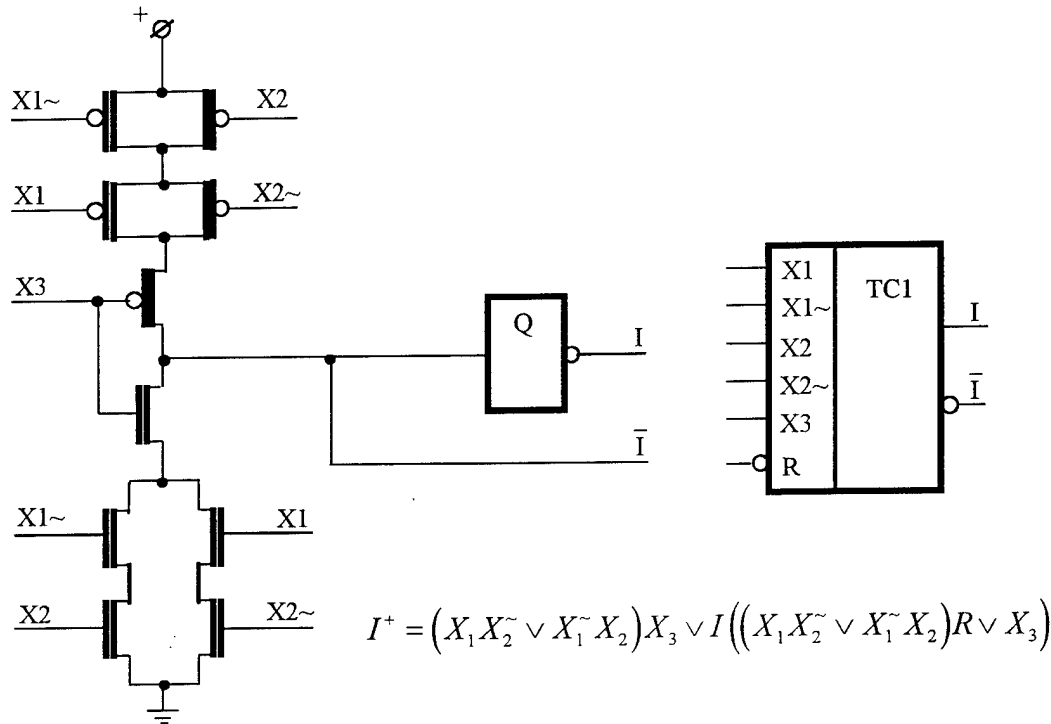
Fig. 1.22. Indicator flip-flop *TC1* and its symbol

9) The indicator flip-flop *TC2* (*fig. 1.23*)

The circuit is designed for mutual indication of a bistable cell and one synphase signal. The circuit functions as follows. The output I becomes 1 (spacer) when both the inputs X_1, X_2 of the bistable cell and the synphase signal X_5 are set to 1s. The output I is set to 0 (work phase) when the synphase signal X_5 becomes 0 and the outputs X_3, X_4 enter a stable state conforming to the inputs:

$$X_3 = X_1 = 0 \text{ or } X_4 = X_2 = 0.$$

The condition $X_1 X_2 X_5 = 0$ must be met for resetting with the signal $R = 0$.
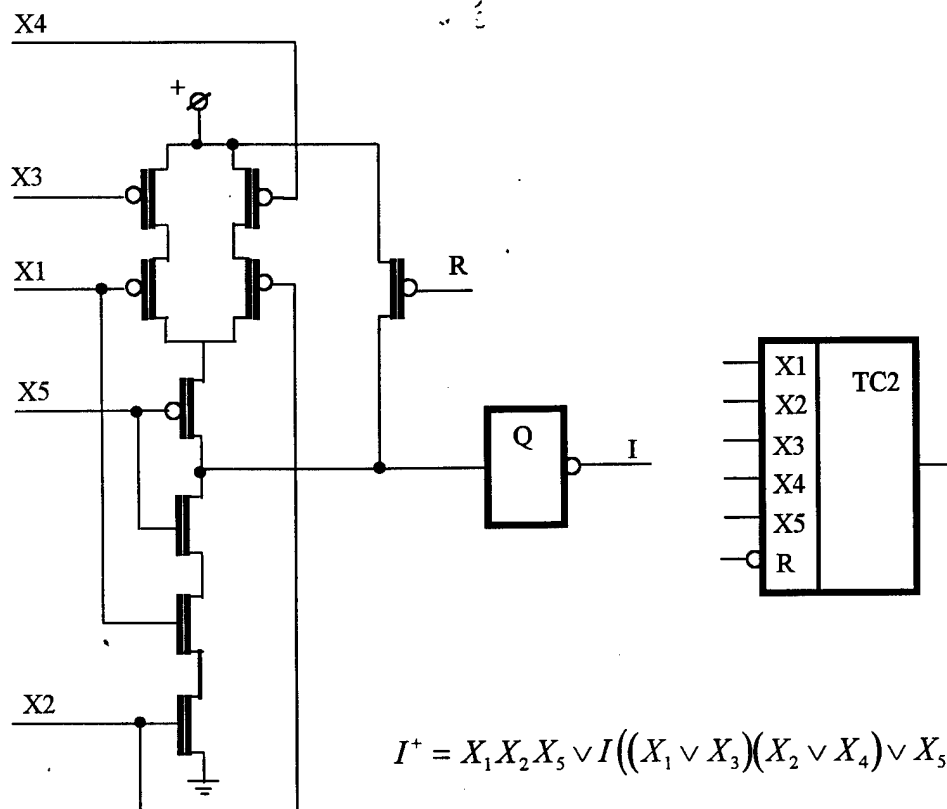
Fig. 1.23. **Indicator flip-flop *TC2* and its symbol**

$$I^{+} = X_1 X_2 X_5 \vee I \big((X_1 \vee X_3)(X_2 \vee X_4) \vee X_5 \big) R$$

10) The indicator flip-flop *TC3* (*fig.1.24*)

The circuit is designed for mutual indication of a bistable cell and two synphase signals. It has an additional input R for independent reset of the output I to 0.

The output I becomes 0 (spacer) when both input synphase signals X_1 and X_2 are set to 0s, with the inputs D, $D\sim$ and the outputs Q, $Q\sim$ of the bistable cell being in an arbitrary state. The output I becomes 1 (work phase) when the input signals X_1 and X_2 are set to 1s and the outputs Q, $Q\sim$ of the bistable cell enter a stable state conforming to the inputs $D \neq D\sim$.

Note.   The circuit with the indicator flip-flop *TC3* keeps to the self-synchronous operation discipline only if the following condition is satisfied: transition of information inputs and outputs of the indicated bistable cell from one work state to another must not pass, even intermediately, the state «11» that can force the indicator flip-flop to switch untimely to a work state.

Note.   The condition $X_1 X_2 (D \vee D\sim) = 0$ must be satisfied to prevent detrimental current through the flip-flop at resetting with the input R.

An application example is indication of an output bit in the *undense pipeline shift register* S^3_FIFO8-2.

Fig. 1.24. Indicator flip-flop *TC3* and its symbol

$$I^+ = \left(DQ \vee D^{\sim}Q^{\sim}\right)X_1 X_2 \vee I\left(X_1 \vee X_2\right)R$$

11) The indicator flip-flop *TC4* (*fig. 1.25*)

$$I^+ = \left(DQ \vee D^{\sim}Q^{\sim}\right)X_1 X_2 \vee S \vee I\left(X_1 \vee X_2\right)$$
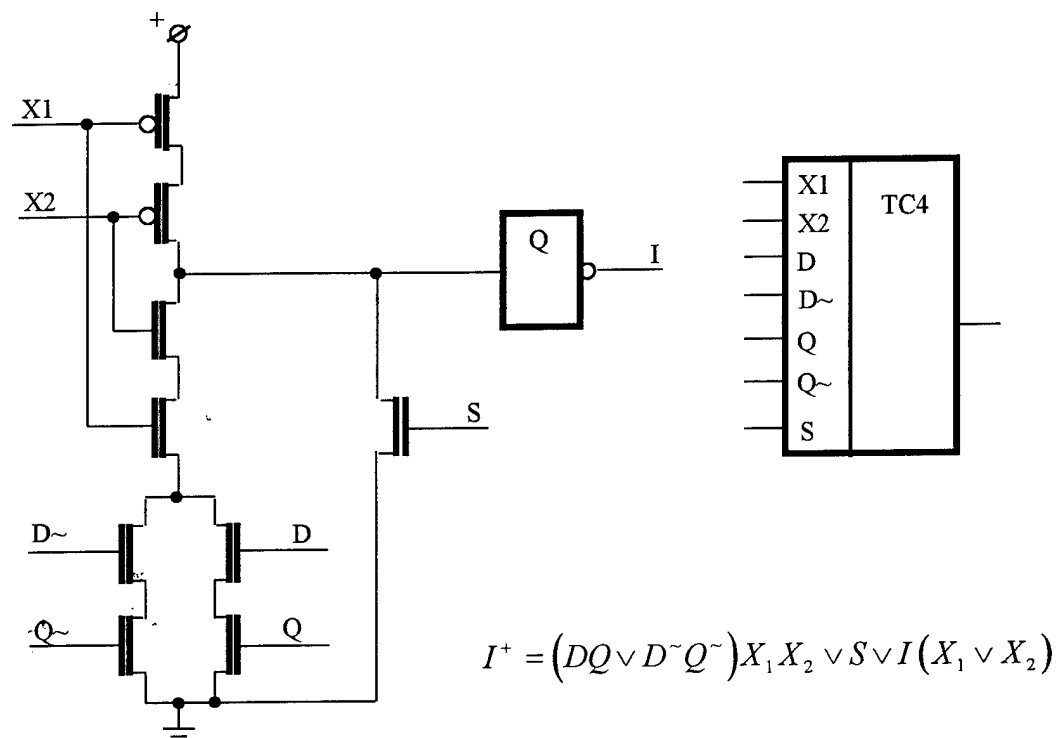
Fig. 1.25. Indicator flip-flop *TC4* and its symbol

The circuit is designed for mutual indication of a bistable cell and two synphase signals. It has an additional input S of independent initial preset $I = 1$. The circuit is functionally analogous to the indicator flip-flop $TC3$ and differs solely by the initial reset type.

The circuit operates as follows. The output I becomes 0 (spacer) when both synphase signals X_1 and X_2 are set to 0, with the inputs D, $D\sim$ and the outputs Q, $Q\sim$ of the bistable cell being in an arbitrary state. The circuit transits to the work state ($I = 1$) when the signals X_1 and X_2 are set to 1s and the outputs Q, $Q\sim$ of the bistable cell enter a stable state conforming to the inputs $D \neq D\sim$.

Note.   The circuit with the indicator flip-flop $TC4$ keeps to the self-synchronous operation discipline only if the following condition is satisfied: transition of information inputs and outputs of the indicated bistable cell from one work state to another must not pass, even intermediately, the state «11» that can force the indicator flip-flop to switch untimely to a work state.

Note.   The condition $X_1 \vee X_2 = 1$ must be satisfied to prevent detrimental current through the flip-flop at presetting with the input S..

An application example is indication of an output bit in the *dense pipeline shift register* S^3_FIFO8-4.

12) The indicator flip-flop $TC5$ (*fig. 1.26*)

The circuit is designed for mutual indication of a bistable cell and three synphase signals. It has an additional input R of independent initial reset $I = 0$.

The circuit operates as follows. The output I becomes 0 (spacer) when all three synphase signals X_1, X_2, and X_3 are set to 0s, with the inputs D, $D\sim$ and the outputs Q, $Q\sim$ of the bistable cell being in an arbitrary state. The circuit transits to the work state ($I = 1$) when the signal X_3 is set to 1 and the outputs Q, $Q\sim$ of the bistable cell enter a stable state conforming to the inputs $D \neq D\sim$.

Note.   The circuit with the indicator flip-flop $TC5$ keeps to the self-synchronous operation discipline only if the following condition is satisfied: transition of information inputs and outputs of the indicated bistable cell from one work state to another must not pass, even intermediately, the state «11» that can force the indicator flip-flop to switch untimely to a work state.

$$I^{+} = \left(DQ \vee D^{\sim}Q^{\sim}\right)X_{3} \vee I\left(X_{1} \vee X_{2} \vee X_{3}\right)R$$
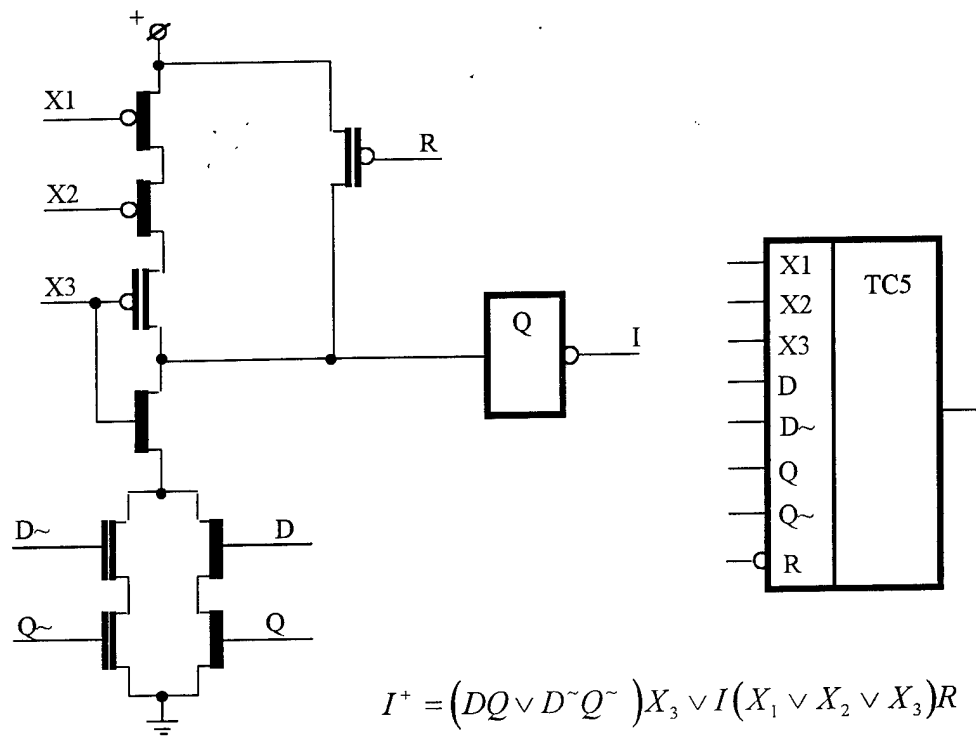
Fig. 1.26. **Indicator flip-flop *TC5* and its symbol**

<u>*Note.*</u>   The condition $X_{3} \wedge (D \vee D^{\sim}) = 0$ must be satisfied to prevent detrimental current through the flip-flop at resetting with the input R. It means preserving of the information inputs D and $D^{\sim}$ in the spacer state «00» (e.g. in respect to a serial input bit of the shift register).

An application example is indication of an input bit in the *pipeline register* S^{3}_FIFO8-4.

13) The indicator flip-flop *TC6* (*fig. 1.27*)

The circuit is designed for mutual indication of a bistable cell and three synphase signals. It has an additional input S of independent initial preset $I = 1$. The circuit is functionally analogous to the indicator flip-flop *TC5* and differs solely by the initial preset type.

Note.   The condition $X_1 \vee X_2 \vee X_3 = 1$ must be satisfied to prevent detrimental current through the flip-flop at presetting with the input $S = 1$.

An application example is indicators of internal bits in the *pipeline register* $S^3\_FIFO8\text{-}4$.

$$I^+ = \left(DQ \vee D^\sim Q^\sim\right)X_3 \vee S \vee I\left(X_1 \vee X_2 \vee X_3\right)$$
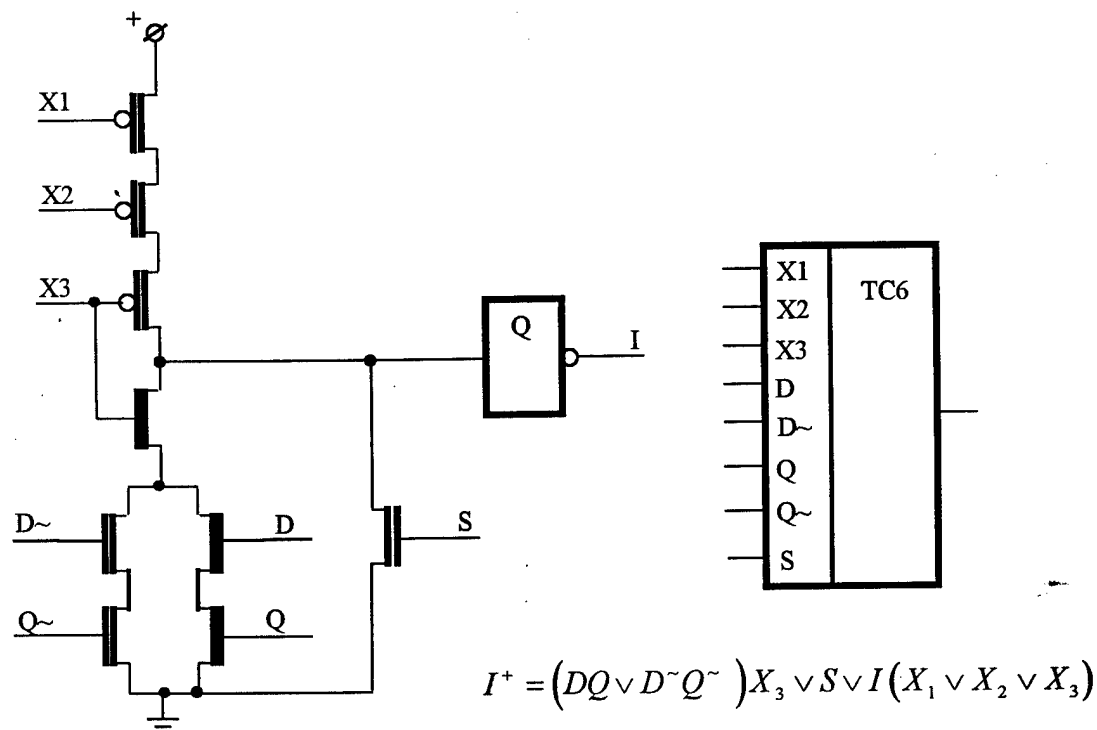
Fig. 1.27. **Indicator flip-flop *TC6* and its symbol**

14) The indicator flip-flop *TC7* (*fig. 1.28*)

The circuit of the indicator flip-flop *TC7* is the same as of the indicator flip-flop *TC5* (*fig. 3.26*), with a block of parallel inverters added on its input for driving extra loads.

$$I^+ = \left(DQ \vee D^\sim Q^\sim\right)X_3 \vee I\left(X_1 \vee X_2 \vee X_3\right)R$$
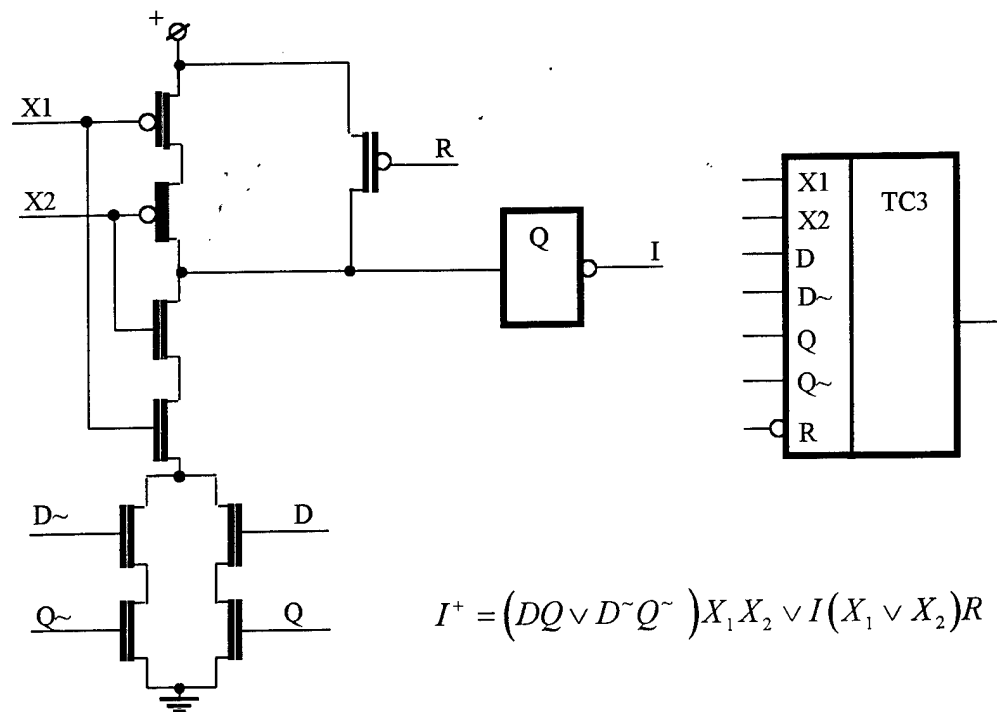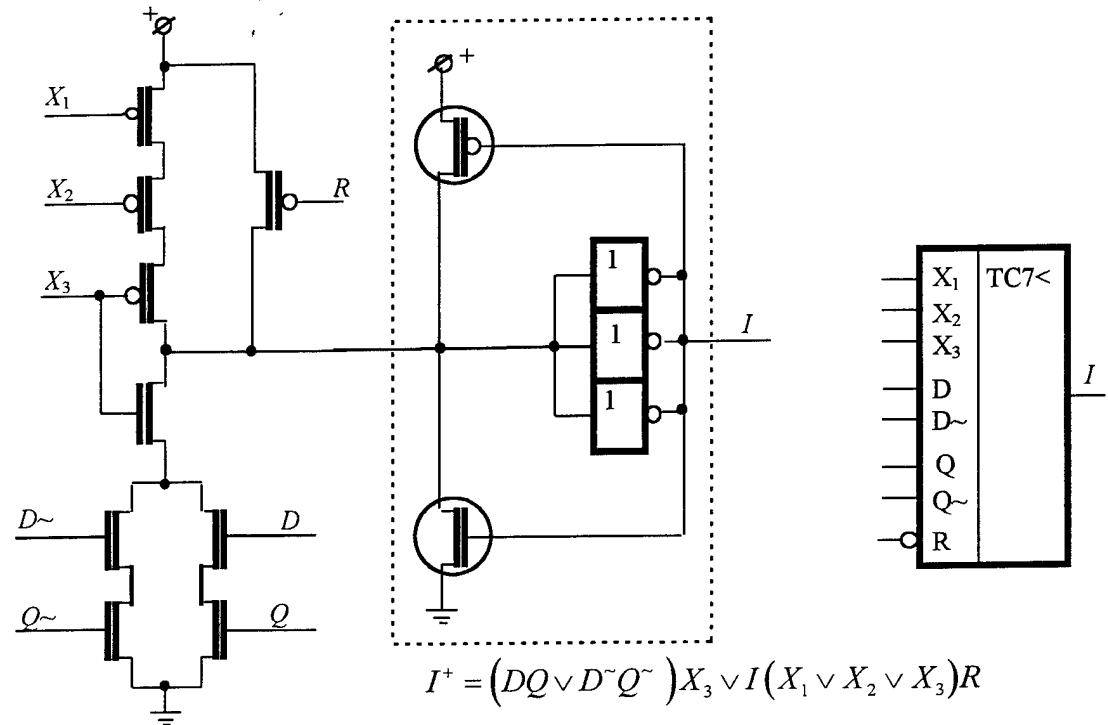
Fig. 1.28. Indicator flip-flop *TC7* and its symbol

15) The indicator flip-flop *TC8* (*fig. 1.29*)
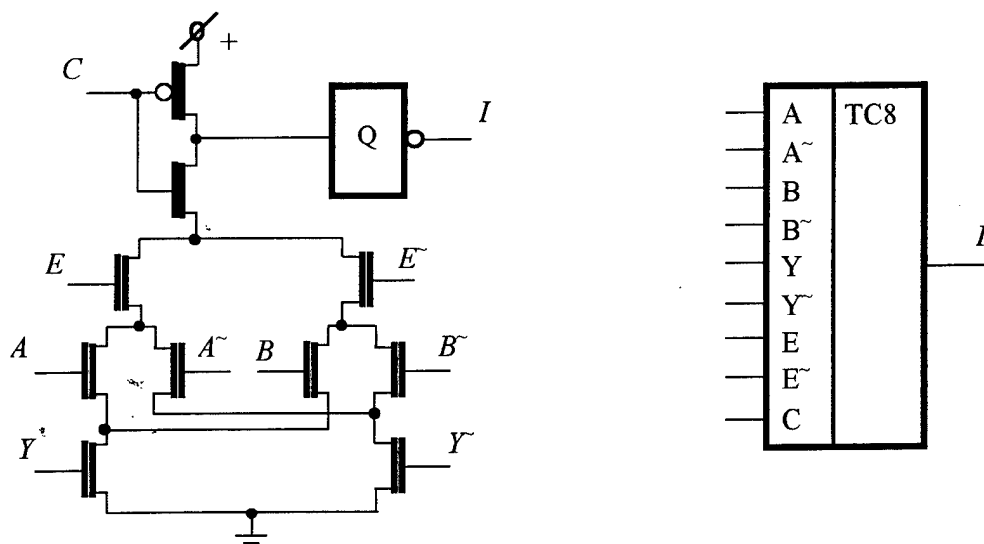
Fig. 1.29. Indicator flip-flop *TC8* and its symbol

The flip-flop is designed for indication of transition process completion in a bit of a multiplexer of register output information signals. In the spacer $(C = 0)$, I is equal to 0. In the work state $(C = 1)$, the signal I becomes 1 only after the multiplexer output signals Y, $Y^\sim$ are set in conformity with its inputs, i.e. transition processes are completed.

If the pair E and $E^\sim$ switches, it must have a zero transit state.

The flip-flop equation:

$$I^+ = C\,((A\,Y\,\acute{\cup}\,A^\sim Y^\sim)E\,\acute{\cup}\,(B\,Y\,\acute{\cup}\,B^\sim Y^\sim)\,E^\sim\,\acute{\cup}\,I\,),$$

where A, $A^\sim$ and B, $B^\sim$ are two pairs of multiplexed input signals,

C — control signal,

E, $E^\sim$ — input pair select signal,

Y, $Y^\sim$ — information outputs of a multiplexer bit.

16) **The service C-element $SC1$ (*fig. 1.30*)**

The circuit is designed for indication of four synphase signals and has an additional input R of independent initial reset $I = 0$.

In order to obtain 1 on the output I (work state) without a leakage current through the element, the inputs X_3, X_4 and at least one of the inputs X_1 or X_2 must be set to 1s. The circuit transits to the spacer $(I = 0)$ when at least one of the signals X_3 or X_4 and both X_1 and X_2 become equal to 0s.

The following condition is sufficient for resetting $I = 0$:

$(X_1 \vee X_2)R = 0$.

Elimination of detrimental leakage currents is provided if the input values meet, at any moment, the condition:

$X_3 X_4 (\bar{R} \vee \bar{X}_1 \bar{X}_2) = 0$.

An application example is state flip-flops for bits in the *pipeline register* S^3_FIFO8-4.

$$I^+ = X_3 X_4 \vee I(X_1 \vee X_2)R.$$
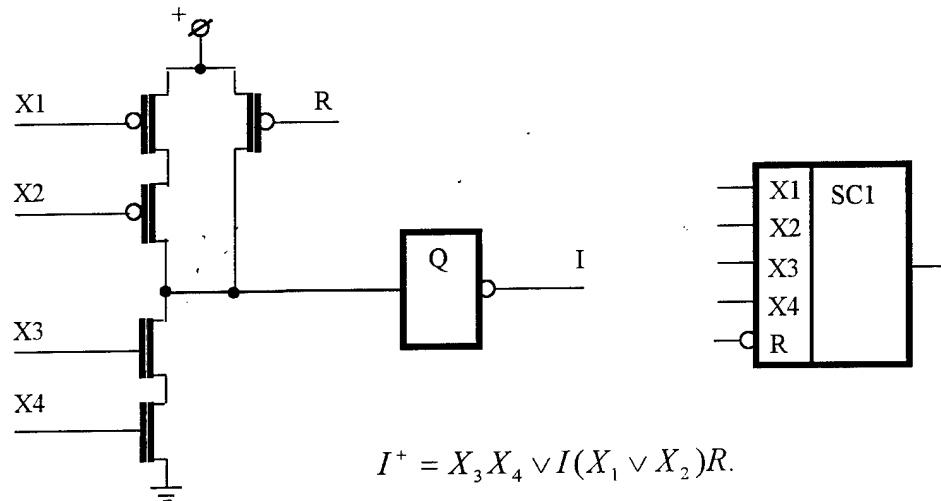
Fig. 1.30. Service *C*-element *SC1* and its symbol

17) The service *C*-element *SC2* (*fig. 1.31*)

$$I^+ = X_1 X_2 \vee I X_1 R.$$
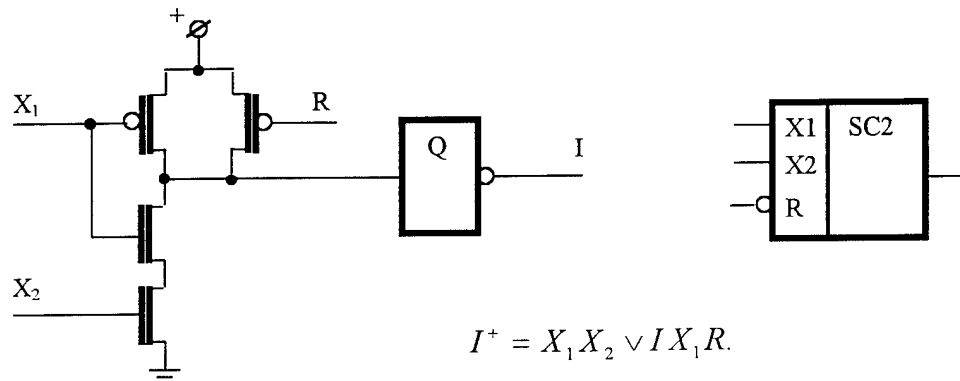
Fig. 1.31. Service *C*-element *SC2* and its symbol

The circuit is designed for indication of two synphase signals and has an additional input R of independent initial reset $I = 0$. In order to obtain 1 on the output I (work state) without a leakage current through the element, the inputs X_1 and X_2 must be set to 1. The circuit transits to the spacer ($I = 0$) when the signal X_1 becomes equal to 0.

The condition $X_1 R = 0$ is sufficient for initial resetting $I = 0$.

An application example is a state flip-flop for an output bit in the *pipeline register* S^3_FIFO8-4.

18) An intermediate bit of the undense pipeline register T2 (*fig. 1.32*)
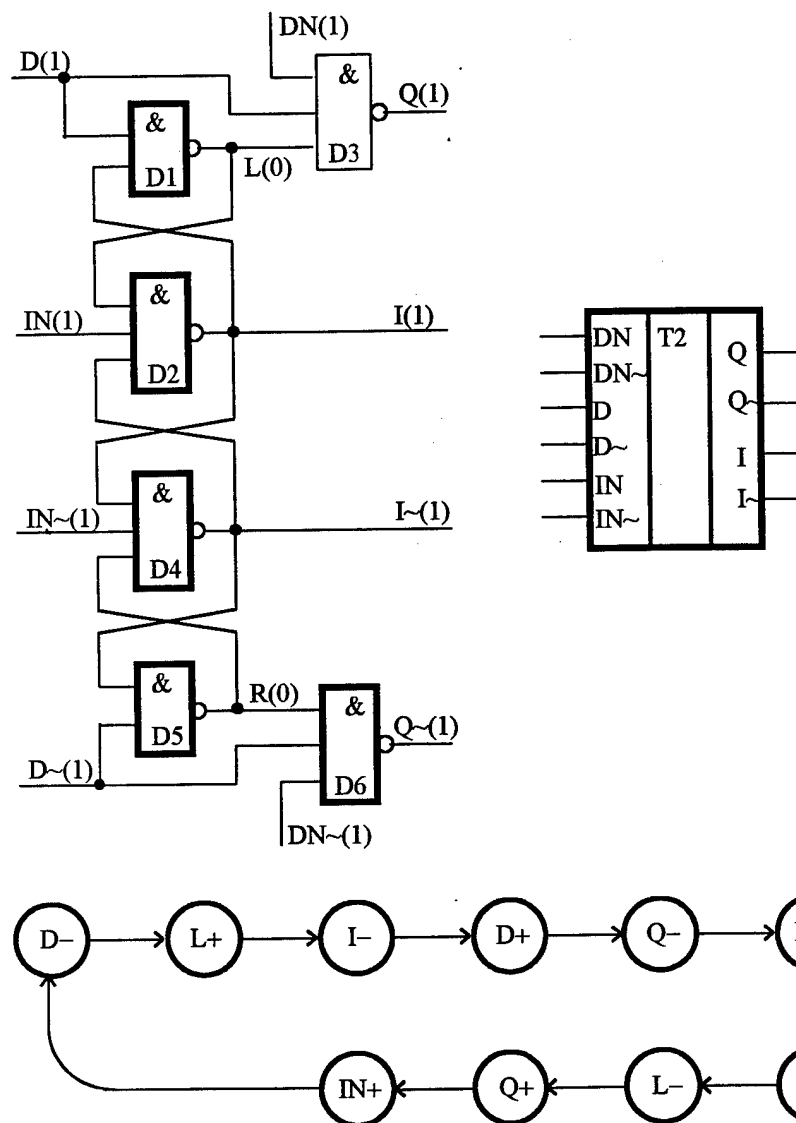
Fig. 1.32. **Intermediate bit T2 of the undense pipeline register, its symbol, and signal graph**

The circuit of the register bit was described in [1.6].

The register bit comprises two bistable cells (D1, D2) and (D4, D5) connected by cross links. A paraphase signal from a previous register bit is applied to the inputs D and $D\sim$ while signals from a next register bit — to the inputs IN, $IN\sim$, DN, and $DN\sim$.

An initial bit state is shown in the diagram. When information is input in the bit, then $D \neq D\sim$, and only one bistable cell is switched. As a response to the switching, signals, either IN or $IN\sim$ (within the last register bit both IN and $IN\sim$), are changed and lock the bistable cell remaining in the initial state. This results in shifting of stored bits along the left or right channels (depends on

the shift direction), with outrunning being eliminated. Restoring of the initial bit state occurs only after the signals D and $D\sim$ return to 1 derived from a previous bit while all other signals — from a next bit. All these transition can be observed in the signal graph, *fig. 1.32.*

An application example is an intermediate bit in the *undense pipeline register* S^3_FIFO8-1.

19) An input bit of the undense pipeline register $T3$ (*fig. 1.33*)
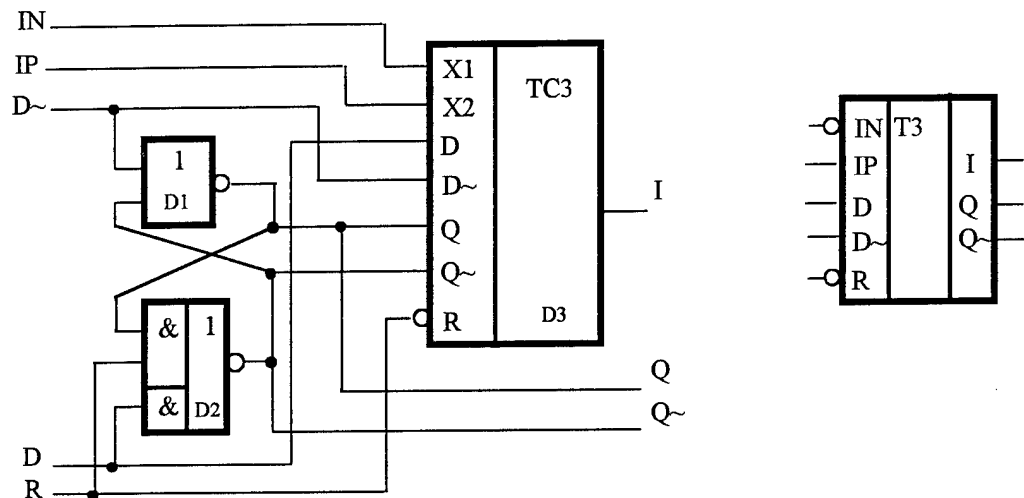
Fig. 1.33. **Input bit $T3$ of the undense pipeline register and its symbol**

The input bit $T3$ consists of a bistable cell (D1 and D2) with outputs Q and $Q\sim$ and an indicator flip-flop (D3) with an output I. A paraphase information signal is applied to the inputs D and $D\sim$, a signal from the indicator of a next bit — to the input IN, a signal from the interface to a source circuit — to the input IP. The input R (active level is $R = 0$) is used for initial resetting of the bit in the state $Q = I = 0$, $Q\sim = 1$.

The bistable cell is implemented on base of an RS flip-flop with reset; the indicator — on base of the Muller's C-element. The bistable cell stores an information bit transferred from one bit to another along the register. The bit indicator fixes completion of transition processes in the bistable cell and stores bit state information during transferring bits written into the register. The value $I = 1$ witnesses that a given bit is active and stores information. The value $I = 0$ shows a given bit is passive (empty). Observability of state of an information source (signal IP) and of a subsequent register bit (the signal IN, i.e. an inverse indicator output of a next bit) allows the register to move correctly an information bit along the pipeline register, enable/disable writing into the register, and provide simultaneously control for a source circuit.

The input bit functions as follows. At the work state on the information inputs $(D \neq D\sim)$, the circuit of the interface to a source generates the signal $IP = 1$, and the bistable cell switches to a new work state. If a next register bit is inactive $(IN = 1)$, then the indicator of the input bit transits to the work state $I = 1$. At the spacer on the information inputs $(D = D\sim = 0)$, the signal $IP = 0$ appears and switches the indicator to the spacer $I = 0$ when a next bit transits to the active (work) state $(IN = 0)$.

An application example is an input bit in the *undense pipeline register* S³_FIFO8-2.

20) A bit of the undense pipeline register T4 (*fig. 1.34*)

Fig. 1.34. **Bit *T4* of the undense pipeline register and its symbol**

The bit *T3* consists of a bistable cell (D1 and D2) with outputs Q and $Q\tilde{}$ and an indicator flip-flop (D3 and D4) with outputs I and $I\sim$. Inputs D, $D\sim$, and IP are connected to outputs of a previous bit, the input IN — to the indicator of a next bit. The input R (active level $R = 0$) is used for initial resetting of the bit in the state $Q = I = 0$, $Q\sim = I\sim$ 1, with the bistable cell of a given bit transiting to the initial state only after the bistable cell of a previous bit has switched to the initial state.

The bistable cell is implemented on base of an RS-flip-flop with reset and ability to disable information inputs at $I\sim = 0$; the indicator — on base of the Muller's C-element. The bistable cell stores an information bit transferred from one bit to another along the register and prevents any changes of its outputs while the indicator is being in the work state. The bit indicator fixes

completion of transition processes in the bistable cell and stores bit state information during shifting bits written into the register. The value $I = 1$ witnesses that a given bit is active and stores information. The value $I = 0$ shows a given bit is passive (empty, being in the spacer). Observability of state of an information source (a direct indicator output *IP*) and of a next register bit (an inverse indicator output *IN*) allows the register to move correctly an information bit along the pipeline register.

The bit *T4* functions as follows. The work state on the information inputs ($D \neq D\sim$) ensured by outputs of a previous bistable cell is always present. The bistable cell switches to a new work state at a change of the inputs D and $D\sim$ only when the indicator of the given bit is in the spacer ($I = 0$, $I\sim = 1$). If a next register bit is inactive ($IN = 1$) and a previous bit is active ($IP = 1$), then the indicator of the given bit transits to the work state $I = 1$. After transition of a previous bit to the spacer ($IP = 0$) and of a next bit to the work state ($IN = 0$), the indicator switches to the spacer $I = 0$.

An application example is an intermediate and output bit in the *undense pipeline register* S^3_FIFO8-2.

21) A cell of the dense pipeline register *T5* (*fig. 1.35*)

The circuit of the cell as well as of the whole register is described in [1.6].

The cell consists of an RS flip-flop (D1 and D2) and a tristable flip-flop (D3, D4, D5). The RS flip-flop regulates transitions of the cell from one state to another. Three states of the tristable flip-flop correspond to:

- information in the cell is absent $C = 0$, $Q = Q\sim = 1$;
- 0 is present in the cell $Q = 0$, $C = Q\sim = 1$;
- 1 is present in the cell $Q\sim = 0$, $C = Q = 1$.

Two mentioned flip-flops determine 6 cell states which encoding is represented in *table 1.2*.

Table 1.2.

	Q	$Q\sim$	C	Y	$Y\sim$
S_0	1	1	0	1	0
S_1	1	1	0	0	1
R_{a0}	0	1	1	1	0
R_{a1}	1	0	1	1	0
R_{b0}	0	1	1	0	1
R_{b1}	1	0	1	0	1
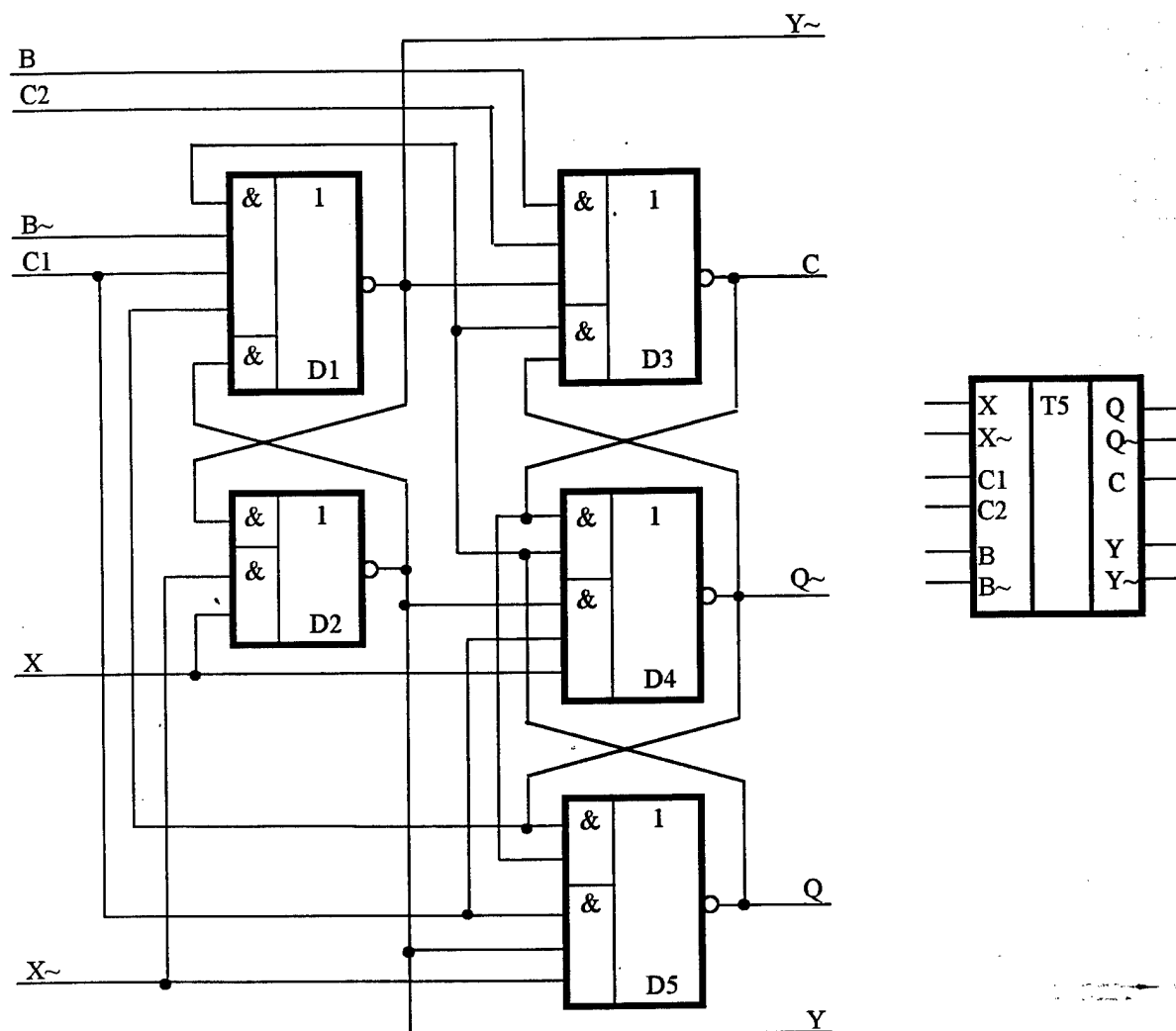
Fig. 1.35. **Bit *T5* of the dense pipeline register and its symbol**

Operation of the cell can be clarified by a transition graph in *fig. 1.36*. The output states for the graph are shown in *table 1.3* (the asterisk «*» means «don't care»).
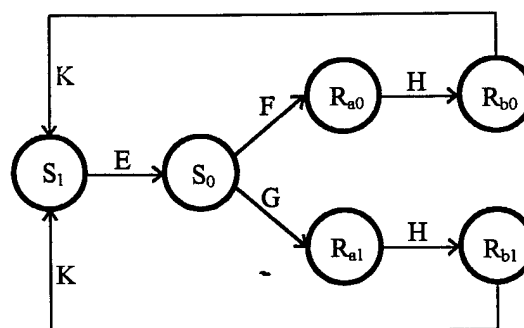
Fig. 1.36. **Transition graph for the bit *T5* of the dense register**

Table 1.3

	X	$X\sim$	$C1$	$C2$	B	$B\sim$
E	*	*	*	*	0	1
F	0	1	1	*	*	*
G	1	0	1	*	*	*
H	1	1	0	*	*	*
K	*	*	*	1	1	0

An application example is a bit of the *dense pipeline register* S^3_FIFO8-3.

22) An input bit of the dense pipeline register*T6* (*fig. 1.37*)

The input bit *T6* consists of a bistable cell (D1 and D2) with outputs Q and $Q\tilde{}$, an indicator flip-flop (D3) with an output I, a state flip-flop (D4 and D5), and an auxiliary inverter D6. The inputs D and $D\sim$ accept a paraphase signal from a source, the inputs IN, $IN\sim$ — signals from an indicator of a next bit, the input IP — a signal from the interface to the source circuit. The input R (active level $R = 1$) is used for initial resetting of the bit to the state $Q = I = 0$, $Q\sim = A = 1$.

Fig. 1.37. Input bit*T6* of the dense pipeline register and its symbol

The bistable cell is implemented on base of an RS flip-flop with reset, the indicator and state flip-flop — on base of the Muller's C-element. The bistable cell accepts and stores an information bit transferred along the register. The bit indicator fixes completion of transition processes in the bistable cell. The state flip-flop stores information on the bit state during moving on bits written in the register. The bit state $I = A = 1$ witnesses the bit is active, contains information and enables a next bit to accept it. The state $I = A = 0$ declares the bit is passive (empty). Other combinations of outputs of the bit indicator and state flip-flop are intermediate and are necessary for correct storing and transferring information in each register bit.

The input bit *T6* functions as follows. At the work state on the inputs $(D \neq D\sim)$, the register interface circuit to a source generates the signal $IP = 1$, and the bistable cell switches to a new work state. The input bit *T6* becomes ready to accept new information $(I = 0, A = 0)$ while its indicator fixes completion of transition processes in the bistable cell and transits to the state $I = 1$ that produces a «write disable» signal to the source. At the spacer on the inputs $(D = D\sim = 0)$ and after the signal $AN = 0$ is present, the state flip-flop is set to $A = 1$. On the input values $IN = 1$, $IN\sim = 0$ (i.e. an information bit is completely shifted to a next bit position), the input bit indicator transits to the state $I = 0$ (spacer) that permits the inputs D, $D\sim$ to accept a new information code.

An application example is an input bit in the *dense pipeline register* S^3_FIFO8-4.
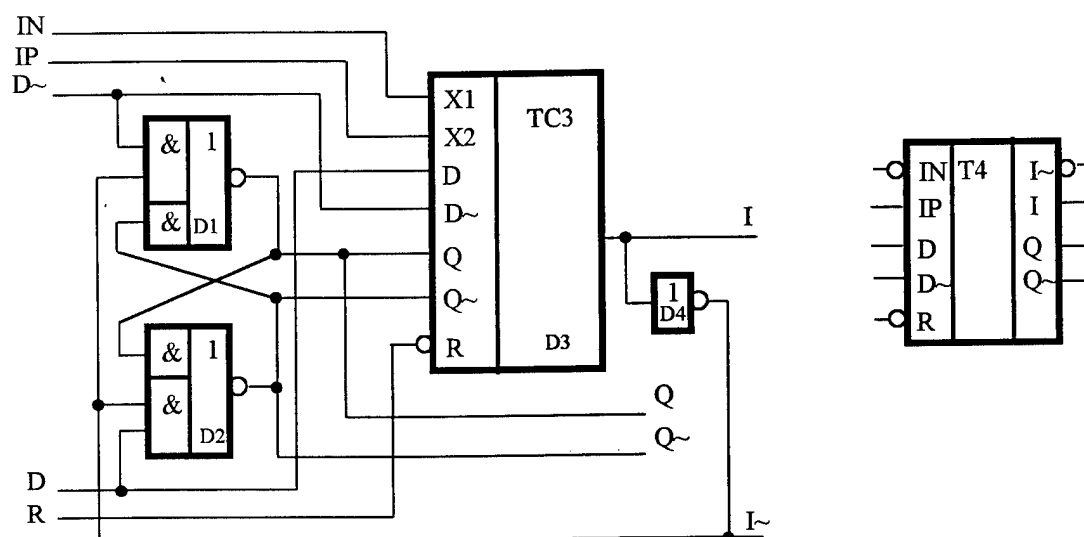
23) A bit of the dense pipeline register *T7* (*fig. 1.38*)

The bit *T7* consists of a bistable cell (D1 and D2) with outputs Q and $Q\sim$, an indicator flip-flop (D3 and D4) with outputs I and $I\sim$, a state flip-flop (D5 and D6), and an auxiliary inverter D7. The inputs D, $D\sim$, AP, IP accept signals from a previous register bit, the inputs IN, $IN\sim$ — signals from an indicator of a next bit. The input R (active level $R = 1$) is used for initial presetting of the bit to the state $Q = I = 1$, $Q\sim = A = 0$.

The bistable cell is implemented on base of an RS flip-flop with reset and ability to disable the information inputs while the bit is being in the work state, and the indicator — on base of the Muller's C-element. The bistable cell stores an information bit transferred along the register. The bit indicator fixes completion of transition processes in the bistable cell. The state flip-flop stores a bit state during shifting bits written in the register. The bit state $I = A = 1$ witnesses the bit is active (i.e. contains information), and enables a next bit to accept it. The bit state $I = A = 0$ shows the bit is inactive (empty). Other combinations of outputs of the bit indicator and state flip-flop are intermediate and are necessary for correct storing and transferring information in each bit.

Fig. 1.38. **Bit *T7* of the dense pipeline register and its symbol**

The input bit *T7* functions as follows. The work state $D \neq D\sim$ determined by outputs of a previous bit is always present on the information inputs. The bistable cell switches to a new work state as soon as the indicator output becomes $I\sim = 1$. This occurs when its inputs are set to $IN\sim = AP = 0$, with all other inputs being in an arbitrary state and the state flip-flop yielding the signal $A = 1$. The signals $IN = AN = 0$ from a next register bit switch the flip-flop to 1 ($A = 0$). The bit indicator transits to the state $I = 1$ on completion of transition processes in the bistable cell and disables the information inputs.

An application example is a bit of the *dense pipeline register* S^3_FIFO8-4.

24) An output bit of the dense pipeline register *T8* (*fig. 1.39*)

An output bit *T8* consists of a bistable cell (D1 and D2) with outputs Q and $Q^\sim$, an indicator flip-flop (D3 and D4) with the outputs I and $I^\sim$, a state flip-flop (D5, D6 and D7), and an auxiliary inverter D8. The inputs D, $D^\sim$, AP, IP accept output signals from a previous register bit, the input IN — a signal from a destination circuit. The input R (active level $R = 1$) is used for initial presetting of the bit to the state $Q = I = 1$, $Q^\sim = A = 0$.
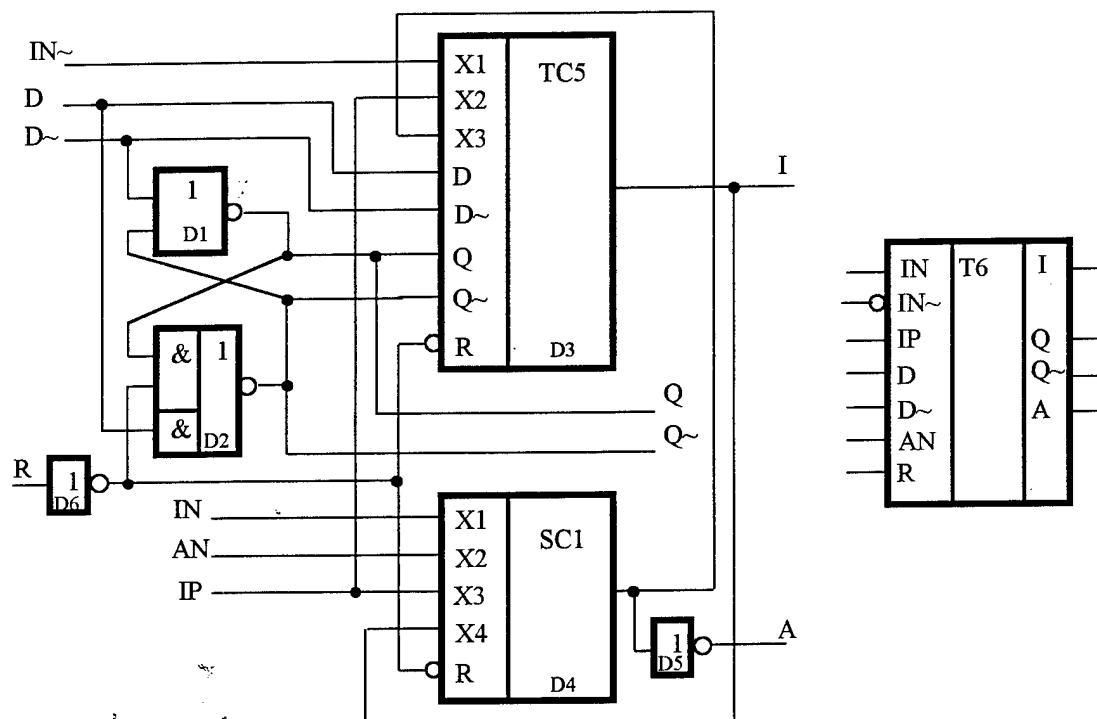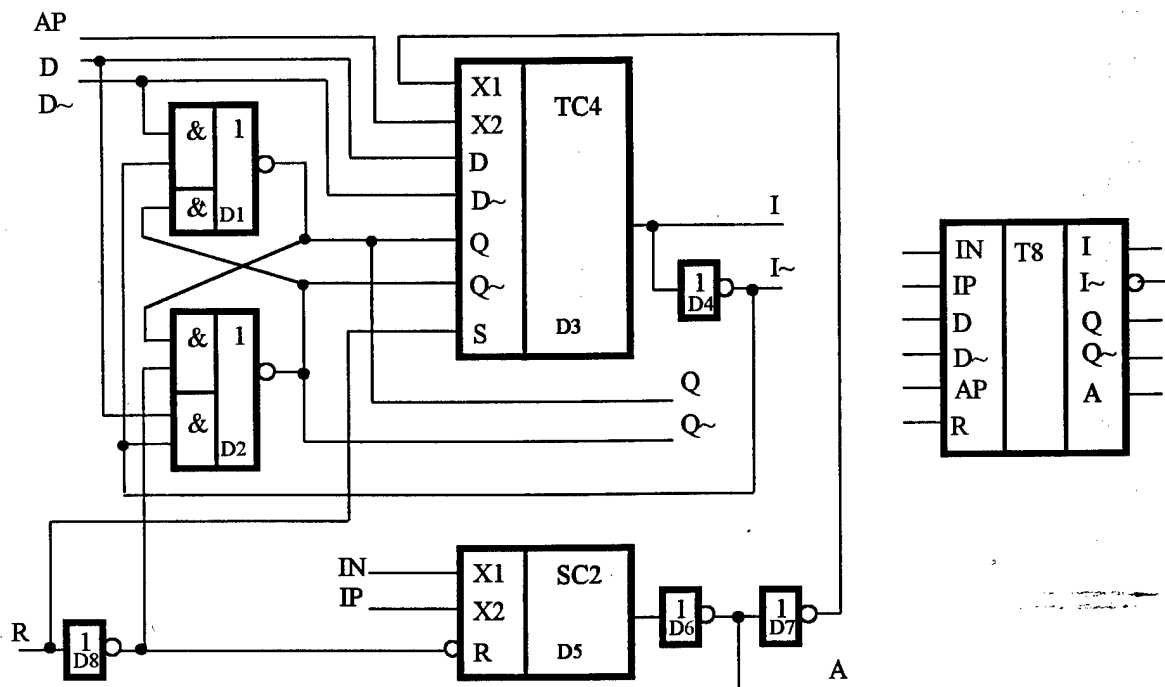

Fig. 1.39. **Output bit *T8* of the dense pipeline register and its symbol**

The bistable cell is implemented on base of an RS flip-flop with reset and ability to disable the information inputs while the bit is being in the work state, and the indicator — on base of the Muller's C-element. The bistable cell stores an information bit transferred along the register. The bit indicator fixes completion of transition processes in the bistable cell. The state flip-flop stores a bit state during shifting bits written in the register. The bit state $I = A = 1$ witnesses the bit is active (i.e. contains information) and enables a next bit to accept it. The bit state $I = A = 0$ shows the bit is inactive (empty). Other combinations of outputs of the bit indicator and state flip-flop are intermediate and are necessary for correct storing and transferring information in each bit.

The bit *T8* functions as follows. The work state $D \neq D^\sim$ determined by outputs of a previous bit is always present on the information inputs. The bistable cell switches to a new work state as

soon as the indicator output becomes $I\sim = 1$. This occurs when the input AP is equal to 0, with all other inputs being in an arbitrary state and the state flip-flop yielding the signal $A = 1$ (the input $IN = 0$). The signal $IN = 1$ from a destination circuit (that reports readiness to accept information) switches the flip-flop to 1 $(A = 0)$. As soon as a previous bit becomes ready to provide information $(AP = 1)$, the output bit indicator transits to the state $I = 1$ on completion of transition processes in the bistable cell, disables the information inputs, and reports to a destination circuit availability of a next information bit on the register output. When the destination circuit replies with the signal $IN = 0$ that indicates completion of reading the output information, the state flip-flop switches to 0 $(A = 1)$ and enables the register output bit to accept information from a previous bit.

An application example is an output bit of the *dense pipeline register* S^3_FIFO8-4.

25) A register output multiplexer bit *MX1* (*fig. 1.40*)
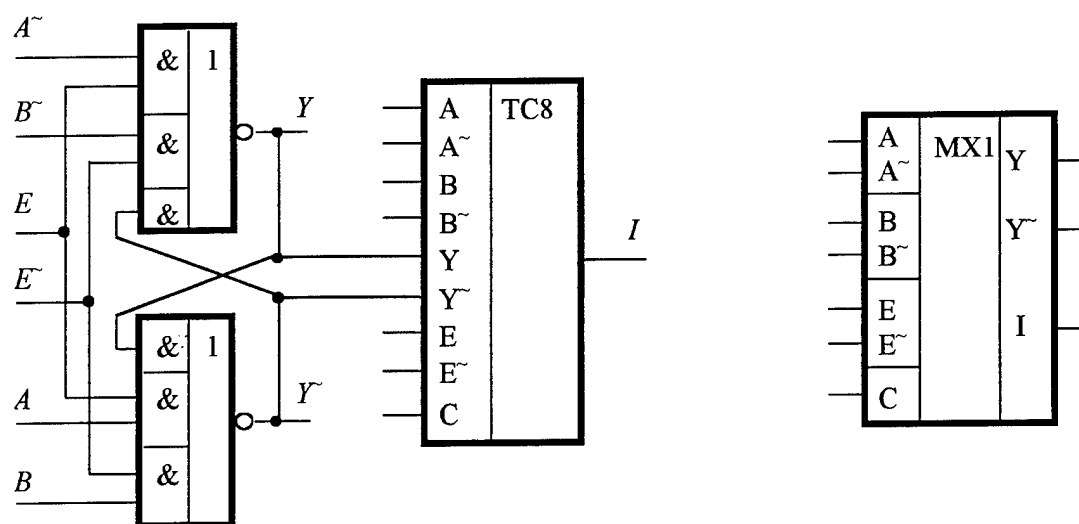
Fig. 1.40. **Multiplexer bit *MX1* and its symbol**

The circuit is designed for attaching one of the signal pairs A, $A\sim$ or B, $B\sim$ to the output Y, $Y\sim$ controlled by the paraphase select signal E, $E\sim$.

26) The 8-bit multiplexer *MX8* (*fig. 1.41*)
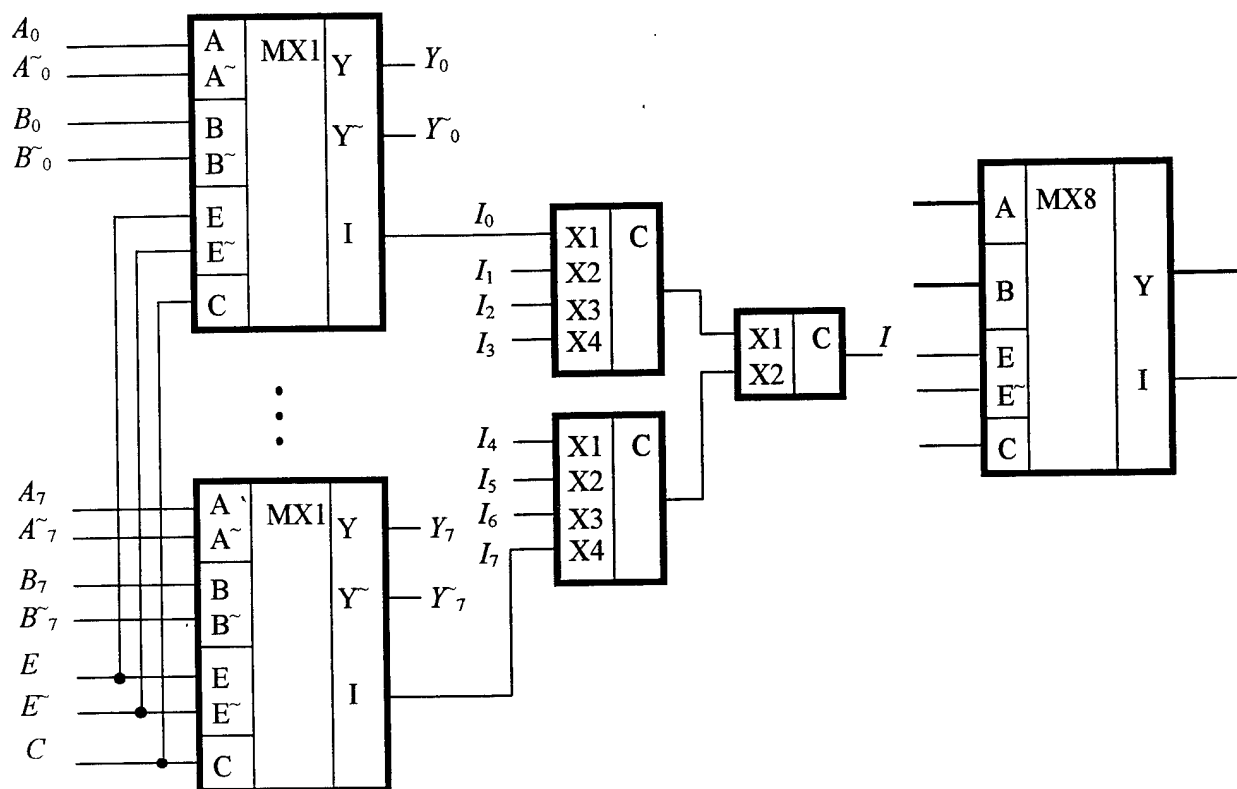
Fig. 1.41. 8-bit multiplexer *MX8* and its symbol

The circuit id designed for commutation of outputs of a basic and reserve 8-bit registers. It is a simple combination of eight multiplexer bits with a common indicator consisting of three *C*-elements.

1.3 The comparative estimation of synchronous, asynchronous, and strictly self-synchronous implementations of a test functional device

Given below are the results of simulation experiments with the purpose of obtaining quantitative characteristics of diverse CMOS implementations of the same test functional device (TFD). The experiments comprised development, for the chosen TFD, of appropriate synchronous, asynchronous, and strictly self-synchronous circuits followed by creation of their VHDL models enabling their quantitative comparisons. Further, all S^3 implementations were verified with the CAD system FORCAGE to confirm independence of their behavior of element delays.

It was suggested that the TFD had to meet the following requirements:

— to be functionally complete and applicable in practice;

— to be sufficiently simple in implementation in order to consider a number of different implementations, conventional (not redundant and not self-checkable) and fault-tolerant (redundant and self-checkable) specifically;

— to be expandable that would enable observation, for the circuits under simulation, of dependency of characteristics on, for example, the number of bits;

— to allow both stand-alone and embedded implementations on basis of a system bus (further, Bus);

— to expose clearly the use in practice of theoretical statements of the S^3 schemotechnique and specificity of the S^3 circuits design.

An additional argument was taken into consideration at choosing the TFD. Since various implementations of combination circuits were considered in the section 1.1 (with comparisons of hardware overheads) while no estimations of circuits with memory were available, it was accepted as advisable to choose a TFD comprising elements with memory — single- and double-clock flip-flops with and without control, and with and without the spacer (i.e. with different kinds of input information signals).

Resulting from consideration of various pretenders, a **serial-to-parallel converter** was chosen as the TFD. This converter will be further referred to as *Acceptor*, while a source of an input serial bit sequence — *Master*.

Each of TFD implementations was supposed to function efficiently in the range of the

following ambient conditions:

    — temperature from $-63°$ to $+63°$ C;

    — power supply voltage from $+3$ to $+7$ V; .

    — Bus fan-out from 0 to 15.

All TFD implementations were compared on speed and the number of transistors. These comparative estimations are correct for any topological and technological basis if the latter is identical for all compared implementations. As such a basis, we chose the well-investigated 3-micron CMOS technology, for which a comprehensive set of necessary topological and technological parameters had been available.

The speed characteristics given below for all TFD implementations are obtained by logic simulation in the chosen basis. These characteristics can be extrapolated to other topological norms approximately proportionally to their numeric values in microns.

Below, the following designations for the test circuits implementations will be used:

    — S_SRn — synchronous not self-checkable optimized on speed n-bit converter on base of a shift register with the serial data input;

    — S_SRn-SC — synchronous self-checkable version of S_SRn; .

    — S_SRn-FT — synchronous fault-tolerant version of S_SRn;

    — A_SRn — asynchronous not self-checkable n-bit converter on base of a shift register with the serial data input;

    — S^3_SRn-0, 1, 2, 3 — strictly self-synchronous self-checkable n-bit converters on base of a shift register with the serial data input;

    — QSS_SRn-4 — quasi-self-synchronous self-checkable n-bit converters on base of a shift register with the serial data input;

    — S^3_FIFOn-1, 2, 3, 4 — strictly self-synchronous self-checkable FIFO-converters on base of an one-bit pipeline register with the depth n;

    — S^3_SRn-3-FT — strictly self-synchronous self-checkable converter (based on S^3_SRn-3).

1.3.1   Synchronous and asynchronous not self-checkable converters

The choice of the synchronous TFD circuit was aimed at attainment of a "near ideal" circuit as a base for comparisons on both performance and hardware expenses. This approach was considered as solely correct for any comparisons.

Fig. 1.42 represents a synchronous not self-checkable converter using two clocks, T1 and T2, and *fig. 1.43* — a synchronous flip-flop. The simulation results for single-clock converters (with single- and double-edge clocking) showed their speed to be lower as compared with the double-clock variant and, besides, their efficiency not provided within the whole range of the required exploitation conditions. And so, the double-clock variant of the converter was chosen, and it will be referred to as "idealized" with regard to the accepted decisions:

— choice of the double-clock timing system supporting maximum speed;

— adjusting of timing system parameters (T1, T2 widths and deskews) to achieve maximum converter performance;

— elimination from consideration of the aggregate of destination clock drivers which reduce, as a rule, total performance;

— using of a flip-flop cell with minimum hardware and maximum speed;

— disregard for technological parameter deviations (e.g. different switching times of identical elements under the same operation conditions).

The flip-flop cell (see *fig. 1.43*) uses four bidirectional switches. Each switch consists of two conjugated transistors: *p*-type transistor T1 and *n*-type transistor T2. The inputs state C1 = 0 and C2 = 1 enables storing on the cell outputs (Q, $\overline{Q}$) of information written in earlier and setting on the 1st stage outputs (U, $\overline{U}$) of a state conforming to the input (U = D). By this, the switches VT1, VT2 and VT7, VT8 are open and two others — locked.

At C1 = 1 and C2 = 0 (the write mode) information is moved from the 1st stage to the 2nd one, and the information input D is blocked. The switches states are opposite to the described above. For ideally correct flip-flop operation, the condition C1 ≠ C2 must be provided, while a C1 value is permitted to equal to C2 value not longer than 4 ns at the top limit conditions and 14 ns — at the bottom ones. Actually, this confines the delay value for the inverter D1 between C1 and C2.

Fig. 1.42 shows simplified block-diagrams of a Master and Acceptor with no respect to mechanisms of Bus arbitration and acquirement, specific Acceptor addressing, and others. It is supposed that the connection between the Master and Acceptor is established and the Master is always ready to supply a next information bit.
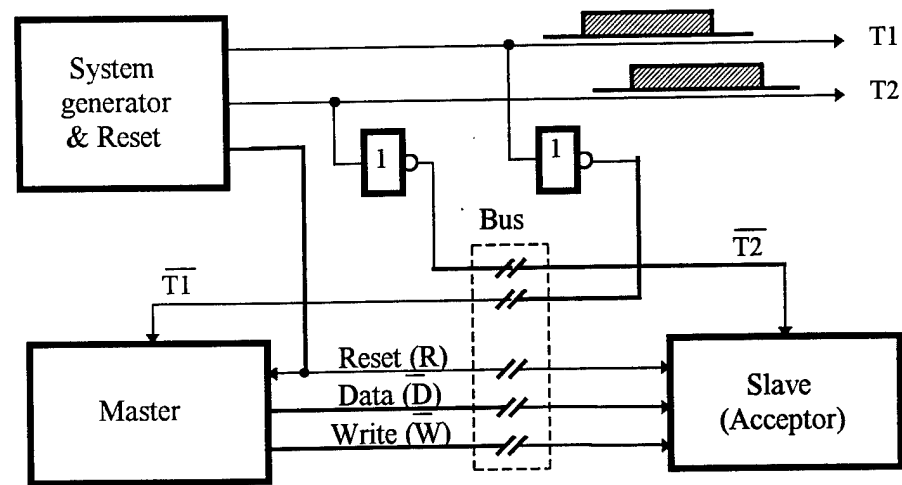
a) Interaction diagram in synchronous system
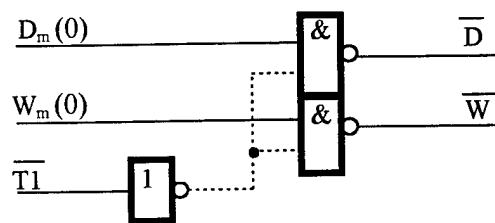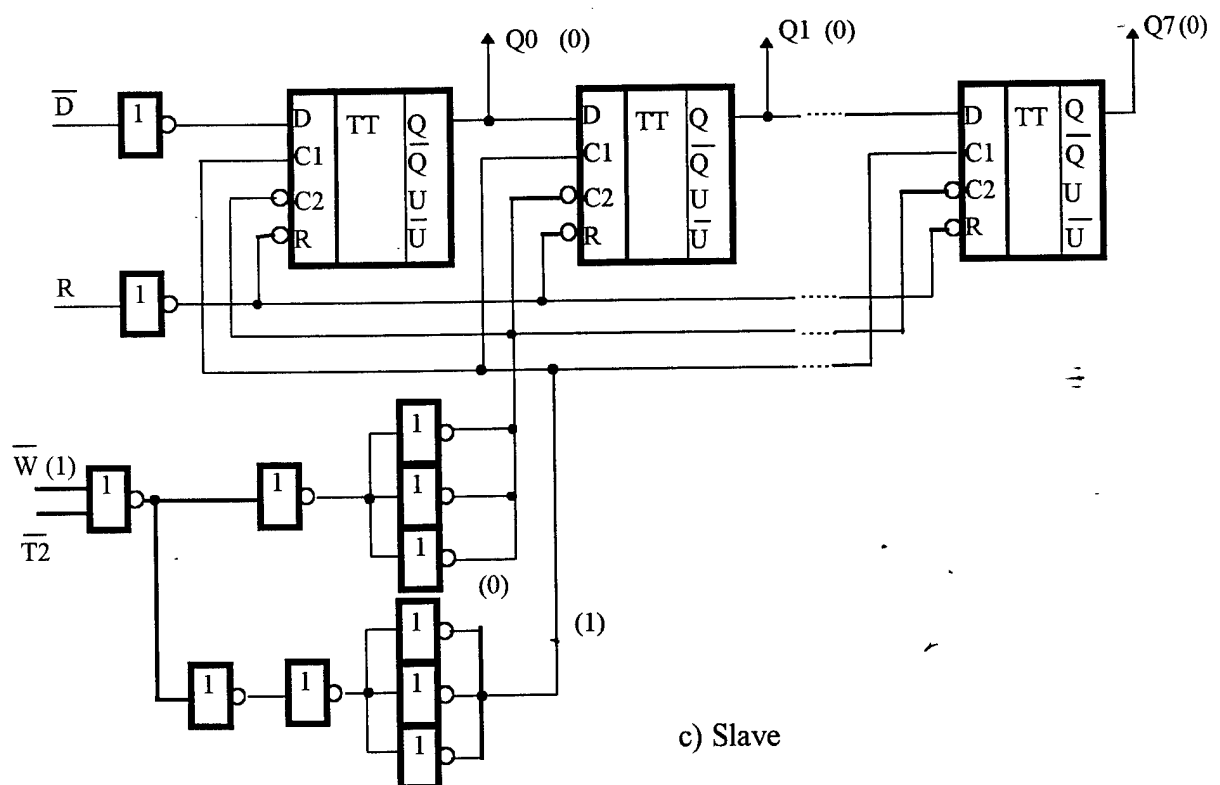
b) Master

c) Slave

Fig. 1.42. Synchronous shifter (nonredundant) S_SR8-2
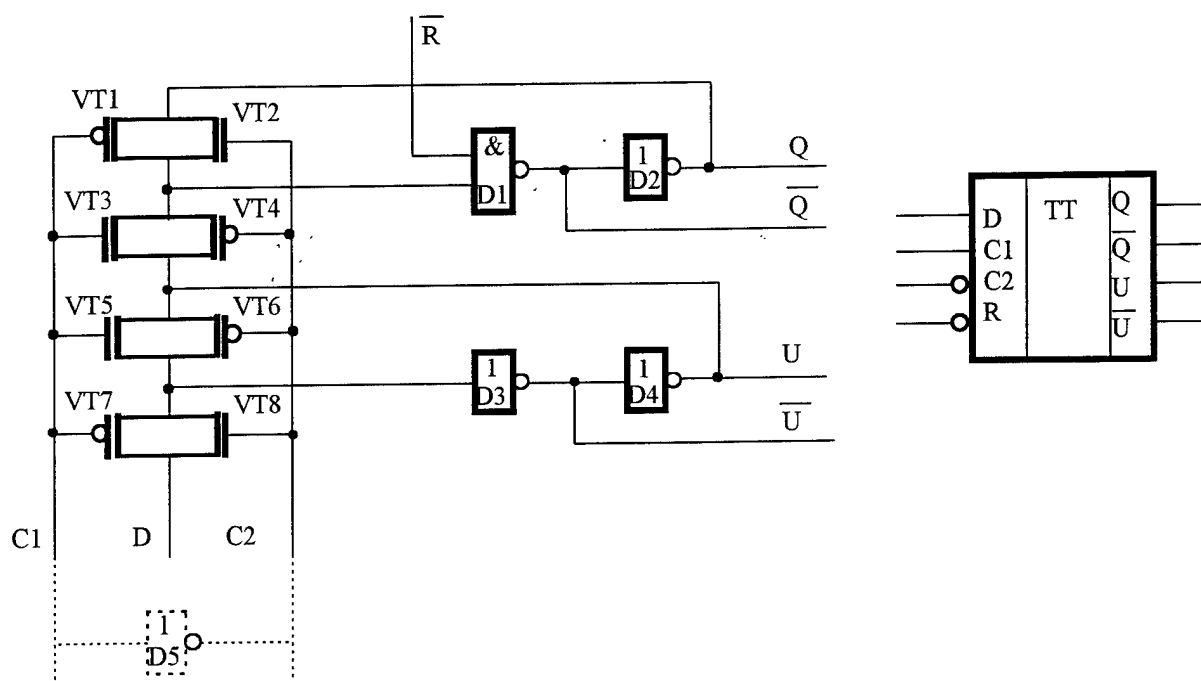
Fig. 1.43. **Synchronous master-slave flip-flop with reset**

The high level of T1 enables the Master to output information on the Data Bus accompanied by the Write signal. The converter uses the signal Write to generate the signal C1, and the positive edge of T2 — for C2. All signals between the system generator, Master, and Acceptor are regarded as Bus signals, in respect to which the following requirements must be met:

— signals are received by inverting Bus receivers;

— signals are transmitted by Bus transmitters (2-NAND or 2-NOR tri-state elements);

— each device represents a single load on the Bus (only one receiver and one transmitter are permitted).

The timing diagram for the **S_SR8** is represented in *fig. 1.44*. *Table 1* of Appendix **1.2** reflects hardware overheads for all register variants in transistors, per a bit; *Table 2* and *Table 3* of Appendix **1.2** show "actual times" of writing in a register bit and inputting an entire serial bit sequence, respectively, at normal ambient conditions.

The term "actual time" should be explained. Necessity to introduce it is derivative from a remarkable ability of S^3 circuits (and a restricted subclass of asynchronous circuits) to operate on real delays of elements and connection wires consisting in a circuit. As it was mentioned earlier, duration of transition processes in any circuit component is not limited neither from the top nor from the bottom.
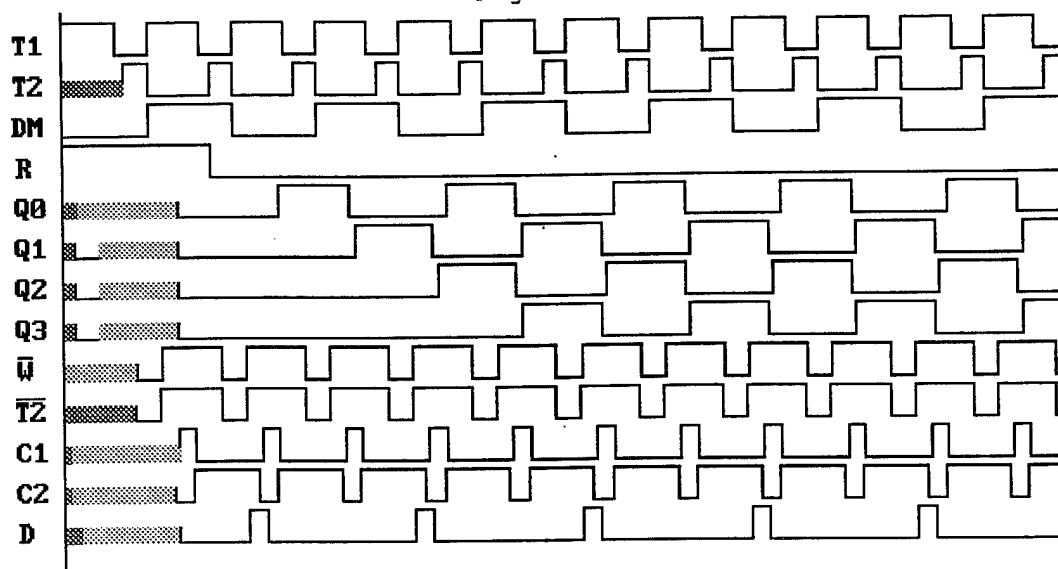
Fig. 1.44. Timing diagram for S_SR8

Completion of transition processes is determined by special self-synchronous codes and is fixed by a special indicator of transition process completion. Synchronous and most of asynchronous circuits suppose, in one way or another, some limitations on duration of transition processes both from the top and bottom in respect to an ambient conditions range, within which circuit efficiency must be provided.

Examples in *fig. 1.45* show dependencies of speed, for three implementations: one synchronous (S_SR8) and two self-synchronous (S^3_SR8-3 and S^3_FIFO8-2), on ambient conditions (temperature, power supply voltage, and the number of Bus loads). For S_SR8, each point on the curves means a maximum speed achievable in a circuit designed for these specific ambient conditions only. A consequence from the dependencies is that the speed of a synchronous converter, for any specific conditions, is higher than of fastest self-synchronous ones.

However, a synchronous device is usually implemented "for a worst case" (e.g. $T = + 125°$ C, $V_{cc} = 3V$, $N_d = 15$), and its real performance does not change at other ambient conditions. Therefore, since some point, it begins to give in the actual performance of self-synchronous implementations. The real performance of self-synchronous circuits can vary in a very wide range. For example, the actual speed (data exchange cycle duration) of the self-checkable circuit S^3_SR8-3 varies from 79 ns ($T = - 63°$ C, $V_{cc} = 7$ V, $N_d = 0$) to 471 ns ($T = + 125°$ C, $V_{cc} = 3$ V, $N_d = 15$) that is three times better and twice worse, respectively, than the actual speed of the not self-checkable synchronous circuit. Comparative estimations of functionally identical self-checkable implementations of these circuits will be given below.

T$_i$ ns

The number of devices loading the Bus (N$_d$)

Operating conditions:
(1) – 63° C, 7 V
(2) + 27° C, 5 V
(3) + 125° C, 3 V
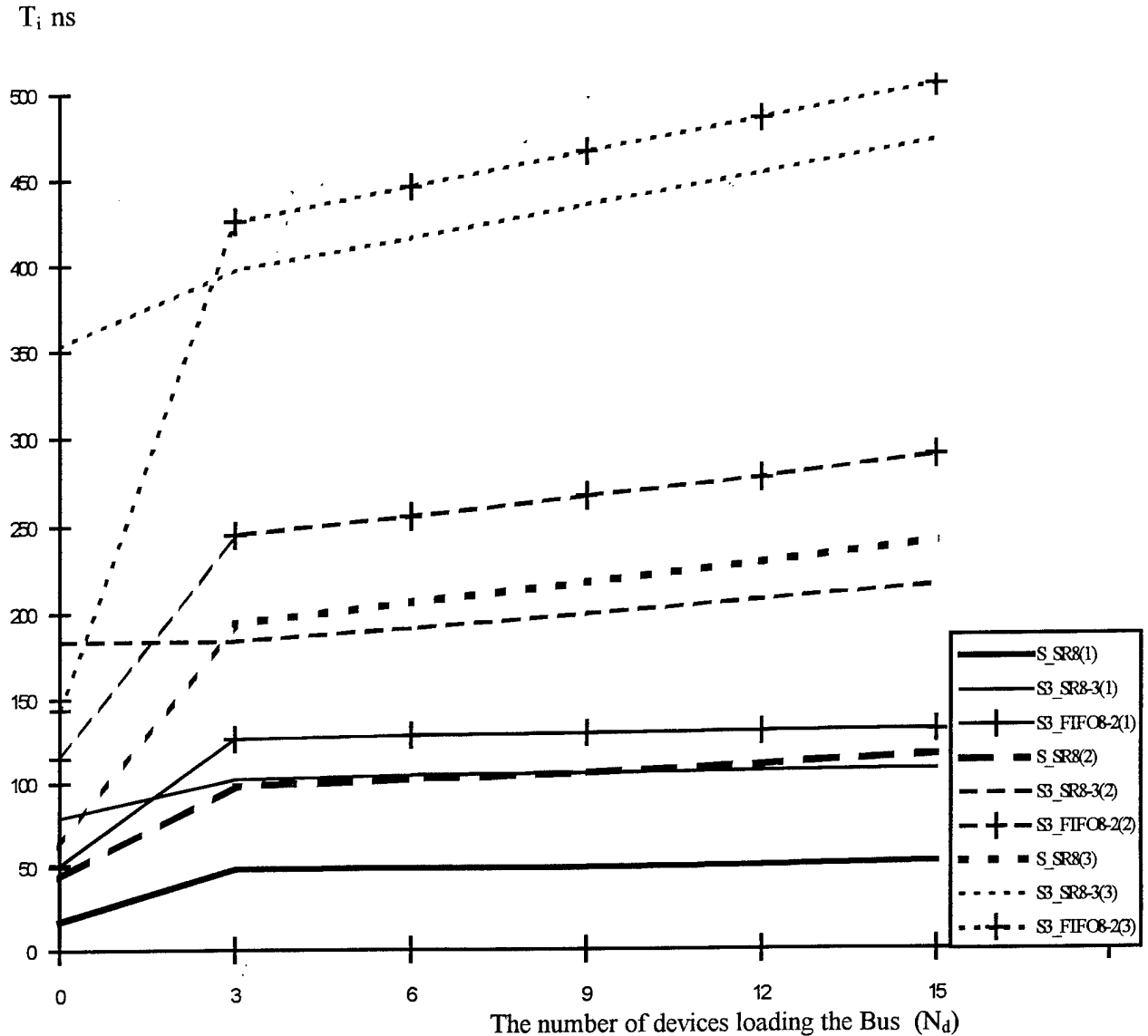
Fig. 1.45. Write time for one register bit

More two arguments for introduction of the notion "actual time" will be considered in **1.3.3.1.2** and **1.3.5**.

Further, if it will not have been stipulated specially, all comparative estimations of actual speed will be given for normal ambient conditions and six Bus loads (T = + 27° C, V$_{CC}$ = 5 V, N$_d$ = 6).

In the asynchronous converter implementation in *fig. 1.46*, interactions between the Master and Acceptor are realized by two signals of mutual synchronization in compliance with the principle "request-reply". The circuit **A_SR8** belongs to that narrow subclass of asynchronous circuits, which steady operation may be provided without any artificially introduced delays.

A delay in the Master is usually intended to compensate data propagation deskews on a parallel bus or to provide a required sequence of arriving of data or control signals to the Acceptor. As a rule, the delay performs in the Acceptor a peculiar "indicator" of transition process completion which upper limit is assigned at the device design and depends on the speed of the circuit being designed.

In our case, serial data are exchanged rather than parallel, and the delay in the circuitry of the signals of mutual synchronization is sufficient for reliable data exchange between the Master and Acceptor. This enabled elimination of predefined delays in the Master and Acceptor (Delay = 0) and, hence, expansion of the notion of the actual speed to this type of converter implementation. In fact, this statement is consequent upon absence of explicit time limitations, e.g. of a timing system or predefined delays. Correct operation of the circuit **A_SR8** is possible however only at observance of **implicit** limitations, e.g. element delays must be within limits specified by their manufacturer. This type of converter implementations can be characterized as quasi-self-synchronous and not self-checkable.

The actual speed of the **A_SR8** implementation varies from 42 ns (T = − 63 ° C, V_{CC} = 7 V, N_d = 0) to 610 ns (T = + 125 ° C, V_{CC} = 3 V, N_d = 15) that is 5.7 (1.6) times better and 2.5 (1.7) times worse, respectively, than the actual speed of the not self-checkable synchronous (self-checkable self-synchronous) implementation.

1.3.2   The synchronous self-checkable converter implementation

In this implementation, the high reliability is achieved by special self-checkable means of functional diagnostics. From a variety of means providing the self-checkability and fault-tolerance that relate essentially to application conditions and requirements of reliability, this implementation involves only a hardware redundancy. The S^3 circuits hardware redundancy enables for correctness of the comparative analysis.

At the design of a self-checkable device, the class of single constant faults was taken into consideration. Identity of source design data is thus provided in respect to the self-synchronous implementation.

a) Diagram of interaction in an asynchronous system

b) Master

c) Acceptor

***Fig. 1.46.* Asynchronous not self-checkable converter implementation A_SR8**

It should be noted that not considered in this implementation were issues related to recovery from occasional (temporal) failures.

The synchronous circuit comprises the following functional units:

— clock generator;

— serial interface;

— shift register.

Both device implementations, self-checkable and fault-tolerant, are discussed in compliance with this division.

Fig. 1.47 represents the block-diagram of the self-checkable device **S_SR8-SC** using a hardware doubling.

Synchronous operation is supported by the clock generator G forming double clock sequences T1 and T2. The clocks are driven by the buffers D5 and D6. T1 clocks the Master and T2 — the shift register.
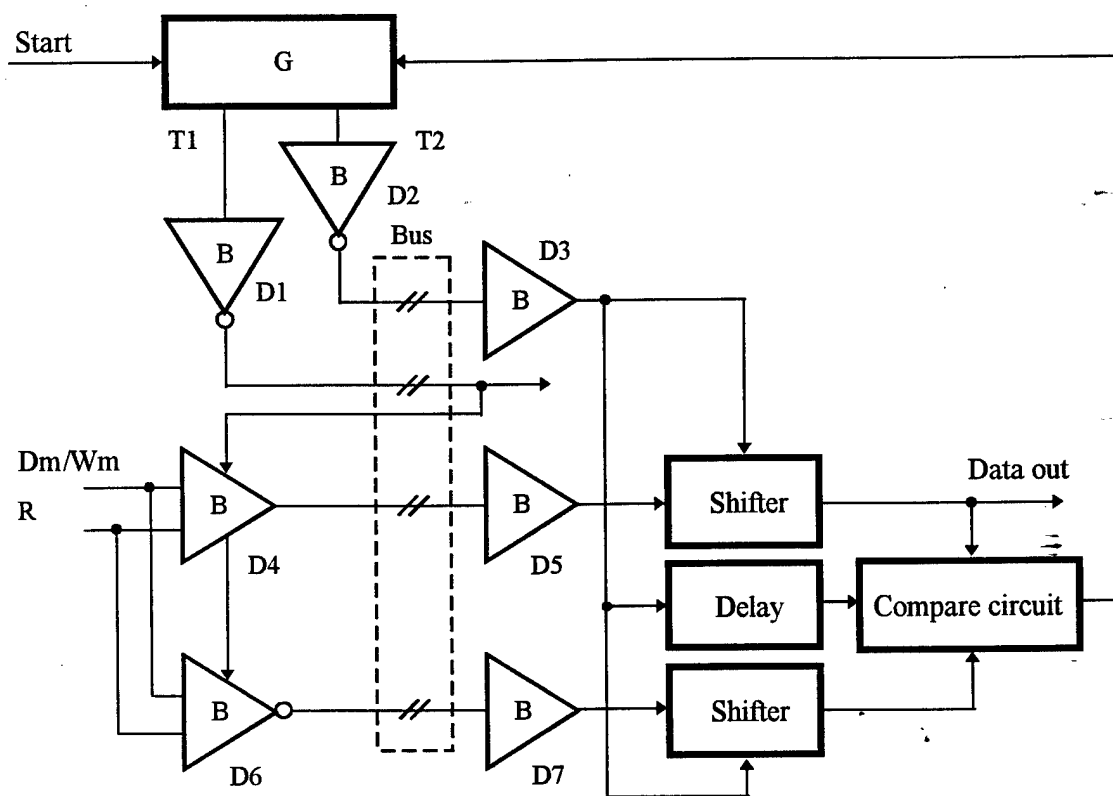
***Fig. 1.47*. Synchronous self-checkable converter implementation S_SR8-SC
Block-diagram**

The generator is a retriggerable one-shot triggered by a pulse controlled by the Comparator output.

The serial interface includes doubled connection wires. One connection wire (D4, D5) transfers the direct serial code and another wire (D6, D7) — inverse one. The doubling supports checkability of information transferred.

To provide the self-checkability, the shift register is doubled (two Shifters in *fig. 1.47*). A self-checking comparator circuit is added on the shift register outputs.

The device functions as follows. After the power is on and the device is reset (by the signal R), the generator G is enabled and starts to deliver the sequences T1 and T2. Input information comes to the inputs of the drivers D4 and D6. As the driver D6 is inverting, direct input information is delivered to the Shifter 1 while inverted — to the Shifter 2. The data transfer is clocked by T1. Outputs of both Shifters are compared by the comparator at the end of every T2 period.

The clock T2 is delayed for a time interval necessary for completion of transition processes in the Shifters and comparator. If no constant faults are revealed in the interface, Shifters, and comparator, then the delayed T2 pulse triggers the generator for a next cycle. If a constant fault is found in any of the listed units, T2 does not pass to the generator, and device operation is thus stopped. The device operation is stopped also when the clocks seize to come. It should be noted that such a checking method enables revealing of only those generator faults which are followed by termination of the clock T2. Nevertheless, simplicity of this generator check implementation is obvious.

The functional diagram of the self-checkable device S_SR8-SC (*fig. 1.48a*) comprises a retriggerable one-shot D14, two shift registers D20, D21, a comparator D22 and a device check circuit implemented with transistors VT1 ÷ VT6. The device check circuit includes, in turn, a transistor bridge (VT1 ÷ VT4) and a switch (VT5, VT6), with paraphase comparator outputs being connected to one of bridge diagonals while the gate pins of *p*- and *n*-transistors of the switch — to another.

The operation timing diagram of the self-checkable device is given in *fig. 1.49*.

After the device is powered on and reset (by a positive pulse on the Reset input), all outputs of D20 are zeros and D21 — ones, the generator G is enabled and forms pulses T1 and T2. T1 clocks data on the serial line, the rising edge of T2 enables information to enter D20, D21. The comparator D22 matches data written in D20 and D21.

a) Functional diagram

Fig. 1.48 **(1 of 4). Synchronous self-checkable converter implementation S_SR8-SC**
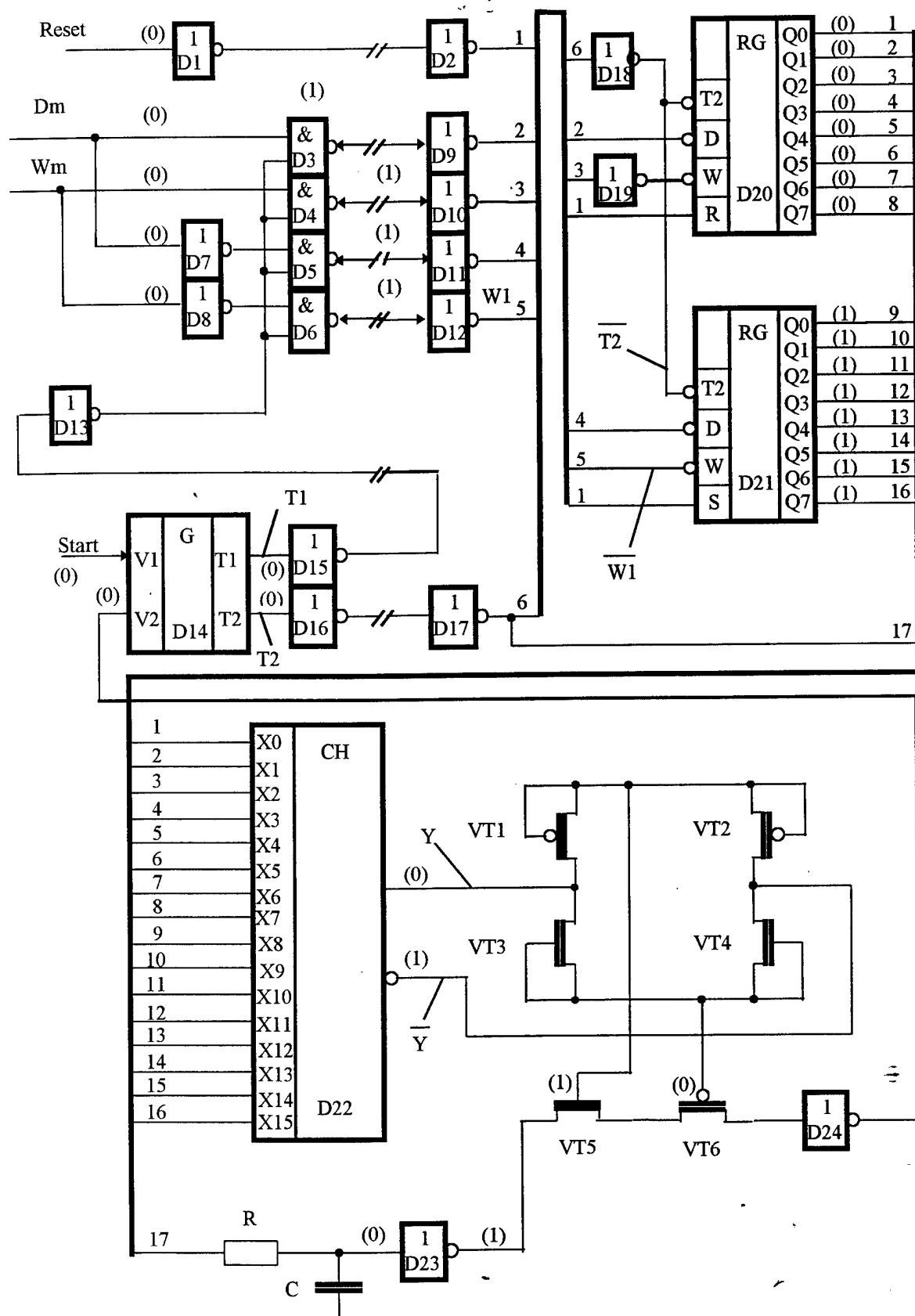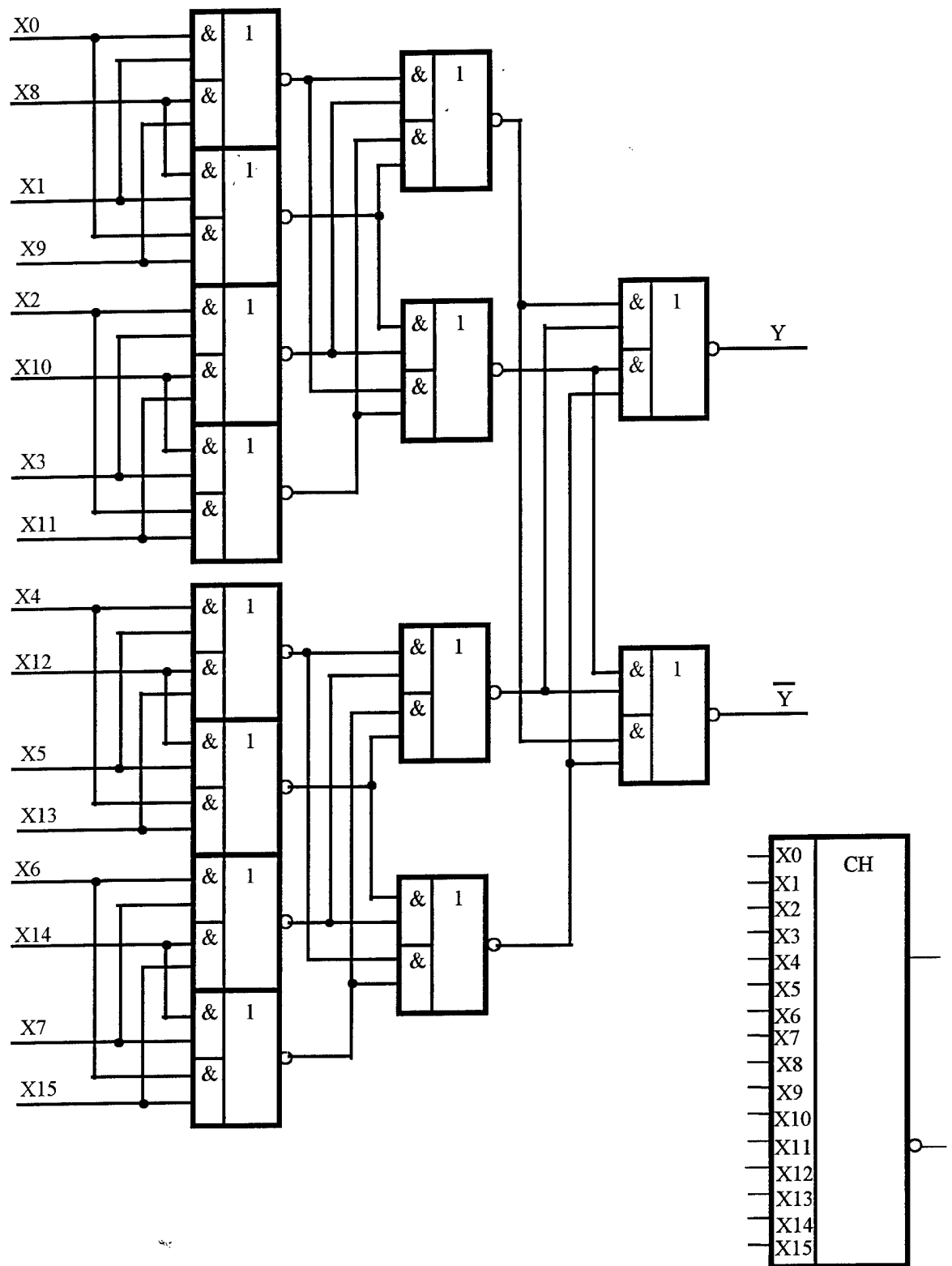
b) Shifter check circuit

***Fig. 1.48* (2 of 4). Synchronous self-checkable converter implementation S_SR8-SC**
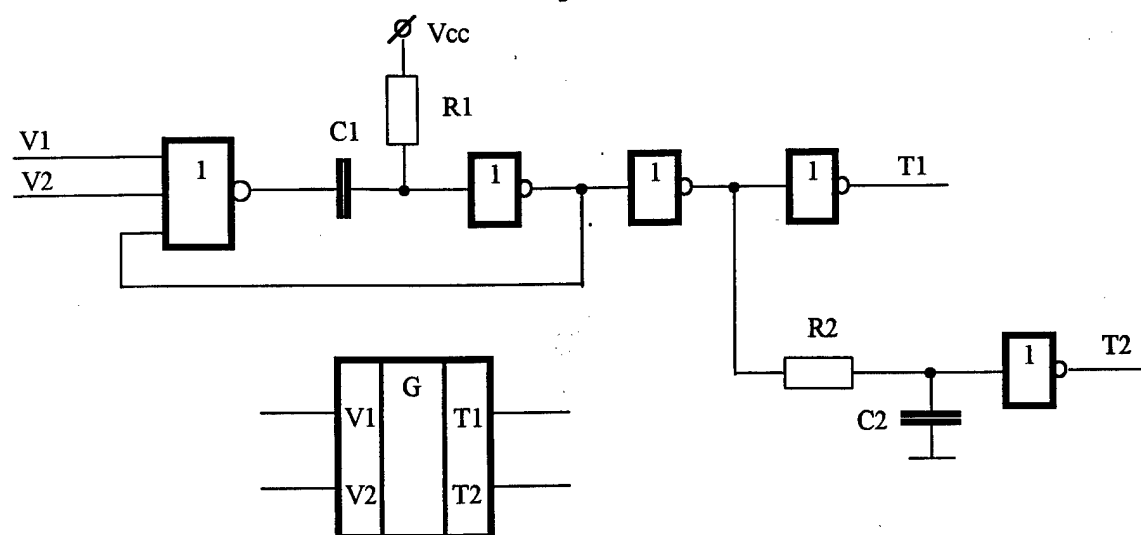
c) Clock generator (retriggerable one-shot)
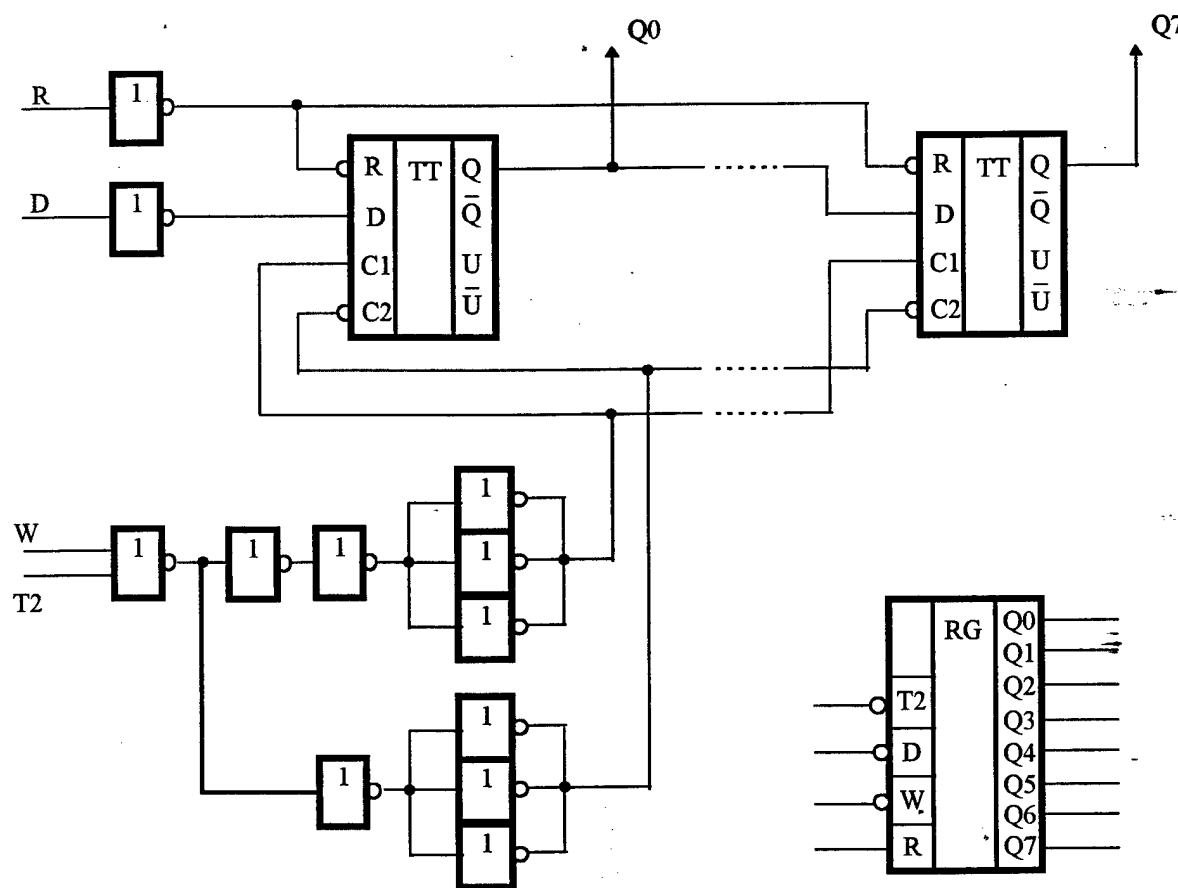
d) Register with reset

Fig. 1.48 **(3 of 4). Synchronous self-checkable converter implementation S_SR8-SC**

e) Register with preset

Fig. 1.48 **(4 of 4). Synchronous self-checkable converter implementation S_SR8-SC**
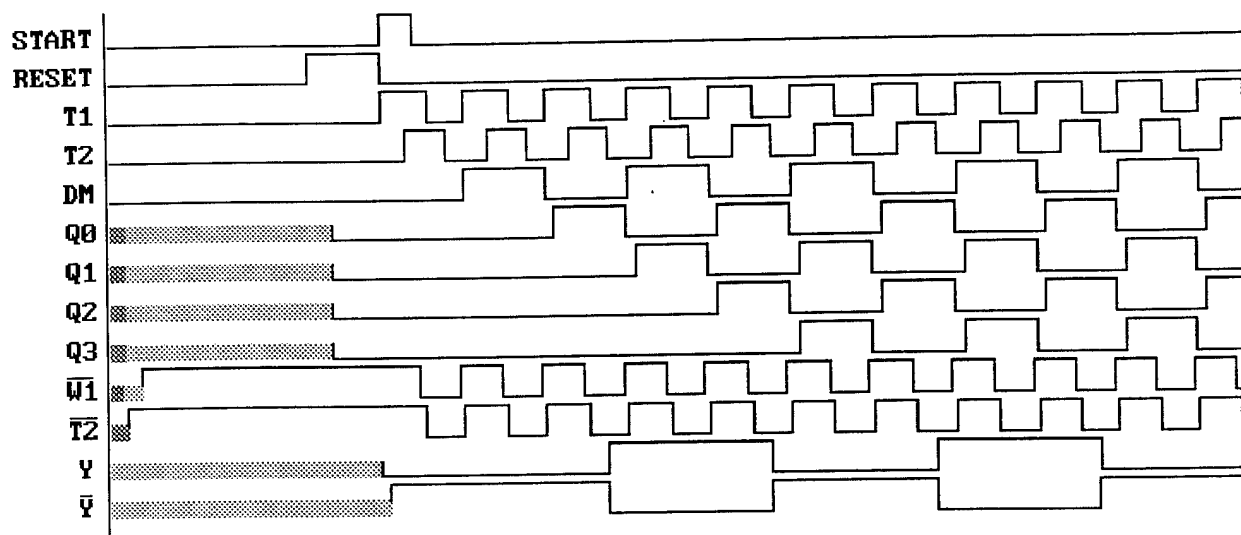
Fig. 1.49. Operation timing diagram of S_SR8-SC

If no faults are revealed in the interface, converter, and comparator, the latter outputs values 01 or 10. In turn, 0 is on the gate pin of the transistor VT6 and 1 — of VT5. All this enables the switch VT5, VT6 to let pass the delayed pulse T2 that triggers the one-shot for a next cycle.

Single faults in S_SR8-SC hardware appear on the comparator outputs as pairs 00 or 11. Accordingly, the values on the gate pins of VT5 and VT6 disable the switch to let pass the pulse T2, the generator idles, and the device seizes to operate. Any faults in the generator followed by absence of a next T2 pulse result in the same.

Electrical simulation of the device check circuit showed significant signal switching delay times (about 100 ns). However, simplicity of its implementation makes the circuit preferable.

1.3.3   The self-synchronous converters

The self-synchronous serial-to-parallel converters are divided below in two groups. The first group includes five modifications built with self-synchronous shift registers (S^3_SRn-0, -1, -2, -3, and -4). Each of converters of this group consists of n double-clock flip-flops (n is the number of bits in the converter) and an indicator of transition process completion. This group is close to conventional synchronous circuits, with a conventional double-clock synchronous flip-flop being replaced by its self-synchronous analog. Functionally, this group is identical to the synchronous self-checkable shift register implementation S^3_SR8-SC.

The second group includes four modifications and is represented by one-bit pipeline registers (FIFOs) with the depth of n. The converters S^3_FIFO8-1 and -2 use the *dense pipeline registers*, in which one useful functional bit is corresponded by two bits of the half-dense register. The converters S^3_FIFO8-3 and -4 use the *dense pipeline registers*, in which the number of functional bits coincides with the number of bits of the dense registers. The half-dense registers have higher speed but require more hardware.

Each of these groups has merits and drawbacks and, consequently, its application areas that will be discussed below. Synchronous implementations based on the pipeline registers were not simulated within this investigation although it would be useful for more comprehensive and correct estimations. However, the authors considered as advisable, with respect to the report objectives (specifically, the self-synchronous schemotechnique) and volume, to involve other synchronous solutions. Nevertheless, a preliminary evaluation showed their speed to differ very little from the synchronous shift registers. The S^3 implementations based on the pipeline registers are included into the report with the purpose to demonstrate both the multitude of schemotechnique methods and tricks, and the variety of library base elements just in the area where application of the self-synchronization is most efficient, namely for coordination of interactions of devices with different speeds.

1.3.3.1   The self-synchronous shift registers

At development of the S^3 shift registers with a serial input and parallel outputs (S^3_SRn-i), decisions on the following choices were to be made:

(1) of a flip-flop cell;

(2) of an interaction discipline between the S^3_SRn register and a transmitter of the serial code (the Master);

(3) of an interaction discipline between the S^3_SRn register and a receiver of the parallel code (a secondary Acceptor);

(4) of a way to built the indicator;

(5) of a way to reset an individual flip-flop and the converter as a whole.

In a number of solutions of the flip-flop cell, sufficient is to enhance its synchronous analog with an additional signal indicating completion of transition processes within the cell. In this case, the synchronous double-clock flip-flop (see *fig. L43*) is built with bidirectional switches, and its proper operation depends on limitations on switching of the inverter D5.

This approach is not compatible with the concept of strictly self-synchronization where any element delay is assumed of an arbitrary but obligatory finite value. Just here lies "the divide" between strictly self-synchronous and quasi-self-synchronous circuits (the latter are a subclass of the former). At the quasi-self-synchronous approach, delays of at least some elements are assumed not to exceed definite values (e.g. signal propagation delays along alternative routs). A number of special technological and topological tricks are applied to meet this assumption. Specifically, some elements of the Event Logic Library for the ARM processor [1.7] are based on this assumption. Refusal from the quasi-self-synchronization concept enables self-synchronous circuits to realize most effectively their potential, the parametric fault-tolerance in particular. Of course, hardware expenses of the S^3 implementation are higher than of analogous quasi-self-synchronous ones. For example, in the given case, the synchronous flip-flop needs 18 transistors (*fig. 1.43*) while quasi-self-synchronous — 24 transistors (*fig. 1.12*).

Solution variants concerning other issues will be considered on implementation examples for the shift registers S^3_SRn-i.

1.3.3.1.1 The shift register S^3_SR8-0

The circuit of the shift register S^3_SR8-0 (see *fig. 1.50*) is an example of a formal approach to the S^3 circuit design. Interaction of the S^3 registers with the Master on the principle "request-reply" assumes an acknowledgment signal back to the Master (like a Slave synchronization — SSYN). Therefore, at the formal approach, the S^3 exchange time (T_{e-sss}) between the Master and S^3_SR8 is defined by the sum:

$$T_{e-sss} = 2t_D + 2t_{SSYN} + t_o + t_I = 4t_{bus} + t_{o-sss} + t_I, \qquad (1.5)$$

where t_D and t_{SSYN} are propagation delays along the Bus (one receiver and one transmitter are included) of assertion and negation of one-bit data and the signal SSYN, respectively, and

t_{o-sss} — receiving and storing time for one accepted bit in the register;

t_I — transition process completion signal generation time in the register.

Accordingly, the synchronous exchange time (T_{e-s}) between the Master and the S_SR8 is defined as follows:

$$T_{e-s} = t_{T1} + t_w + t_o = 2t_{bus} + t_{o-s} \qquad (1.6)$$

t_{T1} and t_w are propagation delays along the Bus of the clock T1 from the generator to the Master and of the Write signal from the Master to the converter (Bus receiver and transmitter included), respectively.

The formulae (1.5) and (1.6) show that, at other equal conditions, the synchronous exchange speed will be higher than the "formal" S^3 exchange speed (although t_{o-s}, as a rule, is greater than t_{o-sss}).

The equation (1.5) is true for the shift register S^3_SR8-0, which circuit is given in *fig. 1.50* and the signal graph — in *fig. 1.51*.

The register operates as follows. In an initial state after reset (see signal values in parentheses in *fig. 1.42*) all register bits are reset ($Q_i = 0$), and the register enables the Master to transfer information ($SSYN = 0$), with the Master outputs having been in the spacer ($D_m = D\tilde{\ }_m = 0$). When the Master initiates the transfer, their outputs (D_m, $D\tilde{\ }_m$) enter the work phase, with the codes 01 or 10. The register forms the signal +C of writing into the 1st stage of the two-clock flip-flops TT2 ($U_i = Q_{i-1}$).

After transition processes completion in the register, the signal +SSYN informs the Master that a transferred information bit is accepted. This makes the Master to transit to the spacer ($D_m = D\tilde{\ }_m = 0$) while the signal C returns to its initial state enabling information to move from the 1st stage of the flip-flops to the 2nd ($Q_i = U_i$). Transition processes completion is indicated by the signal –SSYN marking the moment of parallel code readiness on the converter outputs.

The signal graph in *fig. 1.51* confirms the formula (1.5) and illustrates graphically that such an organization of the converter is based on a **strict sequence of all processes**. Parallelism takes place only within *n* bits of the register. Especially stressed within an exchange act should be the Bus signals designated in the signal graph by double lines: propagation along the Bus of assertion ($D_b \neq D_b\tilde{\ }$) and negation ($D_b = D_b\tilde{\ }$) of one-bit paraphase data code as well as assertion (+SSYN) and negation (–SSYN) of the acknowledgment signal.

At a synchronous exchange, a delay (clock periods) is used instead of an acknowledgment signal, concurrently with initiation of a Bus exchange (two sequential sections in *fig. 1.52*, T1 and –D/–W in the signal graph). After the delay, a parallel process takes place: completion of writing into the converter and return to their initial state of the signals initiating the exchange.

Master   Bus
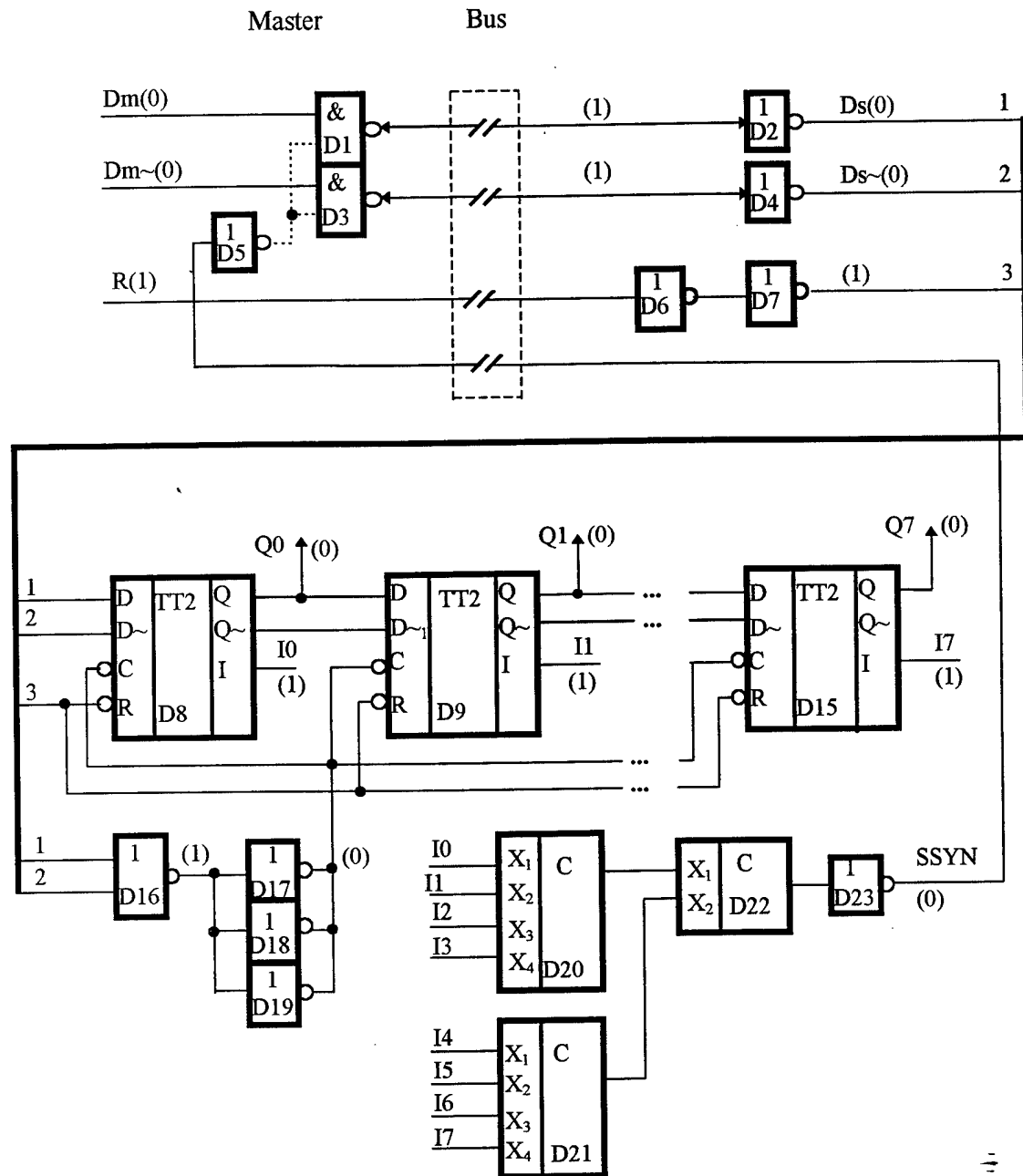
Fig. 1.50. **Strictly self-synchronous shifter S^3_SR8-0**
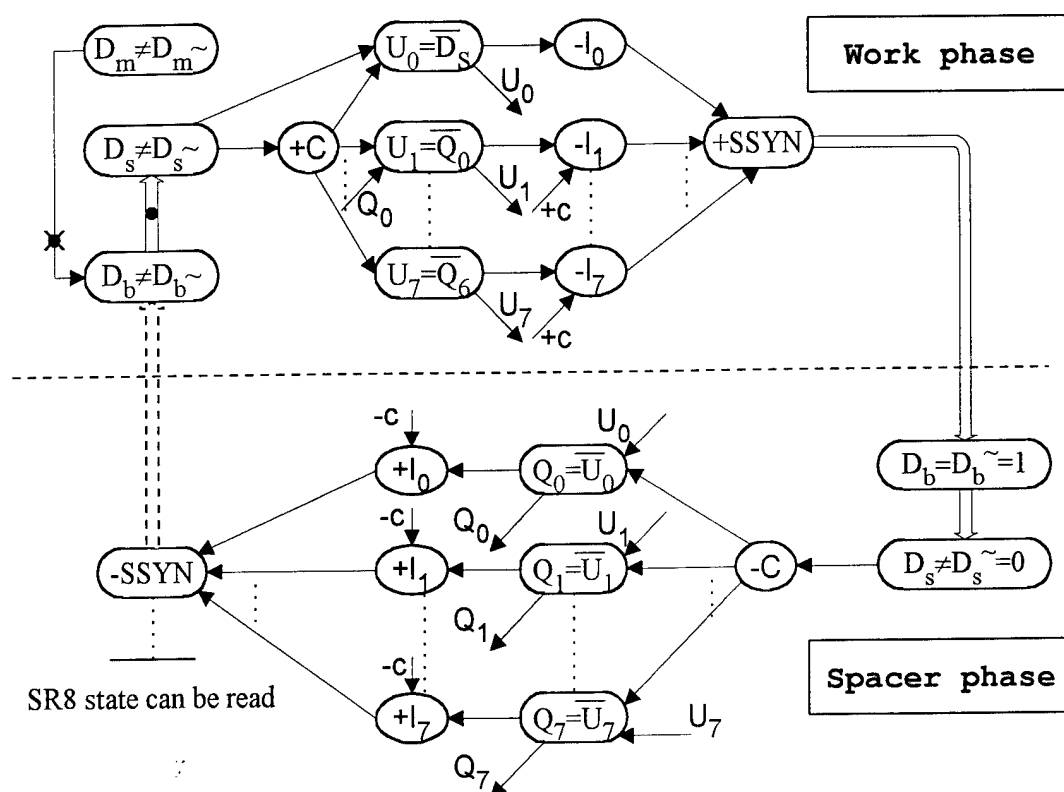
Fig. 1.51. Signal graph of S^3_SR8-0 operation (formal approach)
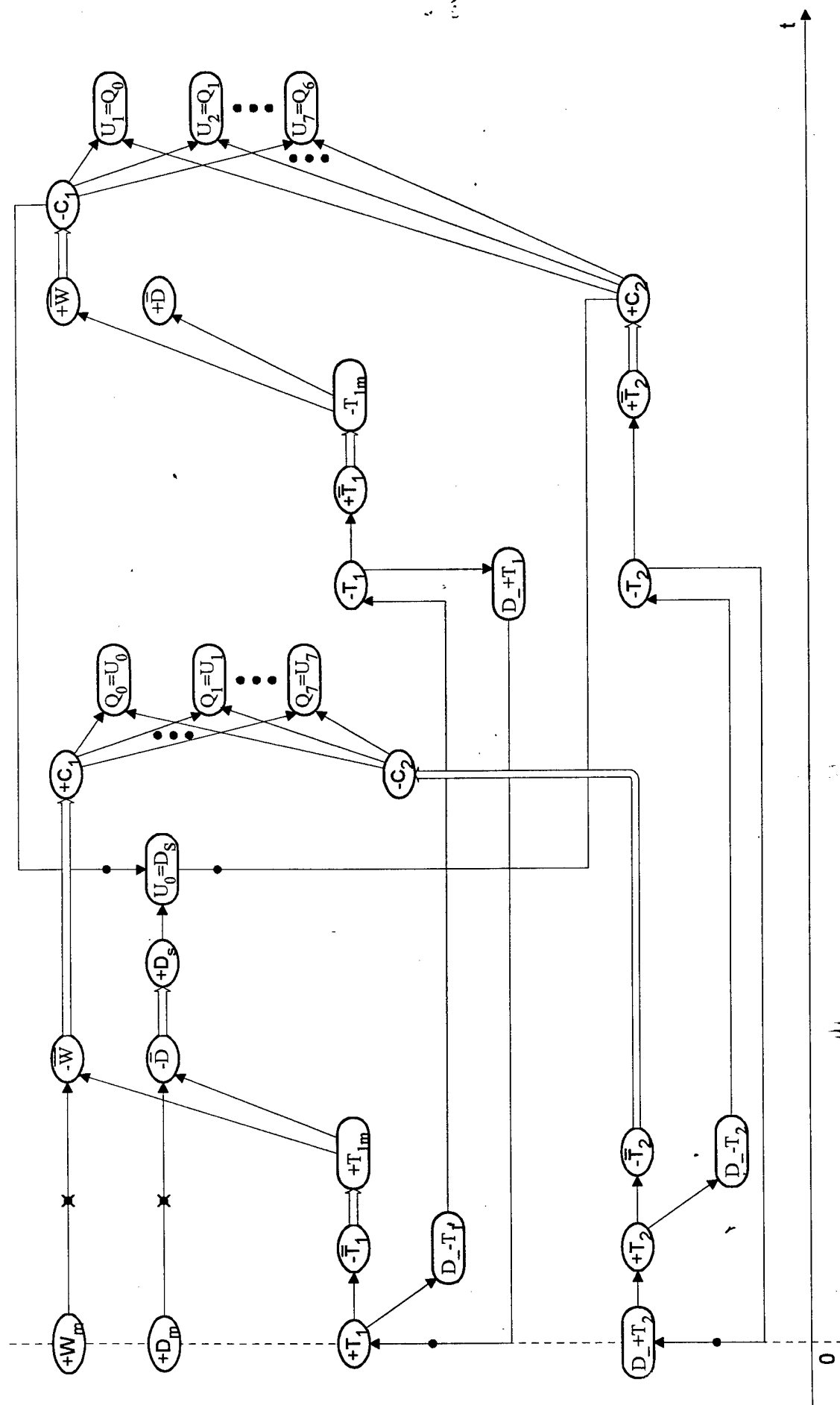
Fig. 1.52. Signal graph and Timing diagram for the S^3_SR8 implementation

If not to take into account this fact, then, at other equal conditions, the speed of the synchronous converters will be higher than of self-synchronous. Even operating of the self-synchronous converter on real delays may happen not to compensate this speed difference with the synchronous converter designed on maximum element delays. *Fig. 1.53* shows dependencies of actual typical write time of an information bit on the total number of Bus loads, obtained from model tests of the synchronous (S), asynchronous (A), and strictly self-synchronous (S^3) converters at normal ambient conditions (T = + 27° C, V_{CC} = 5 V). In particular, the speed of the synchronous implementation S^3_SR8 is 25 % higher versus the self-synchronous implementation S^3_SR8-0.

T_i ns

Operating conditions: + 27° C, 5 V
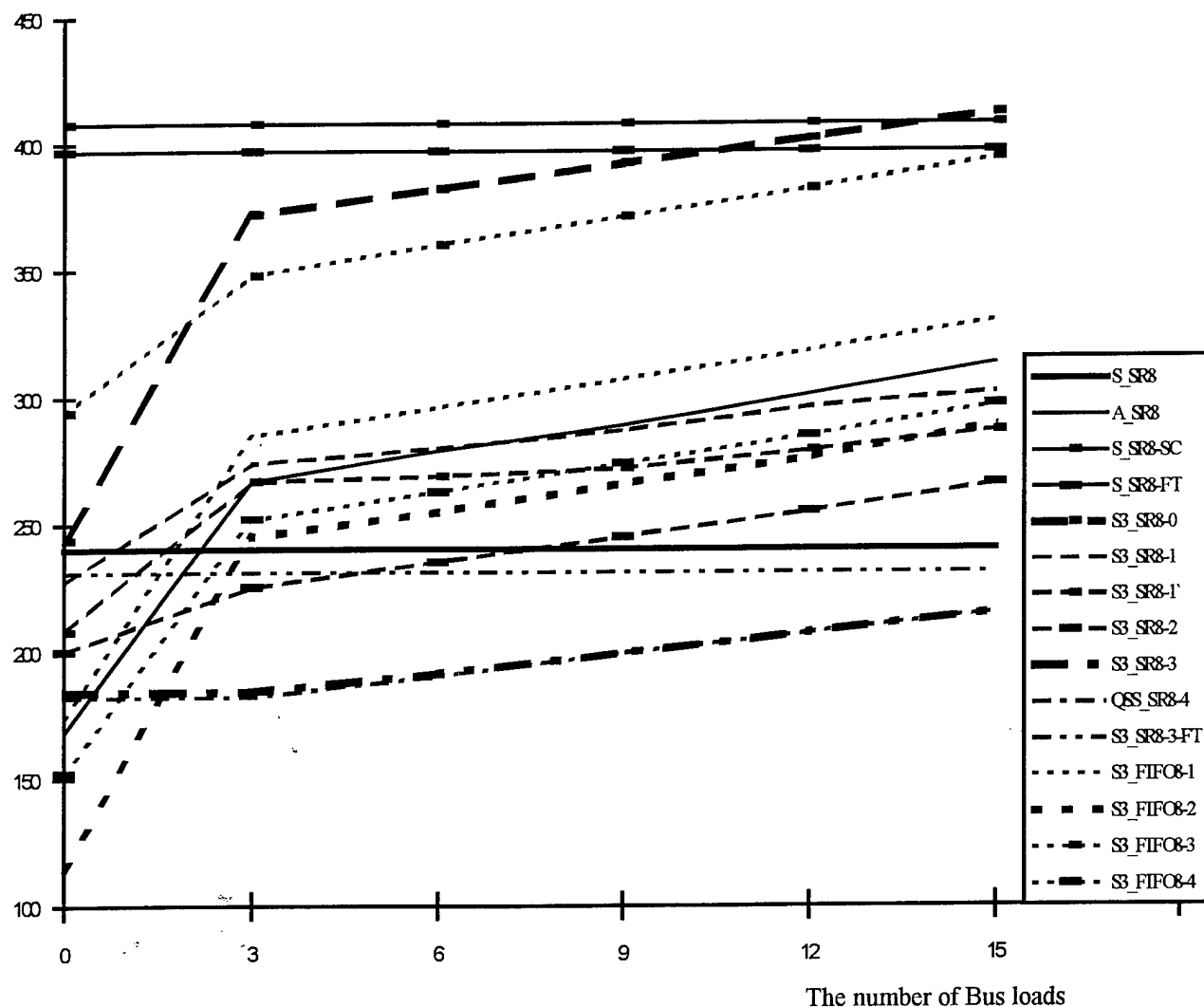
Fig. 1.53. **Actual write time for one register bit**

The two circuits having been compared are not equivalent: the synchronous circuit has no check means while the self-synchronous circuit is self-checkable, the synchronous circuit is "ideal", with an optimal frequency dedicated, clock drivers not included, and technological deviations of gate time parameters not considered. And nevertheless, even at these disadvantageous conditions, correct application of S^3 circuits is able to attain higher speeds as compared with synchronous analogs. The following section explains this paradoxical statement.

The self-synchronous converter features partial concurrence of signal exchange on the Bus and generation of the transition process completion signal. The converter acknowledgment signal is asserted and negated after completion of only those transition processes that relate to information transfer from the Master to the Acceptor rather than of all transition processes in the converter. This means that sufficient is to determine transition process completion in the input flip-flop of the converter (for example, after information is written in its 1st stage).

1.3.3.1.2 The shift register S^3_SR8-1

In general, it is possible to state that the peculiar character of a self-synchronous exchange between interacting units requires a specific approach to hardware design. An external and an internal cycles have to be separated in each general exchange cycle, with the self-synchronous character of interactions being provided within both cycles and between them. The external cycle relates to all transmitting hardware of the Master and a portion of receiving hardware of the Acceptor responsible for reliable exchanging. The internal cycle relates to all remaining hardware of the Acceptor. Every next initialization of an external cycle in the Acceptor has to be synchronized with completion of a previous internal cycle. This approach permits increase of speed of self-synchronous exchanges between interacting units.

Specifically, the increase of speed of the input flip-flop is derivative from peculiarity of its input signal — paraphase with the spacer. Therefore, the input (0th) bit in the $\tilde{S}^3$_SR8-1 implementation differs from other bits. Its 1st stage is implemented, see *fig. 1.54*, by the elements D3, D7 while the 2nd stage — by the one-clock cell T1 (D10).

The register functions as follows. The Master initiates the transfer process by the work phase (codes '01' or '10') of its outputs $D_m, D^{\sim}_m$ that forms the signal $+C$ in the register and writing into the 1st stage of the input flip-flop ($U0 = D_s$). Meeting these conditions is necessary and sufficient to finish an exchange cycle and render the acknowledgment signal ($+SSYN$) to the Master. From this point, two processes start to develop concurrently (see *fig. 1.55*):
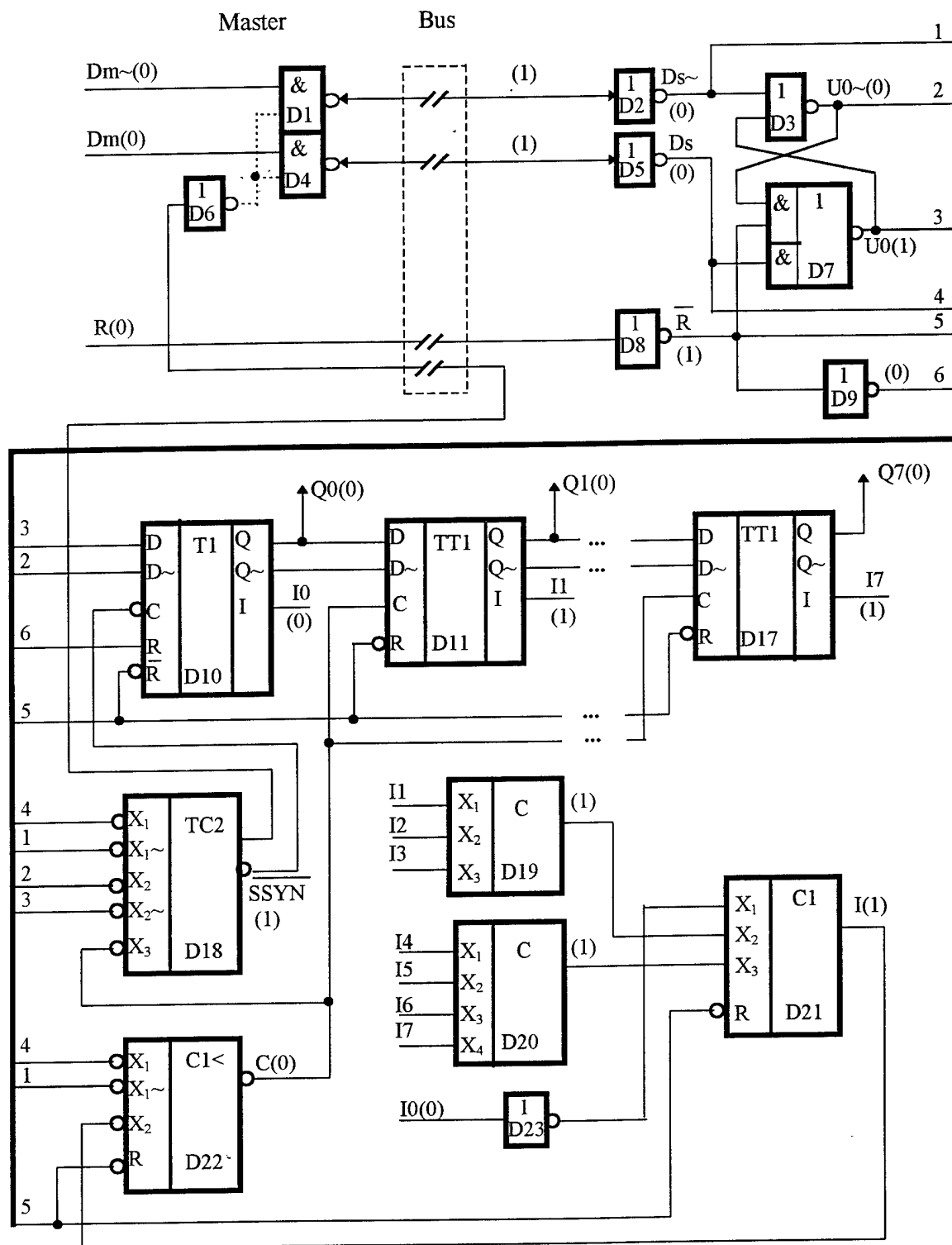
Fig. 1.54. Strictly self-synchronous shifter S^3_SR8-1
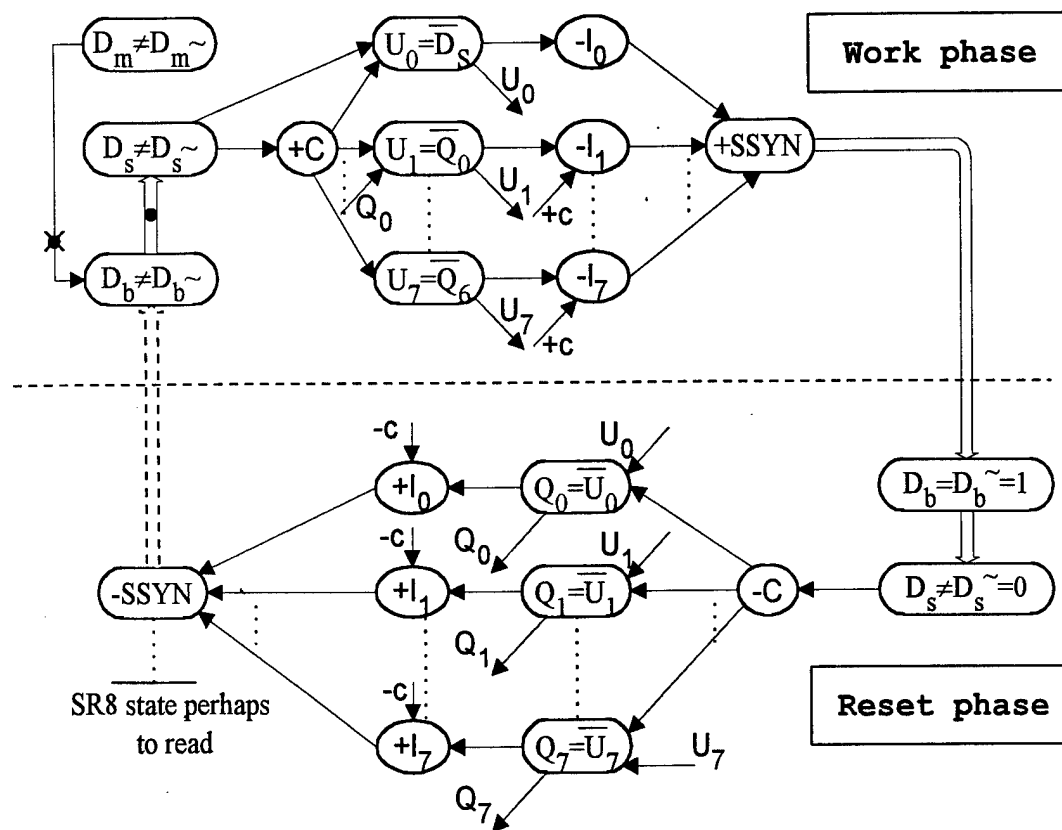
***Fig. 1.55.* Signal graph for S^3_SR8-1**

(1) generation of the signal +SSYN to the Master, initiation of the reset phase ($D_b = D_b\sim = 1$), and detection of this fact by the Acceptor ($D_s = D_s\sim = 0$);

(2) moving of information from the 1st stage of flip-flops to the 2nd and detection of the fact of completion of all transition processes in the work phase, indicated by the signal –I and meaning readiness of a parallel code on the converter outputs.

The circuit returns to an initial state (the reset phase) analogously.

The following formula allows evaluation of the period $T_{e\text{-}sss}$ for the S^3_SR8-1:

$$T_{e\text{-}sss} = \max (4t_{bus}, t_I) + t_0, \quad (1.7)$$

Thus, the interaction period is determined by only one value being larger, $4\,t_{bus}$ or t_I. In the case of a single load on the Bus, with standard push-pull drivers instead of tri-state buffers, the implementation S^3_SR8-1 has higher speed than the S^3_SR8-0 but lower than the S^3_SR8-2 (refer to *fig. 1.53* and *table 2* of Appendix **1.2**). Even a simple replacement of usual inverters by inverting bus receivers (D2, D5, and D6) and tri-stated inverting bus transmitter (the output cascade D18) equalizes speeds of the synchronous and self-synchronous converters, and the

self-synchronous implementation becomes faster at larger numbers of Bus loads ($n \geq 2$). The following factors enhance the speed of the $S^3\_SR8\text{-}1$ implementation:

— time overheads to signal transition processes completion are entirely compensated by the faster Bus exchange due to concurrent process organization;

— operation on real delays (at normal ambient conditions) enables the $S^3\_SR8\text{-}1$ implementation to gain more speed than to lose because of a slower Bus signals exchange: 4 t_{bus} for $S^3\_SR8\text{-}1$ and 2 t_{bus} for $S^3\_SR8\text{-}2$.

Advantages of the implementation $S^3\_SR8\text{-}1$ are more salient at the lower limit ambient conditions ($T = -63°$ C, $V_{CC} = 7$ V), and the self-checkable implementation $S^3\_SR8\text{-}1$ becomes slower than the not self-checkable implementation S^3_SR8 only at the upper limit ambient conditions ($T = -125°$ C, $V_{CC} = 3$ V).

One more aspect of the term "actual speed of the S^3 circuits" should be noted. Duration of transition processes in them depends on not only real element delays but also kind of information being subject to processing. For example, duration of transition processes in the self-synchronous converter will be almost twice shorter if zero symbols or zero bits in a symbol are received just after the Reset signal. Naturally, a next zero bit on the converter input does not force any changes in the converter state when all its bits have been reset to zero. Duration of transition processes is determined only by duration of switching of indicator elements.

Of course, the example above is uncommon. An example of a more probable event is given below. Every time when a next received bit is equal to preceding one, transition processes in the converter complete faster because the input converter bit does not change its state. The high probability of such combinations on the input was taken into account in the chosen test input sequences: 00-11-00-11.

Knowledge on statistical properties of incoming data sequences, e.g. on inputs of operational devices and others, can be regarded as a means to enhance performance of S^3 circuits. Hence, S^3 circuits can be adjusted to incoming information that is impossible for conventional synchronous circuits.

Let us consider some peculiarities of resetting the self-synchronous converter in an initial state after it is powered on. This concerns primarily state determination of the Muller's C-elements when the state of their internal nodes (Y in *fig. 1.56*) is not determined unambiguously by the state of their inputs, for example, if the initial state of the inputs is $X_1 = 0$, $X_2 = X_3 = 1$ and of the output — $I = 0$. By this, one of three following resetting variants can be involved.
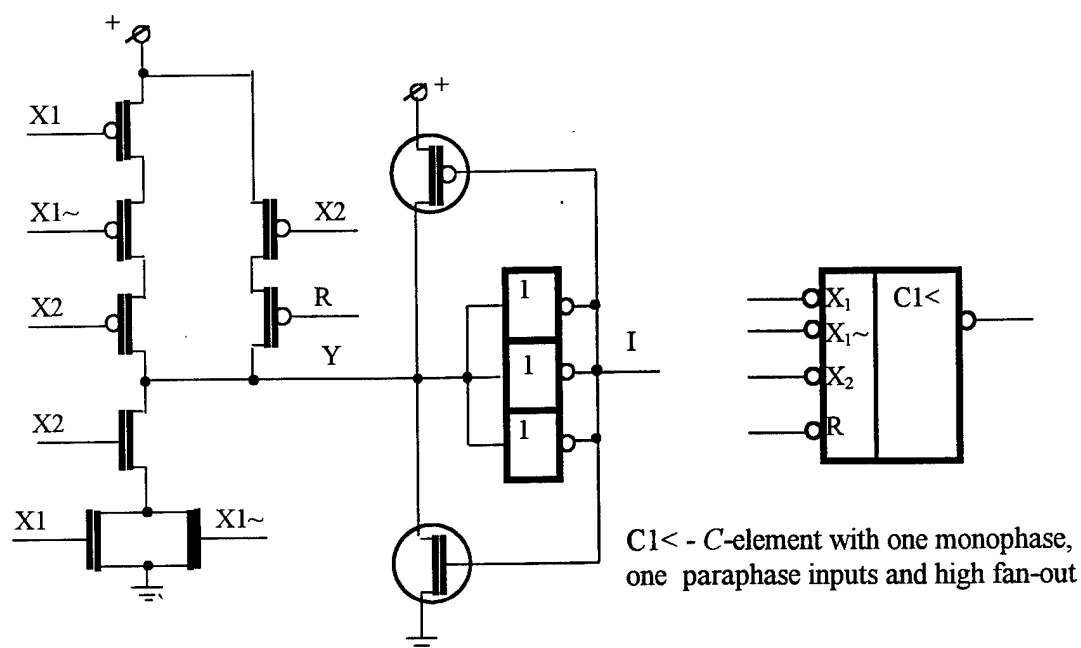
Fig. 1.56. **Pseudo-static Muller's *C*-element**

a) Conventional, *fig. 1.57a*, eliminating currents-through. The variant is used, for example, in [1.6]. Its drawbacks are: the longer switching of the Muller's *C*-element at $X_1 = X_2 = X_3 = 1$ and the smaller maximum number of inputs (e.g. 3 at 4 serially connected transistors permitted).

b) Nonconventional, *fig. 1.57b*, allowing short-term currents-through at resetting. This variant is acceptable if the reset signal R is used rarely, specifically only at powering on.

c) Combined, *fig. 1.57c*, free of drawbacks of variants (a) and (b). The reset signal R starts acting only after the input signal X_1 is set to its initial state. The variant is good for operative resetting of circuits.

Presetting of the Muller's *C*-element can be also implemented on the variant (c). The implementation S^3_SR8-1 is the sole S^3 converter circuit applying the variant (c). All others apply the variant (b). Some specificity of resetting the circuit S^3_SR8-1 is illustrated by the signal graph in *fig. 1.58* that shows the reset procedure to have not self-synchronous nature.

The effective reset can be applied to flip-flops, both one-clock and two-clock. For example, resetting the flip-flop D3, D7 in *fig. 1.54* is made by the signal R introduced in the flip-flop feedback circuitry. The additional serial transistors do not practically increase its switching time at writing both 0 and 1 as compared with a flip-flop without reset.

a) Noncomplementary reset

b) Slow complementary reset

c) Fast complementary reset
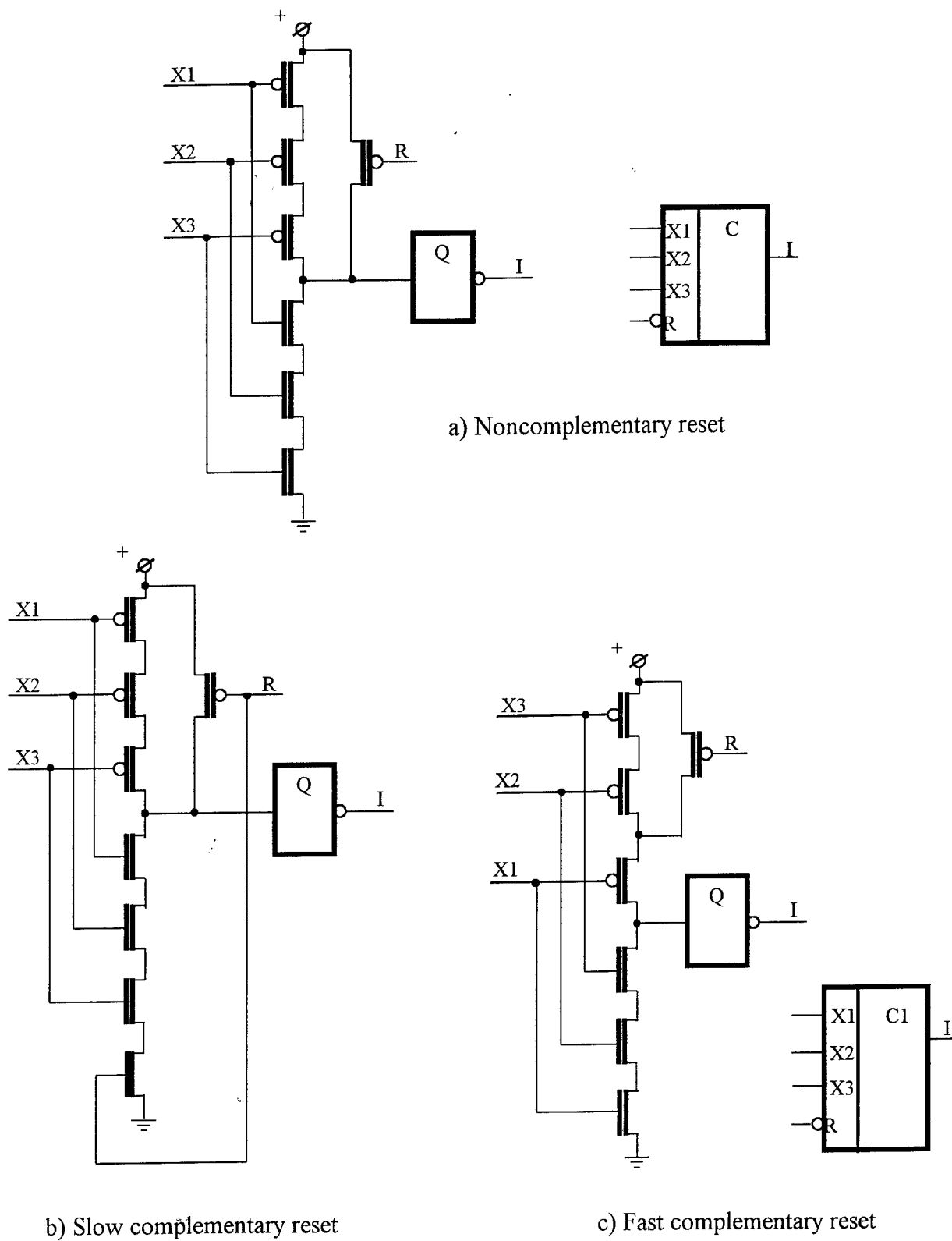
Fig. 1.57. **Variants of resetting 3-input *C*-elements**

Start

+R

$U_0=1$

$D_m=D_m{}^\sim=0$

-R

$Q_0=0$

$U_0{}^\sim=0$

$-I_0$

-I

During reset

-C

$\overline{R}$ $\quad$ $\overline{R}$ $\quad$ • • • $\quad$ $\overline{R}$

$Q_1=0$ $\quad$ $Q_2=0$ • • • • $Q_2=0$

-c $\quad$ -c $\quad$ -c

$U_1=\overline{Q_0}$ $\quad$ $U_2=\overline{Q_1}$ • • • • $U_7=\overline{Q_6}$

Delay $\quad$ $+I_1$ $\quad$ $+I_2$ • • • • $+I_7$

-R $\quad$ $+\overline{R}$ $\quad$ $+I_0$ $\quad$ +I
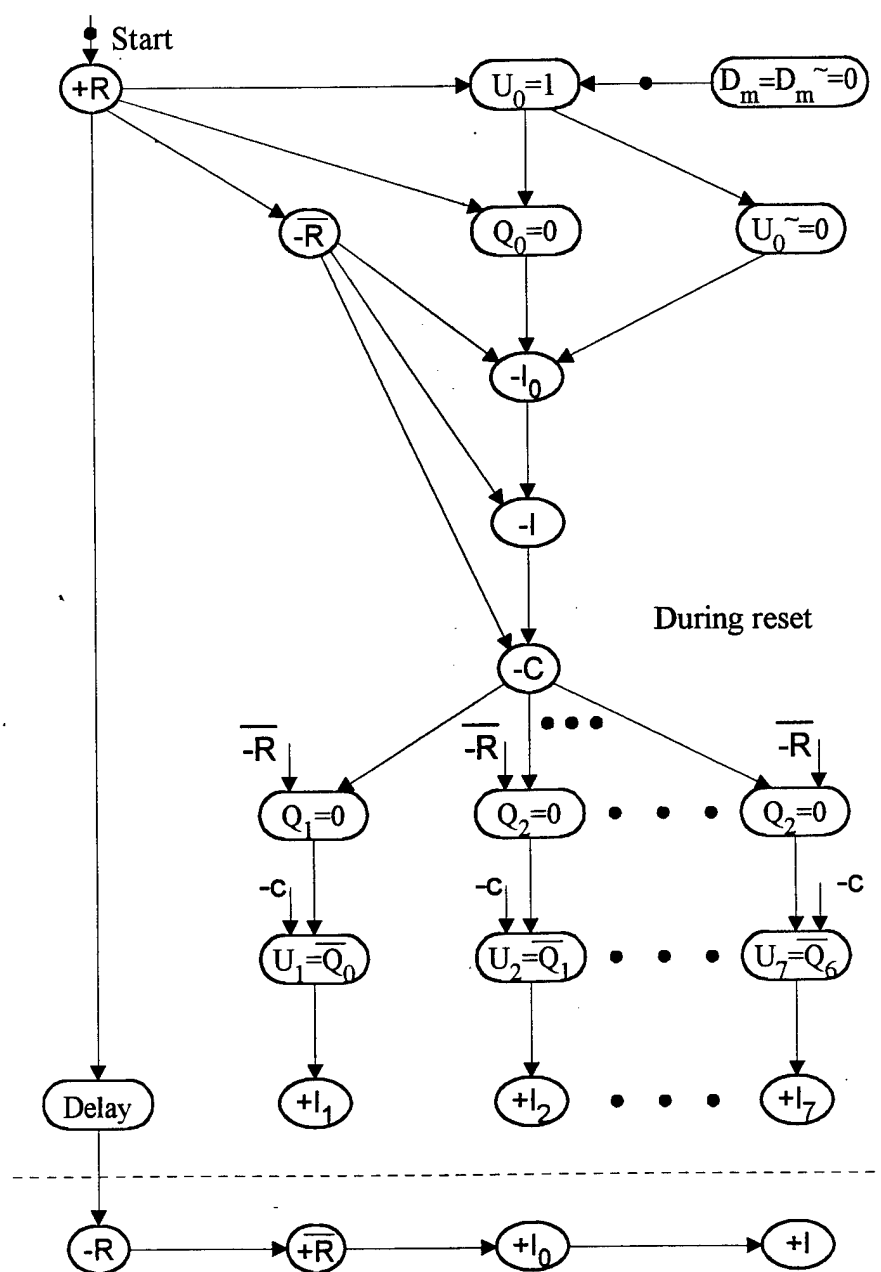
Fig. 1.58. **Signal graph of resetting S^3_SR8-1**

Speed of the circuit S^3_SR8-1 grows if the two-cascade element D18 in *fig. 1.54a* is replaced by its four-cascade modification S^3_SR8-1' given in *table 2* of Appendix 1.2. This replacement is just used in the converters considered below.

1.3.3.1.3 The converters S³_SR8-2 and S³_SR8-3

There are three more techniques of speed enhancement for S^3 converters based on serial shift registers. 1st technique uses a "prefetch" mechanism. At resetting, the converters under consideration transit to a required initial state and disables writing in them new information. Negation of a reset signal initiates the prefetch process of outstripping rewrite of information from Q_i to Q_{i+1}. After this process finishes, the device is enabled and ready to accept new information. As a rule, any accesses of potential Masters to the converter after the reset occur rather later than the prefetch process in the converter completes. Therefore, a parallel code on $Q1 \div Q7$ is already established, as a rule, at the moment of a 1st and every next Bus exchange.

The converter S^3_SR8-2 operates as follows (see *fig. 1.59* and the graph in *fig. 1.60*).

As soon as a next information bit enters the converter ($D_s \neq D_s\sim$) it is being written into the 1st stage of the input bit (D8, D9). Completion of the write generates the signal I_u0. Then completion of transition processes of a previous exchange ($C = 0$) initiates two concurrent processes: completion of the current Bus exchange indicated by the signal +SSYN and completion of transition process in the converter.

The considered feature of the converter S^3_SR8-2 enabled it to be sped up by 12 % comparing with S^3_SR8-1' (at the number of Bus loads $n = 6$).

In the converter S^3_SR8-2, transition processes completion in the input flip-flop is indicated by the element D10 (the output signal I_u0), and the Muller's C-element (D11) combines it with the signal C. Both functions can be implemented by one *TC*-element (the element D9 of the converter S^3_SR8-3 in *fig. 1.61*).

The last technique of speed enhancement is connected with a possibility to apply in a converter of a one-clock input flip-flop rather than of two-clock one (the 2nd stage) in supposition that the 1st stage is in the Master. The self-synchronous way of information transfer from the Master (1st stage) to the converter (2nd stage) enables this supposition to realize. This reduces duration of internal conversion cycle.

Application of the *TC*-element and simplification of the input flip-flop in the converter permitted the implementation S^3_SR8-3 to achieve the speed 14 % higher in comparison with the implementation S^3_SR8-2. Finally, the speed of the self-checkable converter S^3_SR8-3 is better than of the not self-checkable synchronous converter: the actual write time for one register bit is 191 and 240 ns, respectively.

Master   Bus
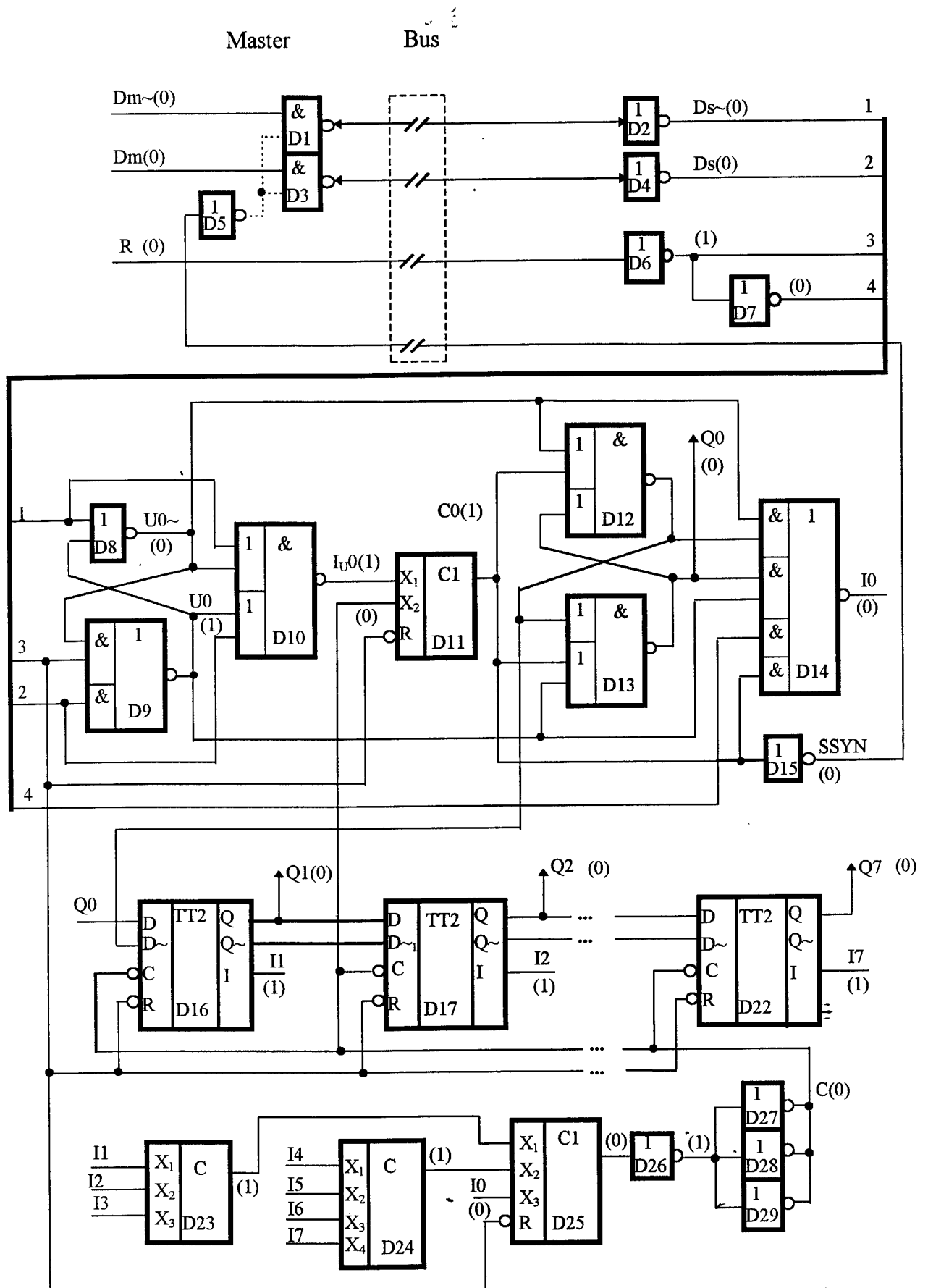
Fig. 1.59. **Strictly self-synchronous Shifter S³_SR8-2**

Work phase   Reset phase
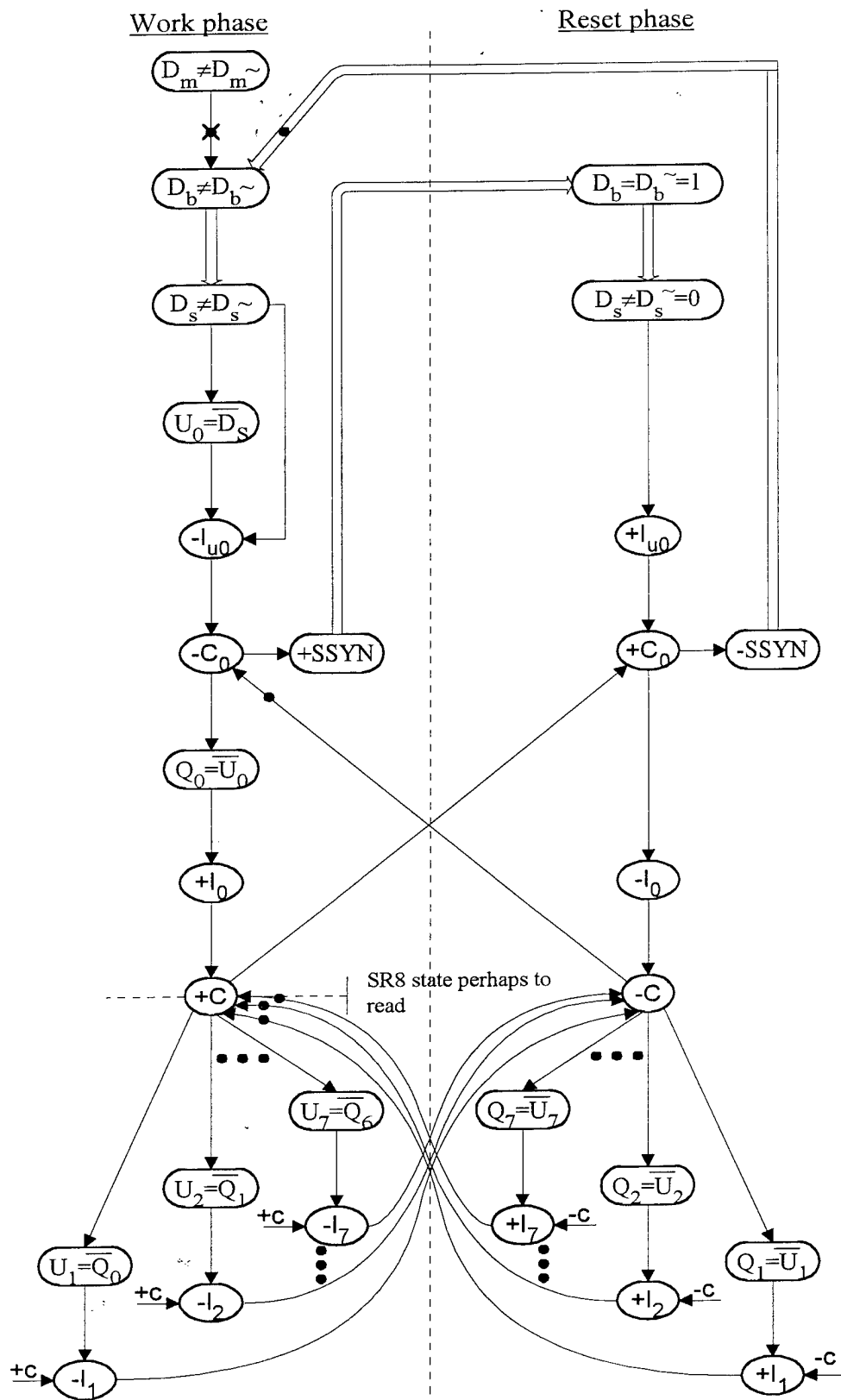
Fig. 1.60. Signal graph for the S^3_SR8-2
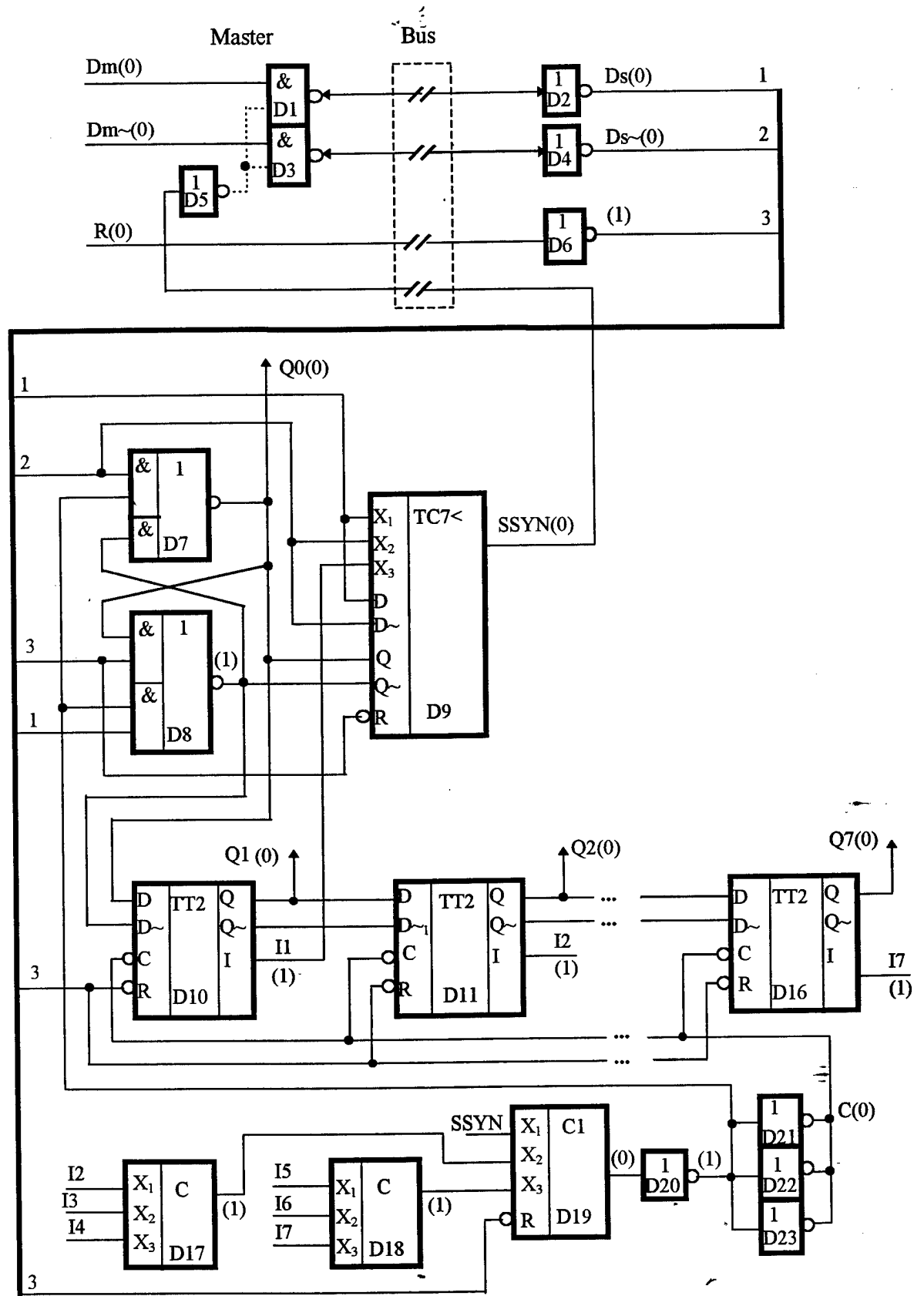
Fig. 1.61. Strictly self-synchronous Shifter S_SR8-3

1.3.3.1.4 Using of a multi-input H-flip-flop

In real designs with regular units (multi-bit registers, counters, ALUs, etc.) there arises a problem of efficient indication of transition processes which occur, as a rule, in all bits. Implementation of a common indicator on Muller's C-elements described above results in excessive hardware overheads and slows down device operation. In a number of cases, the problem can be solved using a multi-input H-flip-flop.

Given in [1.8] is the circuit of a H-flip-flop, in which the number of inputs is not limited, *fig. 1.62*. It consists of one input and one output cascades connected by flip-flop links. The output cascade represents a usual CMOS inverter (the element D2). The input cascade contains two CMOS inverters (D1 and D3).
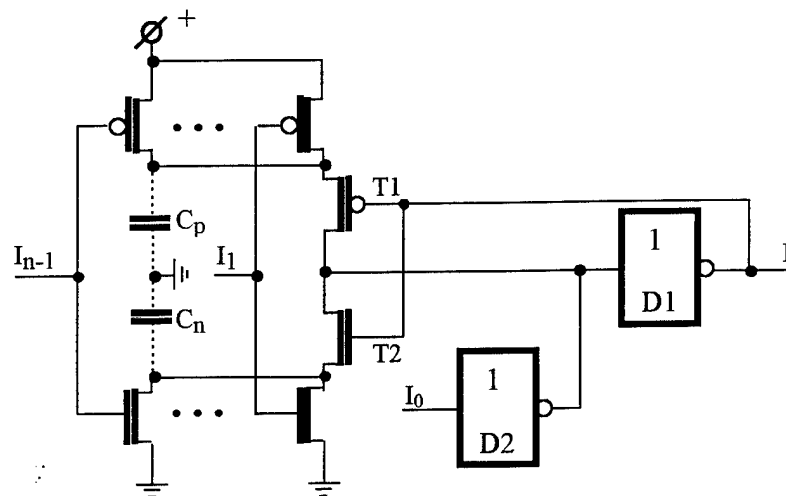
Fig. 1.62. **Multi-input H-flip-flop**

The former is connected to the power supply via p-transistors and to the "ground" — via n-transistors, with their gate pins tied together, that form an (n - 1) input pairs. The input pairs receive signals from indicator outputs $I_1, \ldots, I_{n-1}$ of register bits. The input of the inverter D1 is connected to the output I of the H-flip-flop. The input of the inverter D3 is connected to the indicator output I_0 of the input (0th) register bit.

The multi-input H-flip-flop functions as follows. Let the H-flip-flop be in the initial state: $I_i = 0$, $i = 0, \ldots, n - 1$. Then the n-transistors of the input pairs and inverters D1 and D3 are closed while their p-transistors — opened. The high level on the inverter D2 input makes $I = 0$.

Switching of the input signals $I_i = 1$, $i = 0, \ldots, n - 1$ opens the n-transistors and closes the

p-transistors of the input pairs and inverter D3. The parasite capacitor C_p of the sources pin of the *p*-transistors is being discharged through the *p*-transistor of the first input inverter and the *n*-transistor of the inverter D3, and the voltage on the input of the inverter D2 is being lowered. As a result, the inverter D2 starts to switch in the state $I = 1$ opening the *n*-transistor and closing the *p*-transistor of the inverter D1 that accelerates switching of the flip-flop in the state $I = 1$.

Transition of the H-flip-flop in the initial state is symmetrical to the process described above.

The H-flip-flop does not consume power in a steady state. The following condition is necessary for elimination of currents-through at transition of an indicated circuit from one state to another: the signal I_0 must change not earlier than the signals I_1, ... , I_{n-1} have been changed. Otherwise, a current-through appears, via not closed transistors of the input pairs and inverters D1 and D3, that is able, under definite conditions, to provoke a premature switching of the H-flip-flop (a change of its output), that is a circuit malfunction. The malfunctions can be prevented if transistor characteristics are chosen properly: the transistors in the inverter D3 must be several times weaker than the transistors in the input pairs and inverter D1. However, such a differentiation makes switching of the H-flip-flop longer since the parasite capacitors are recharged by weak currents via the transistors of the inverter D3.

In a number of practical cases, proper design allows a required delay of the signal I_0 to ensure with respect to the signals I_1, ... , I_{n-1}. In particular, the input bit indicator I_0 switches in the registers S^3_SR8-1, ... , S^3_SR8-3 certainly later of indicators of other bits. In this connection interesting is estimation of efficiency for a multi-input H-flip-flop as a common indicator in the shift registers in comparison with an implementation on conventional *C*-elements.

With the number of bits in shift registers growing, a common indicator becomes more complicated and loses performance in a logarithmic dependency on the number of bits. It is consequent upon practical restrictions on the number of *C*-element inputs. Since the multi-input H-flip-flop is based on different operation principles, it permits a larger number of register bit indicators to be combined. Its performance degrades only because of additional parasite capacitors C_p and C_n, with the degradation rate being essentially less versus *C*-elements.

Fig. 1.63 exposes dependencies of a typical (averaged on both phases) switching time of a common shift register indicator versus the number of bits in the register for the implementations on the multi-input H-flip-flop and conventional *C*-elements. At the delay evaluations taken into account were only the bit indicator switching times relative to register control signals.

Comparison of two curves shows the C-element implementation being better for small numbers (≤ 8) of register bits and giving in for the larger numbers (≥ 16).

It should be noted that the performance estimations in *fig. 1.63* were obtained in supposition that the Master of a register does not introduce additional delays in the shift register operation. Otherwise, the switching delay of the input bit indicator I_0 is excessively large, and a common "pyramidal" indicator on conventional H-flip-flops, with the output I_0 connected to its last cascade, has enough time to form all intermediate signals before I_0 is changed, and, as a result, appears being faster than the multi-input H-flip-flop.
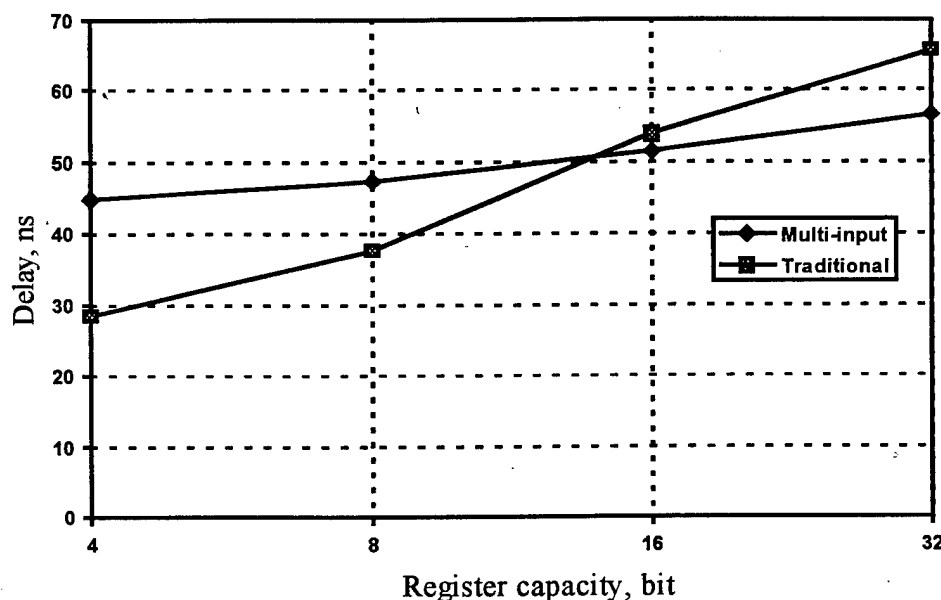
Fig. 1.63. **Typical switchinf time of the indicator in the register S^3-SR8**

Thus, effective using of the multi-input H-flip-flop depends on a specific implementation of the indicated circuit ant timing characteristics of its interface with the environment.

Hardware expenses to implement a common indicator on the multi-input H-flip-flop are significantly less than on conventional C-elements, with the difference growing proportionally to the number of bits in the shift register. Furthermore, general hardware` overheads can be additionally reduced by embedding of bit indicator circuits into the multi-input H-flip-flop as shown in *fig. 1.64* for an 8-bit register.

Some example results of applying the combined multi-input H-flip-flop are given in *table 1-3* of Appendix **1.2** within the transmitter S^3_**SRi-4**.
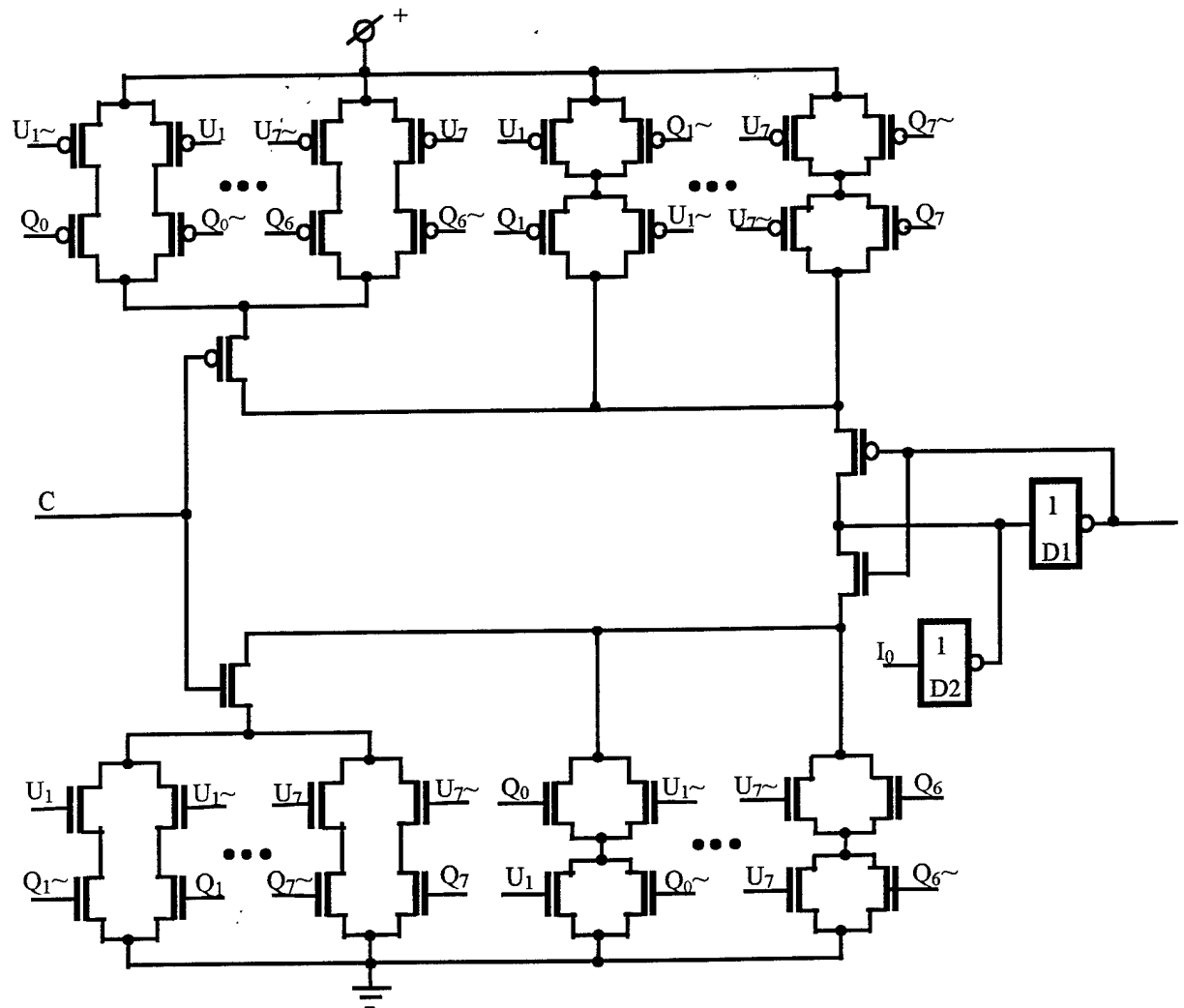
Fig. 1.64. **Combined multi-input H-flip-flop**

Thus, the multi-input H-flip-flop, at a proper S^3 circuit schemotechnique design, allows solution of the problem of efficient indication for regular units with sufficiently large number of bits. This solution is prospective since data width is growing permanently. For example, the interface Futurebus+ allows the computer data bus widths to vary from 32 to 256 bits [1.6]. The multi-input H-flip-flops are most preferable in such systems as saving hardware, especially if register bit indicators are combined (embedded) in the multi-input H-flip-flop as shown in *fig. 1.64* for an example 8-bit shift register.

1.3.3.2 Self-synchronous pipeline registers

Pipeline register is a serial register intended to input, store, and output serial information bits in accordance with the FIFO discipline. Due to a special organization, a pipeline register is able to perform the role of a buffer-accumulator between a source and receiver of bit information. Such a register contains several serially connected cells, with the first cell being connected to a source (input Master) and the last — to a receiver (output Master). An information portion output by the source is moved along all register cells. Depending on speed ratio of the information source and receiver, the pipeline register is filled in at one moment and emptied out at another. Interactions between an input bit of the register with the source and of an output bit with the receiver is implemented on the "request-reply" principle. The source and receiver can operate independently one from another until the register becomes entirely empty or entirely full.

The exchange time of one information bit in the register includes write time in the input (0th) bit — 1st constituent, and switching of a common indicator showing transition processes completion in all register bits — 2nd constituent. The 2nd constituent is determined only by duration of moving information from 0th bit to 1st one. If, with the number of register bits growing, speed of a shift register is degraded because of complication of a common indicator, speed of a pipeline register does not change. Therefore, pipeline registers provide in many cases higher performance in the serial-to-parallel converters in comparison with the shift registers.

Shift registers, as opposed to pipeline registers, are indispensable for continuous inputting sequential codes and outputting parallel codes at arbitrary time. Readiness of a parallel code of any length is indicated by transition to the work state of a common register indicator. Pipeline register do not have such a common indicator that makes difficult indication of readiness of the register to output an accumulated parallel code at arbitrary time.

1.3.3.2.1 The undense pipeline register S^3_FIFO8-1

The undense pipeline register, *fig. 1.65*, has been proposed in [1.2] and consists of 15 memory cells (bits). Its first bit (entrance bit) has an additional element to form a signal of readiness to accept information (this signal is referred here as SSYN). Its last bit (exit bit) differs from an internal cell by absence of feedback inputs from information outputs of a subsequent bit.

Information on register inputs and outputs is paraphase encoded. A parallel code is represented on the register Q_i outputs.

Master   **Bus 1**

Dm~(1)   1 D1
Dm(1)   1 D3
                1 D5

Ds~(1)   1
Ds(1)   2
SSYN(1)

Q0(1)   Q7(1)

DN   T2   DN   T2   "1"   DN   T2
DN~   DN~   DN~
2 D   Q   D   Q   D   Q
1 D~   Q~   D~   Q~   U1(1)   D~   Q~
   IN   I   IN   I   IN   I
   IN~ D6 I~   IN~ D7 I~   IN~ D20 I~

$I_Q0(1)$ $I_U1(1)$ $I_Q7(1)$

&   1 D23
D21   1 D22

Ds2(0)   **Bus 2**
                                     Receiver
1 D24   1 D26   SSYN2
1 D25   (1)
                 Ds2~(0)
         MSYN2(1)
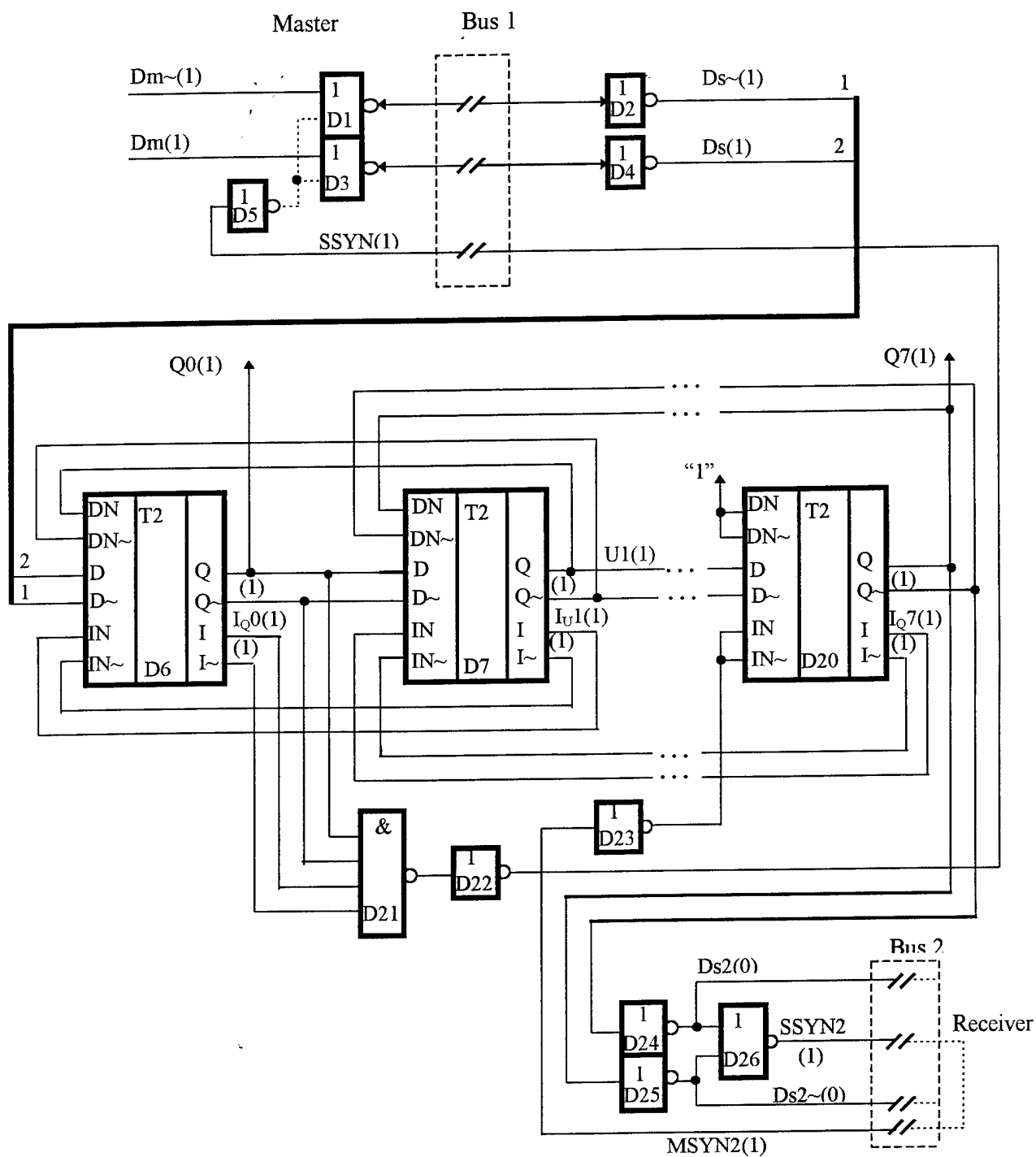
Fig. 1.65. **Undense pipeline register S^3_FIFO8-1**

The schemotechnique implementation and functioning specificity of the cell *T2* are described in the section 1.3. The register can be represented, in a lengthwise section, as two sequential channels, left and right, formed by connection of half-cells of adjacent cells. One channel transfers information ones, another — zeros. The half-cells are connected by cross-links preventing simultaneous writing into both half-cells. This ensures bits within different channels not to outrun one another and to preserve their sequential order. Register operates as follows (refer to the signal graph in *fig. 1.66*). In its initial state (signal values are given in the figure in parentheses) all half-cells must be in the spacer $(Q_i = Q\tilde{}_i = U_i = U\tilde{}_i = I_{Qi} = \Gamma_{Qi} = I_{Ui} = \Gamma_{Ui} = 1)$. This is provided by the initial preset of the register accomplished by multiple assertion of the read-out signal on the input $SSYN_2$ of the exit bit, with the Master forming the spacer $D_s = D\tilde{}_s = 1$ on the information inputs of the register.

Under operation, the Master must set on the information inputs D_s, $D\tilde{}_s$ codes either 01 or 10. These codes force the entrance bit to enter an intermediate state $(Q_0 = 1, I_{Q0} = 0$ or $Q\tilde{}_0 = 1, \Gamma_{Q0} = 0)$ and to issue to the source the signal $SSYN = 0$ disabling the write. The source replies with the spacer state 11 on the register inputs that enables transition of the entrance bit half-cell to the work state $(Q_0 = I_{Q0} = 0$ or $Q\tilde{}_0 = \Gamma_{Q0} = 0)$. Further, if the 2nd and 3rd half-cells of the same channel are in the spacer, the accepted bit is moved to the 2nd cell, and the entrance bit returns to the spacer.

The signal value $SSYN = 1$ means, for the Master, permission to issue a next information code while the accepted bit is moved to the register exit bit independently of the source.
If $MSYN_2 = 1$, the accepted bit moves to next to last bit position of the register since $MSYN_2 = 1$ confines transition of the exit register bit to the work state, and the spacer $Q_7 = Q\tilde{}_7 = 1$ is preserved on the register outputs. To read the result, the read-out request $MSYN_2 = 0$ has to be set.

If the register is full, then $SSYN = 0$ that means the writing is disabled, and the entrance register bit does not react to changes of information signals. If the register is empty, then the register exit bit outputs $Q_7 = Q\tilde{}_7 = 1$. They do not change at 0 on the input $MSYN_2$, and the register exit bit does not react to the read-out request. The hardware overheads and speed characteristics for the register $S^3\_FIFO8-1$ are given in *table 1-3* of Appendix 1.2.
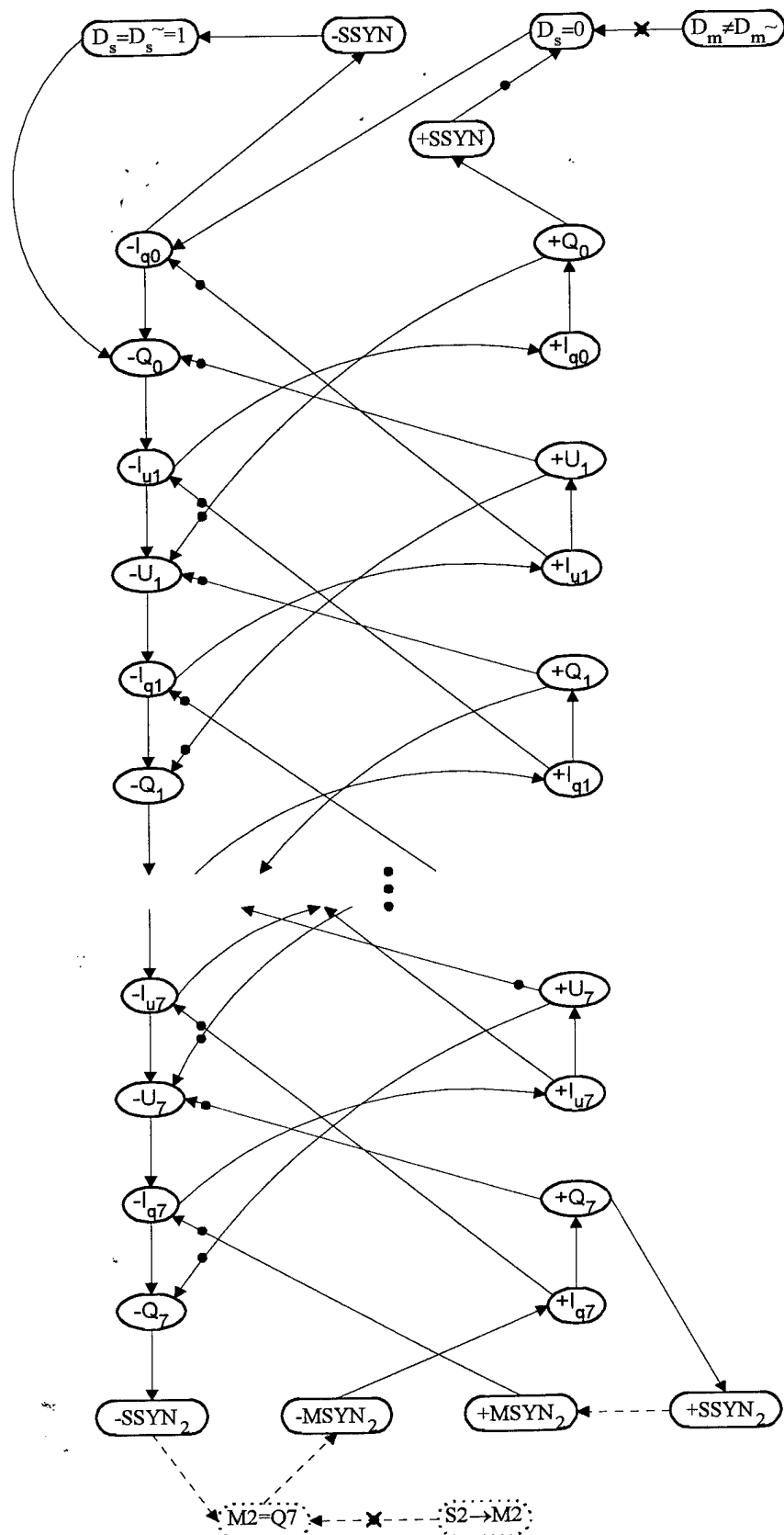
Fig. 1.66. Signal graph of the left and right halves of the undense pipeline register
S^3_FIFO8-1

1.3.3.2.2 Undense pipeline register S^3_FIFO8-2

The undense pipeline register S^3_FIFO8-2, *fig. 1.67*, comprises 15 bits, with information on its inputs and outputs as well as in its cells being paraphase encoded. A parallel code is read out from the outputs Q_i of the full register. Intermediate register bits, with outputs U_i are necessary for shifting information bits. The schemotechnique implementation and operation specificity of the library cells *T3* & *T4* consisting in the register S^3_FIFO8-2 are described in the section 1.2.

A bit indicator fixes transition processes completion within the bit and stores information on the bit state during moving bits along the register. The value $I_{Qi} = 1$ ($I_{Ui} = 1$) witnesses a given register bit is active and contains an information bit. The value $I_{Qi} = 0$ ($I_{Ui} = 0$) witnesses a given register bit is inactive (empty).

The information inputs are disabled, in all register bits but the entrance bit, by the signal Γ_{Qi} (Γ_{Ui}) from the indicator output of the same bit to preserve their current state until an information bit is moved to a next register bit. In the entrance register bit there is no such necessity as the source holds the spacer on its information inputs until an information bit is moved to the 2nd register bit and the entrance bit indicator transits to the state $I_{Q0} = 0$.

Register operation is explained by the signal graph in *fig. 1.68*. The initial state, with all register bits being reset (the signal values are given in the figure in parentheses), is achieved by the reset signal R = 1. The Master must form the spacer state 00 on the information register inputs D_s, $D\tilde{\ }_s$.

After negation of the reset signal, the Master has to set on the information register inputs D_s, $D\tilde{\ }_s$ the information codes either 01 or 10 that makes the signal MSYN = 1 to be generated. The entrance register bit transits to the work state $I_{Q0} = 1$, the write disable signal SSYN = 0 is issued to the source. The source replies with the spacer 00 on the register inputs that enables the entrance bit indicator to transit to the state $I_{Q0} = 0$ after an information bit is moved from the first register bit to second, and the 2nd bit indicator is set to the state $I_{U1} = 1$. Appearance of the value $I_{Q0} = 0$ is followed by the signal SSYN = 1 that enables the Master to set a new information code on the register inputs. The accepted bit is moved to the register output independently from the source. Availability of the information bit in the exit register bit position is fixed by the value $SSYN_2 = 1$. If, by this, the signal $MSYN_2 = 0$, then the information bit is stopped in the exit bit since $MSYN_2 = 0$ confines transition of the exit bit indicator to the spacer.

Fig. 1.68. Signal graph of the undense pipeline register S^3_FIFO8-2

Reading is performed by the read-out request $MSYN_2 = 1$, after which the exit bit indicator transits to the spacer $I_{Q7} = 0$ ($\Gamma_{Q7} = 1$), and the exit register bit is ready to accept information. By this, $SSYN_2 = 0$ that enables the Master to set $MSYN_2 = 1$ and the exit bit indicator to transit to the work state $I_{Q7} = 1$ ($\Gamma_{Q7} = 0$) at coming of a next information bit.

If the register is full then the signal $SSYN = 0$ disables writing. If the register is empty then the signal $SSYN_2 = 0$ disables the output Master to read out information from the register.

The initial reset of the register S^3_FIFO8-2 results in the state $Q_i = U_i = 0$, $Q^\sim_i = U^\sim_i = 1$ after assertion of the reset signal $R = 1$. This circuit is featured by independent parallel setting of indicators of all register bits $I_{Qi} = I_{Ui} = 0$ and of entrance register bit outputs $Q_0 = 0$, $Q^\sim_0 = 1$ while outputs of all other bits are set consecutively during movement of an accepted information bit along the register. Such an action is required, as a rule, only at initial resetting of a circuit and, practically, does not influence its general performance. The hardware overheads and speed characteristics are seen from *table 1-3* in Appendix 1.2.

1.3.3.2.3 The dense pipeline register S^3_FIFO8-3

The dense pipeline register has been proposed in [1.2] and comprises 8 uniform bits, *fig. 1.69*. Information on its inputs and outputs as well as in its cells is represented in a paraphase code. A peculiarity of this register is that the transition between two neighboring states of storing a bit in a cell is possible only via its spacer.

Operation of the register is explained by the signal graph in *fig. 1.70*.

After powered on, register bits are set in an arbitrary state. The FIFO organization of the register results in concentration of states corresponding to storing information bits within last register bits. The initial state (signal values are given in parentheses in the figure) at which all register bits are in the spacer is achieved by a sequence of N pulses on the input $MSYN_2$, with the source keeping the spacer state 11 on the information inputs (D_s, $D^\sim_s$) of the register.

After the initial state is set, the input Master must provide on the inputs D_s, $D^\sim_s$ information codes 01 or 10 followed by the control signal $MSYN = 1$. The entrance bit reacts to it by transition to the work state $C_0 = 1$ followed by the write disable signal $SSYN = 0$ into the source. As a reply, the source must set on the register inputs the spacer 11 bringing the mark flip-flop of the entrance bit in the state $I_0 = 0$.
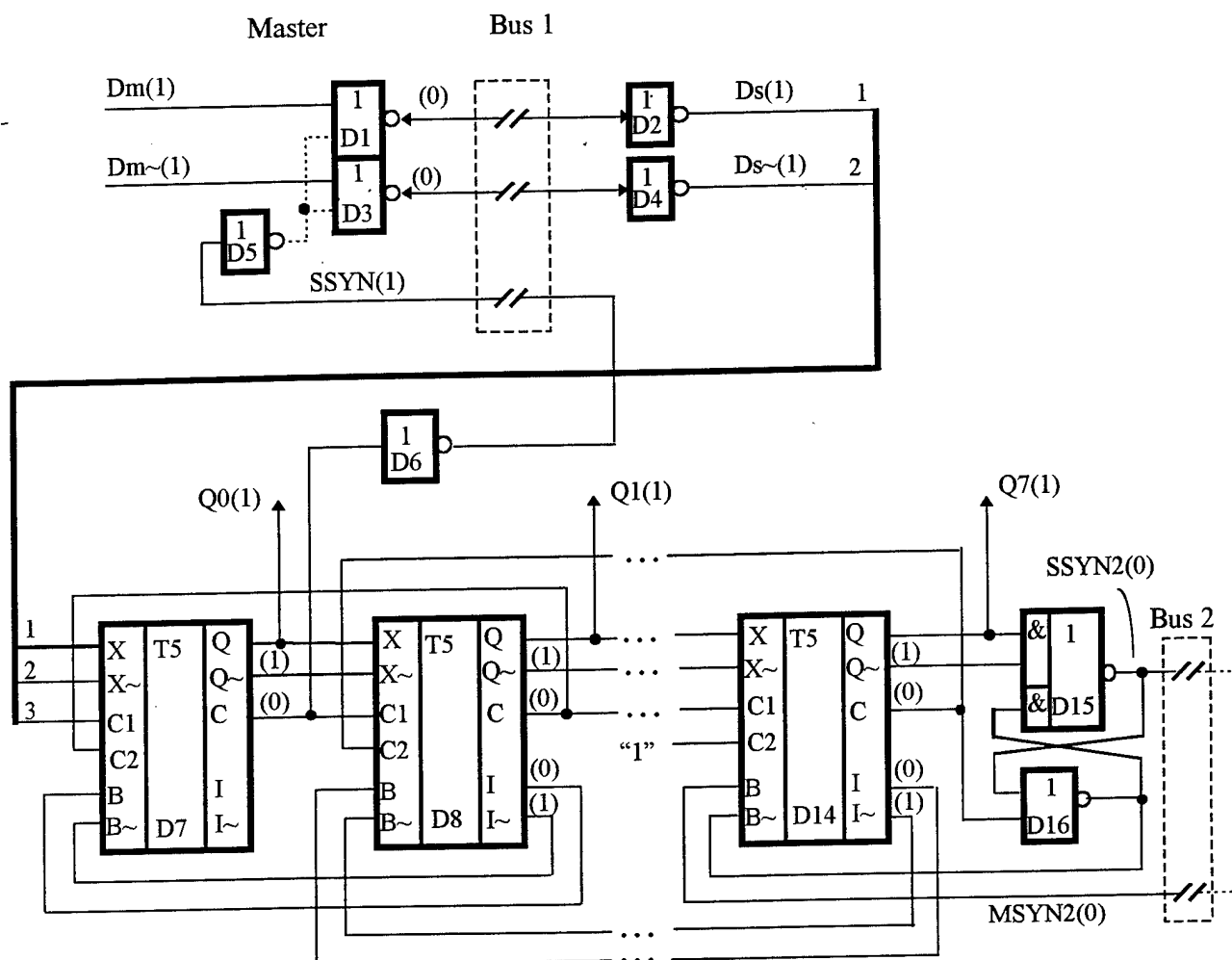
Fig. 1.69. Dense pipeline register S^3_FIFO8-3

After the information bit is shifted out from the entrance bit to a next one, the output C_0 of the tri-stable flip-flop of the entrance bit switches to the state $C_0 = 0$ that makes the entire tri-stable flip-flop to transit to the spacer $Q_0 = Q^\sim_0 = 1$. Appearance of the value $C_0 = 0$ forces the signal $SSYN = 1$ that means, for the Master, permission to set a next information code on the register inputs.

The first information bit moves to the register output independently of the source. Availability of an information bit in the register exit bit is fixed by the value $SSYN_2 = 1$. If, by this, the signal $MSYN_2 = 0$ then the bit is stopped in the exit bit as $MSYN_2 = 0$ prevents transition of the exit bit indicator to the spacer $C = 0$, $Q_7 = Q^\sim_7 = 1$, and the exit bit is ready to accept information again.
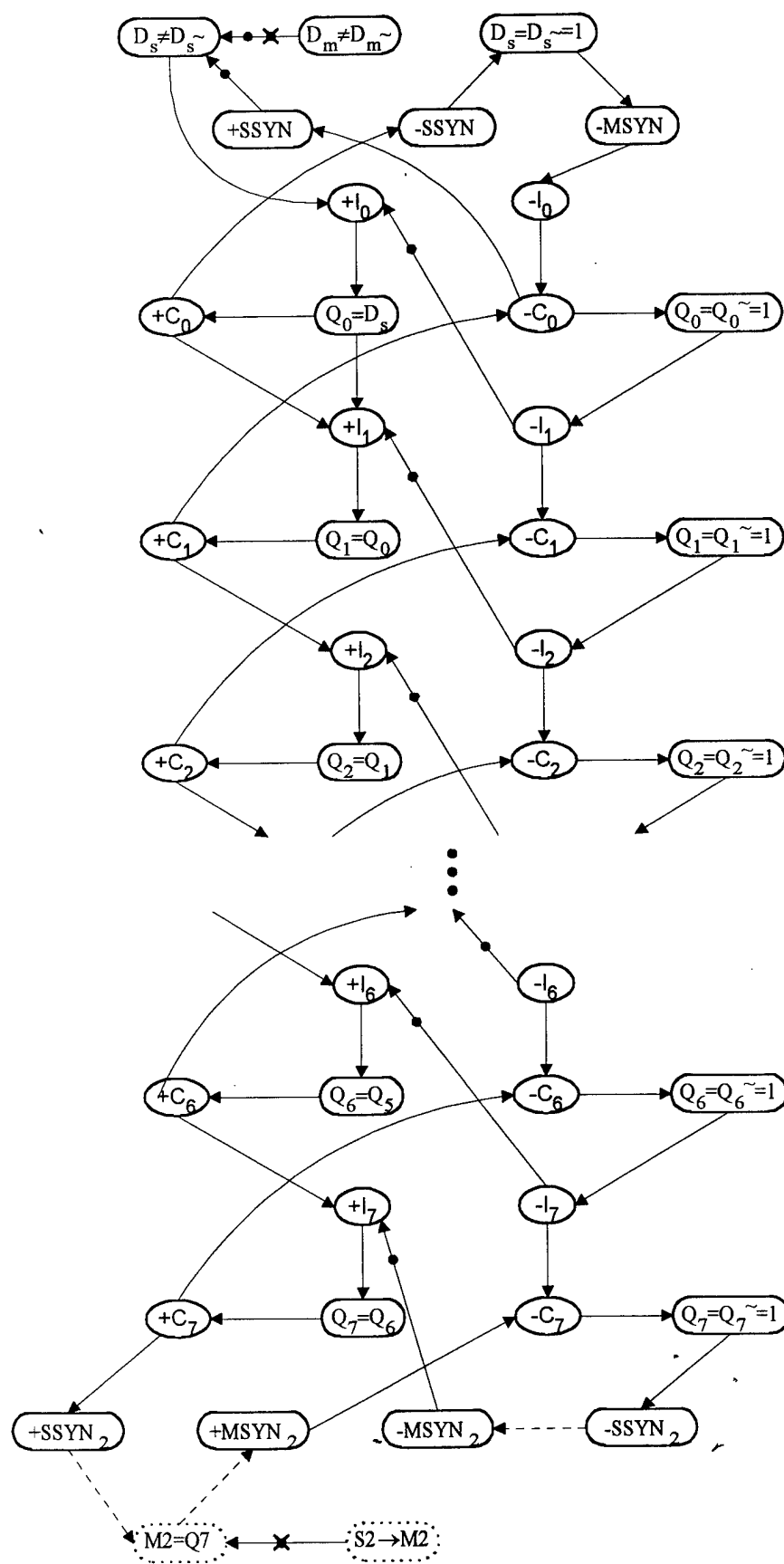
Fig. 1.70. Signal graph of the dense pipeline register S ^{3}_FIFO8-5

By this, $SSYN_2 = 0$ that reports to the Master necessity to set the signal value $MSYN_2 = 1$.

If the register is full, the signal $SSYN = 0$ that disables writing. If the register is empty, then $SSYN_2 = 0$ that disables the output Master to read out information from the register.

Hardware overheads and speed characteristics of the register $S^3\_FIFO8-3$ are given in *table 1-3* of Appendix 1.2.

1.3.3.4 The dense pipeline register $S^3\_FIFO8-4$

The dense pipeline register shown in *fig. 1.71* consists of 8 bits.

Information on register input and outputs as well as in its cells is represented in a paraphase code. Note that, as distinct from the register $S^3\_FIFO8-3$, the information outputs Q_i, $\tilde{Q}_i$ of the register $S^3\_FIFO8-4$ store always a current or previous state (an information bit) transiting dynamically through the spacer 00 only at replacement of the value stored in the register bit. In other words, even if a register bit is considered as being empty ($I_i = 0$), its information outputs store a most recent information bit passed through it.

The register bit state $I_i = A_i = 1$ witnesses the bit is active, contains an information bit, and enables a next register bit to accept this information bit if that is empty. The state $I_i = A_i = 0$ indicated a given register bit is inactive (empty). Other combinations of the outputs of the indicator and bit state flip-flop are intermediate and necessary for correct moving of information along the register.

The entrance bit is distinguished from others as not disabling the information inputs by the signal $\tilde{I}_i$ from the indicator output of the same register bit that prevents any changes of information outputs states until the information bit is moved to a next register bit. The entrance bit avoids this necessity since the source holds on the information inputs of the entrance bit the spacer until the information bit is rewritten to a next register bit, and the entrance bit indicator transits to the state $I_0 = 0$.

Operation of the register is explained by the signal graph in *fig. 1.72*. The register functions as follows. An initial state (see signal values in parentheses) at which all register bits are reset is achieved after a general reset on the input $R = 1$, with the Master providing the spacer 00 on the information inputs D_s, $\tilde{D}_s$ of the register.

Master   Bus 1

Dm(1) ── │1 D1│ ─○─ │1 D2│ ◄── ╱╱ ── │1 D3│ ─○── Ds~(0)   1

Dm~(1) ──────── │1 D4│ ─○─ │1 D5│ ◄── ╱╱ ── │1 D6│ ─○─ Ds(0)   2

│1 D7│   │1 D8│ ─○ │1 D9│ ○ MSYN1   3
                                                              (0)

SSYN(1) ──── ╱╱ ────

R(1) ─────────── ╱╱ ── │1 D10│ ─○─ │1 D11│ ──○ (1)   4
                                    │1 D12│ ─○

│1 D13│ ─○

Q0(0) ↑   Q1(0) ↑   Q2(0) ↑

                                                                          1
                                                                          2
                                                                          3

     │IN T6 I │   │IN   T7 I~│ ○(0)   │IN   T7 I~│ (1)
  ─○ │IN~   │   ○ │IN~   │ (1)   ○ │IN~   │ (0)
 3   │IP   │   │IP   │   │IP   │   4
 2   │D   Q │ (1)   │D   Q │ (1)   │D   Q │ (1)   5
 1   │D~   Q~│ (1)   │D~   Q~│ (1)   │D~   Q~│ (1)   6
     │AN   A │   │AP   A │   │AP   A │
     │R D14   │   │AN   │   │AN   │
 4   │   │   │R D15   │   │R D16   │   7
                                                                          8

                    O7(0) ↑
          1
          2   MSYN2(0)   ╱╱
                    │IN   T8 I │ (1)
          3   │IP   I~│ ○(0)
          4   │D   Q │   │1 D22│ ○ SSYN2
          5   │D~   Q~│   (0)
          6   │AP   A │   ╱╱
          8   │R D21   │
          7   Bus 2
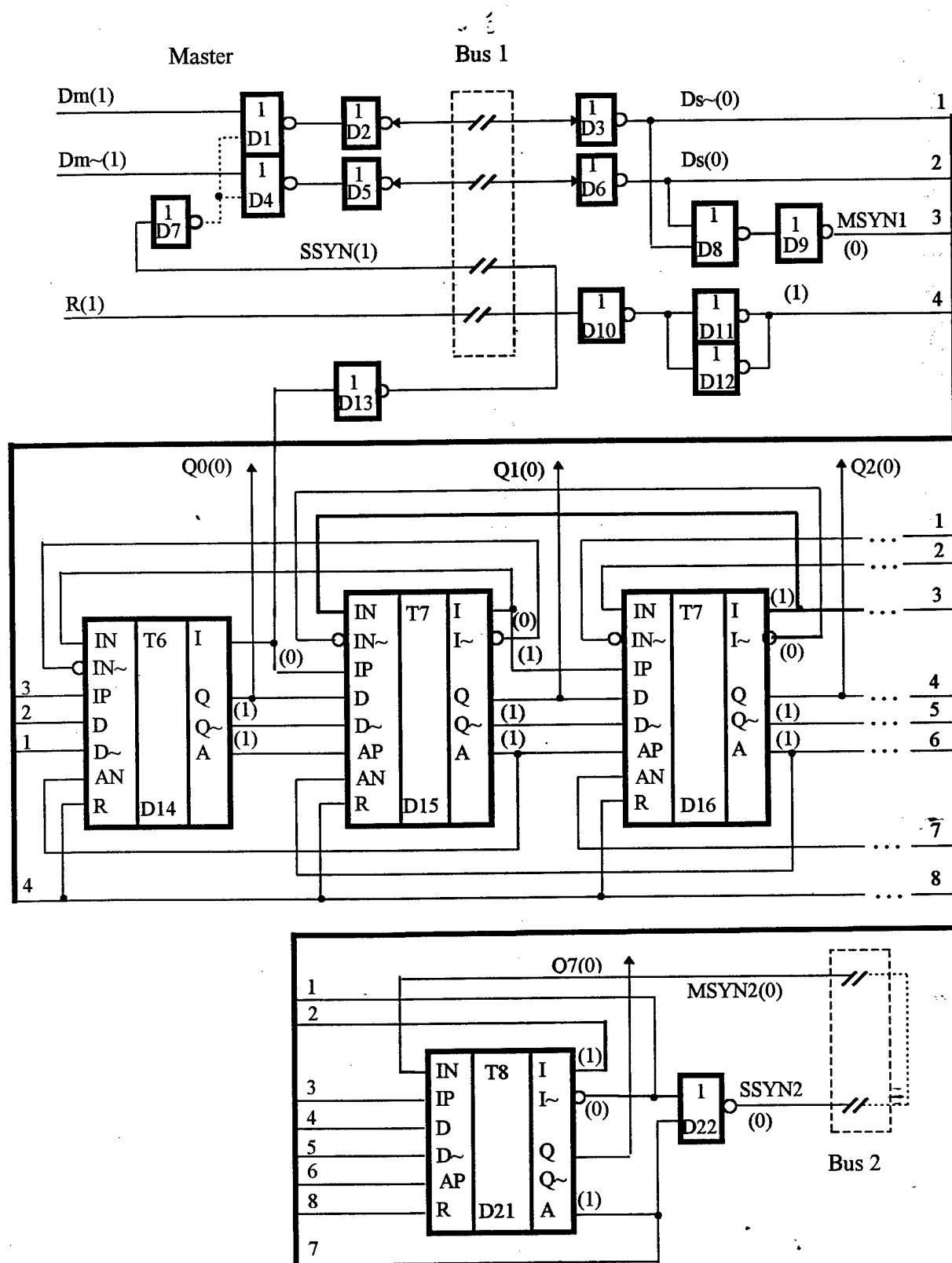
Fig. 1.71. Dense pipeline register S³_FIFO8-4
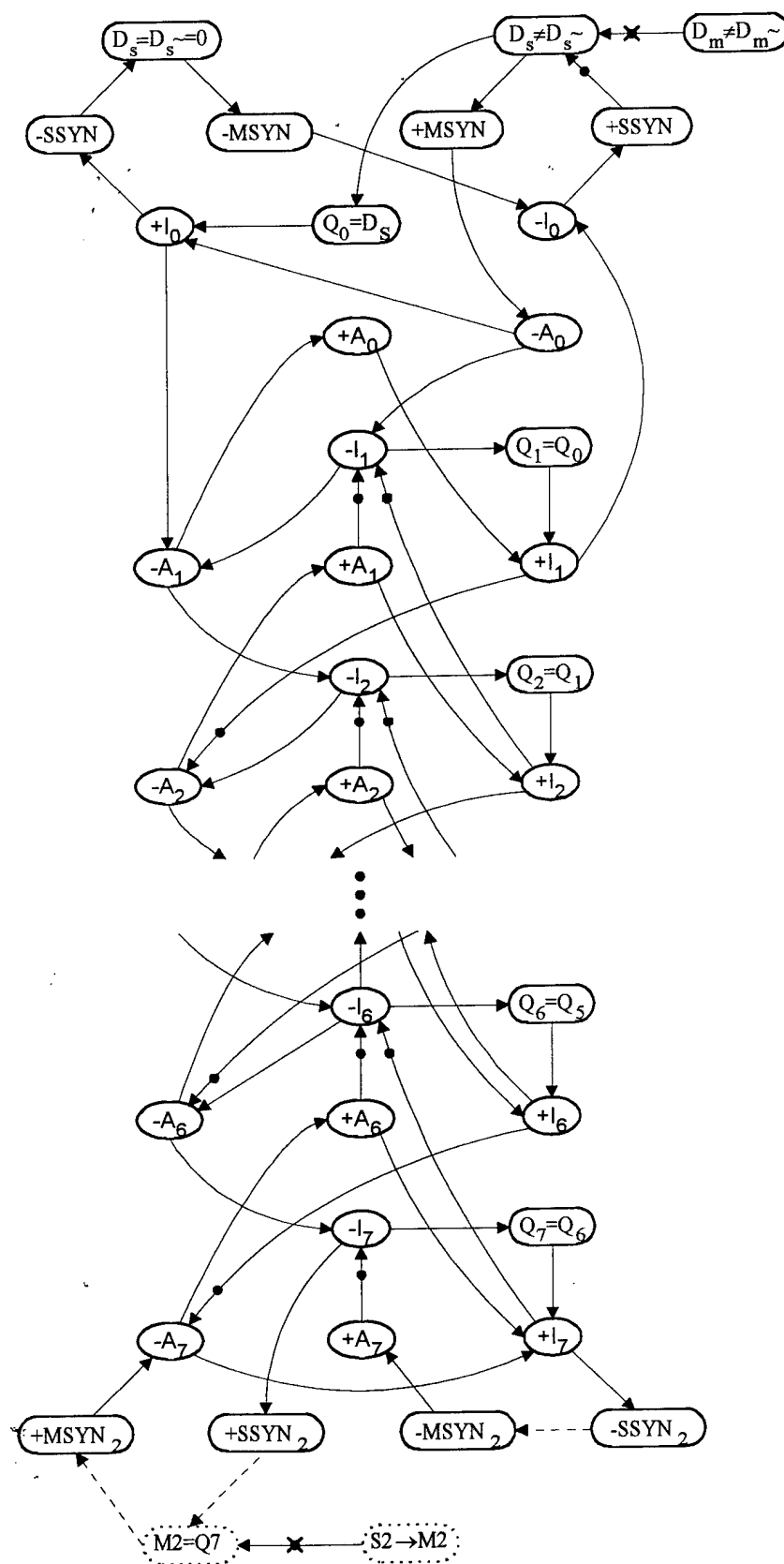
Fig. 1.72. Signal graph of the dense pipeline register S^3_FIFO8-4

After the reset signal is negated, the Master must set on the inputs D_s, $D\tilde{}_s$ information codes 01 or 10 followed by the control signal MSYN = 1. The entrance bit becomes ready to accept information $I_0 = 0$, $A_0 = 0$, and its indicator monitors transition processes completion in the bi-stable cell and transits to the state $I_0 = 1$ that forces the write disable signal SSYN = 0. As a reply, the source has to set the spacer 00 on the register inputs that is one of conditions enabling transition of the entrance bit indicator to the state $I_0 = 0$.

The shift of an information bit from 1st register bit to 2nd bit is accomplished as a result of a number of sequential transitions of the 2nd bit: the 2nd bit indicator is set to the state $I_1 = 0$ enabling: the information bit to be written in the bi-stable cell, the value $A_1 = 0$ to be set on the output of its state flip-flop, and the value $A_0 = 1$ to be set on the output of the entrance bit state flip-flop. The latter value enables indication of completion of rewriting the information bit from the entrance bit to the 2nd bit followed by $I_1 = 1$. In turn, this value allows transition of the entrance bit indicator to the state $I_0 = 0$ that makes the signal SSYN = 1 to appear. After this, the Master is permitted to generate a next information code on the register inputs. The 1st bit is moved along the register independently from the source.

Availability of an information bit in the exit register bit is fixed by the value $SSYN_2 = 1$. If $MSYN_2 = 0$ then the bit is stopped in the last but one register bit since $MSYN_2 = 0$ interferes transition of the exit bit indicator in the work state $I_7 = 1$. The read-out is performed on the read request $MSYN_2 = 1$ followed by transition of the exit bit indicator to the work state $I_7=1$ that enables the read-out. After an information bit is read out, the output Master (a receiver of information from an input Master) must generate the signal $MSYN_2 = 0$ enabling transition of the register exit bit indicator to the state $I_7 = 0$, $A_7 = 1$. Completion of the transition is followed by the signal $SSYN_2 = 0$ that makes the output Master to set the signal value $MSYN_2 = 1$ allowing transition of the exit bit indicator to the work state $I_7 = 1$ at coming of a next information bit.

If the register is full then the signal SSYN = 0 disables writing. If the register is empty then $SSYN_2 = 0$ disables the output Master to read information from the register.

The initial reset of the register S^3_FIFO8-4 by the signal R = 1 provides the source state $Q_i = 0$, $Q\tilde{}_i = 1$. The circuit is featured by independent concurrent initial resetting of the bi-stable cells, indicators, and state flip-flops in all register bits that speeds up the reset. Hardware overheads and speed characteristics are given in *table 1-3* of Appendix 1.2.

1.3.4 The synchronous fault-tolerant code converter

Fig. 1.73 represents the block-diagram of the fault-tolerant code converter **S_SR8-FT** with a hardware redundancy (doubled are the generator G, Bus, and the self-checkable converter).
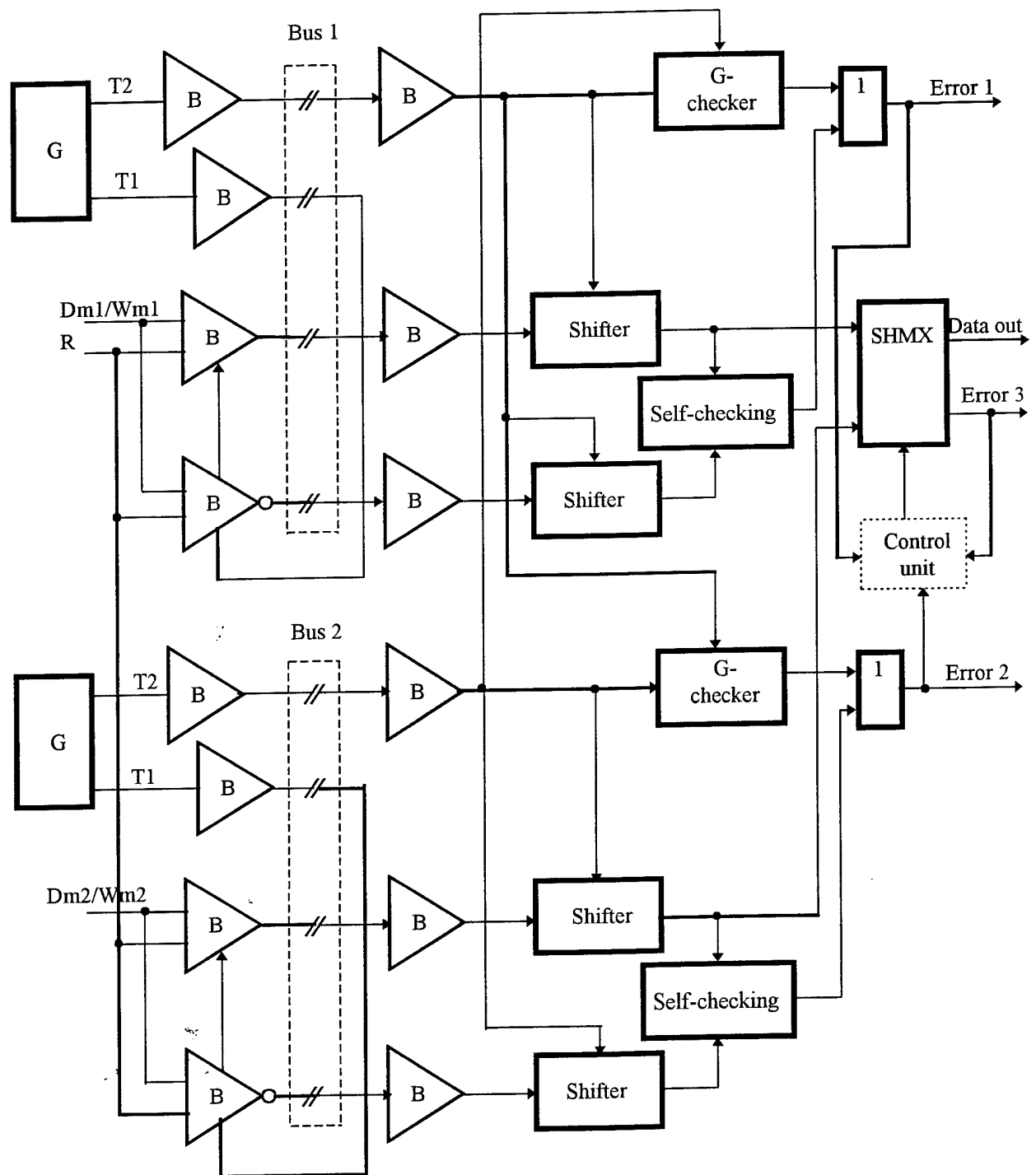
Fig. 1.73. Synchronous fault-tolerant converter S_SR8-FT

The device has two channels: basic (active) and reserve ones. Both channels are based on the self-checkable converter S_SR8-SC (see the subsection 1.3.2). Essential changes have been made in the pulse generator G, and a special circuit (G-checker) has been added to check its efficiency. The G-checker is built with a 3-bit binary counter (*fig. 1.74*, sheet 5). The generator of the same channel resets periodically the counter (with the pulse TX1) incremented by pulses from the generator of another channel. If the generator of the same channel stops, the counter overruns and indicates the generator failure. Thus, a cross-checking is implemented for both generators.

Any failures in the generator, serial interface, or shift register of both channels result in the signals *Error 1* and *Error 2*, respectively. These signals make a special dispatcher to exchange the basic and reserve channels or disable the circuit entirely if there is no efficient reserve channel. For such a switching, the self-checkable multiplexer SHMX is added. The multiplexer applies hardware doubling of MX (D43 and D44) and the self-checkable comparator D45. When a failure in multiplexer operation is revealed, the signal *Error 3* is issued.

The functional diagram of the fault-tolerant converter is shown in *fig. 1.74*, sheets 1 and 2. It comprises the basic and reserve channels, and the data multiplexer. The basic channel consists of the generator D17, shift register D35, D36, G-checker D49, comparator D41, circuits of issuing the error signals D42, D43, D44, D51. Correspondingly, the reserve channel consists of the generator D28, shift register D39, D40, G-checker D50, comparator D45, circuits of issuing the error signals D46, D47, D48, D52. Data are switched between both channels by the self-checkable multiplexer D53, D54, D55.

The timing diagram of correct operation of the converter S_SR8-FT is given in *fig. 1.75*. The generator of the basic channel elaborates the pulse sequences T1_1 and T2_1, and of the reserve channel — T1_2 and T2_2. The leading edge of T2_1 forms reset pulses CHG_R1 for the G-checker of the basic channel. The signal W1 enables receiving of information in the shifter. Data pass the shifter, appear on the outputs $Q0 \div Q7$ and further on the multiplexer outputs $Y0 \div Y7$.

The timing diagram of operation of the device at a failure in the shifter is given in *fig. 1.76*. The failure results in a positive pulse on the output *ERROR 1*.
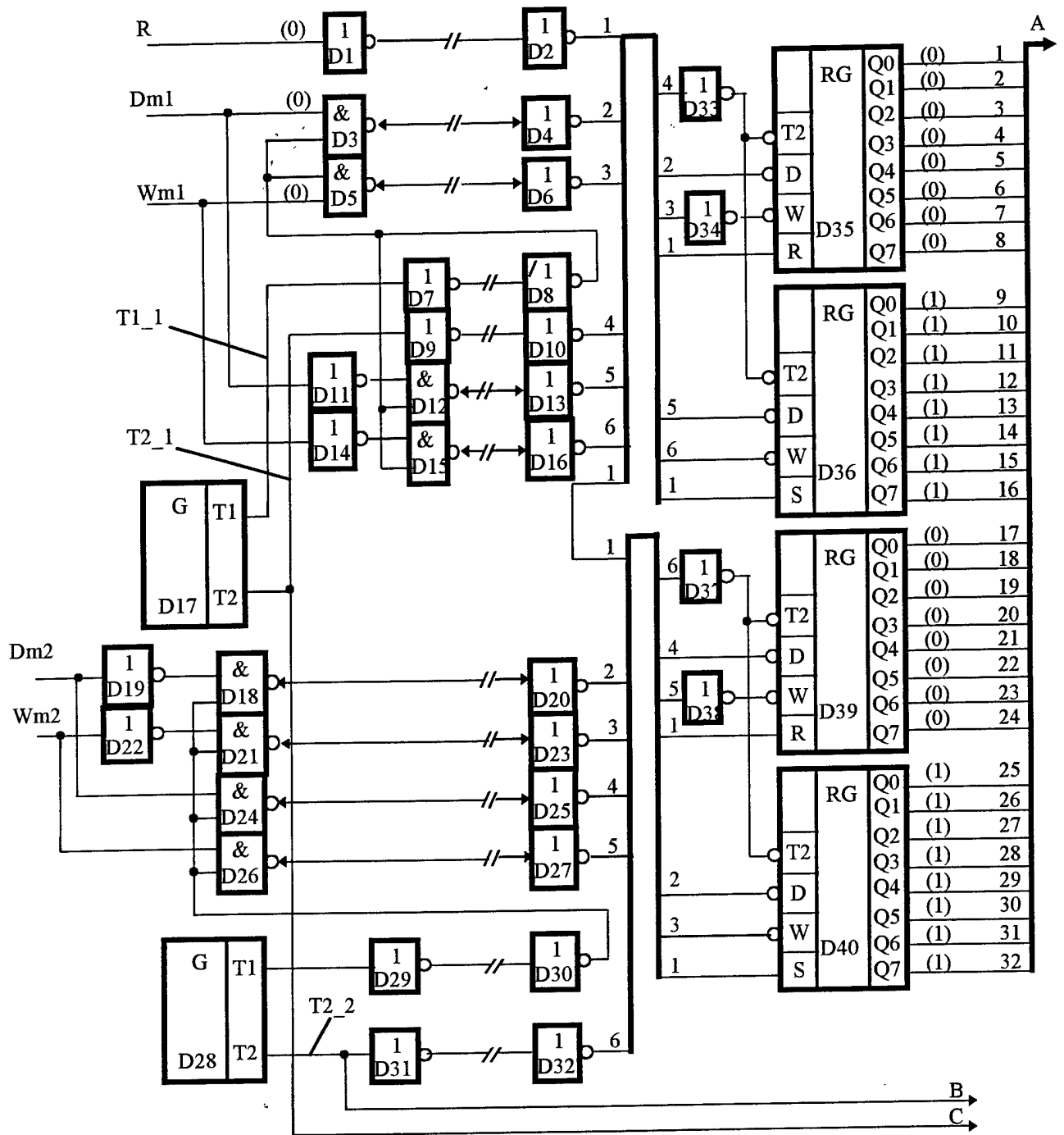
a) Functional diagram (sheet 1 of 2)

Fig. 1.74 (sheet 1 of 5). Synchronous fault-tolerant converter S_SR8-FT
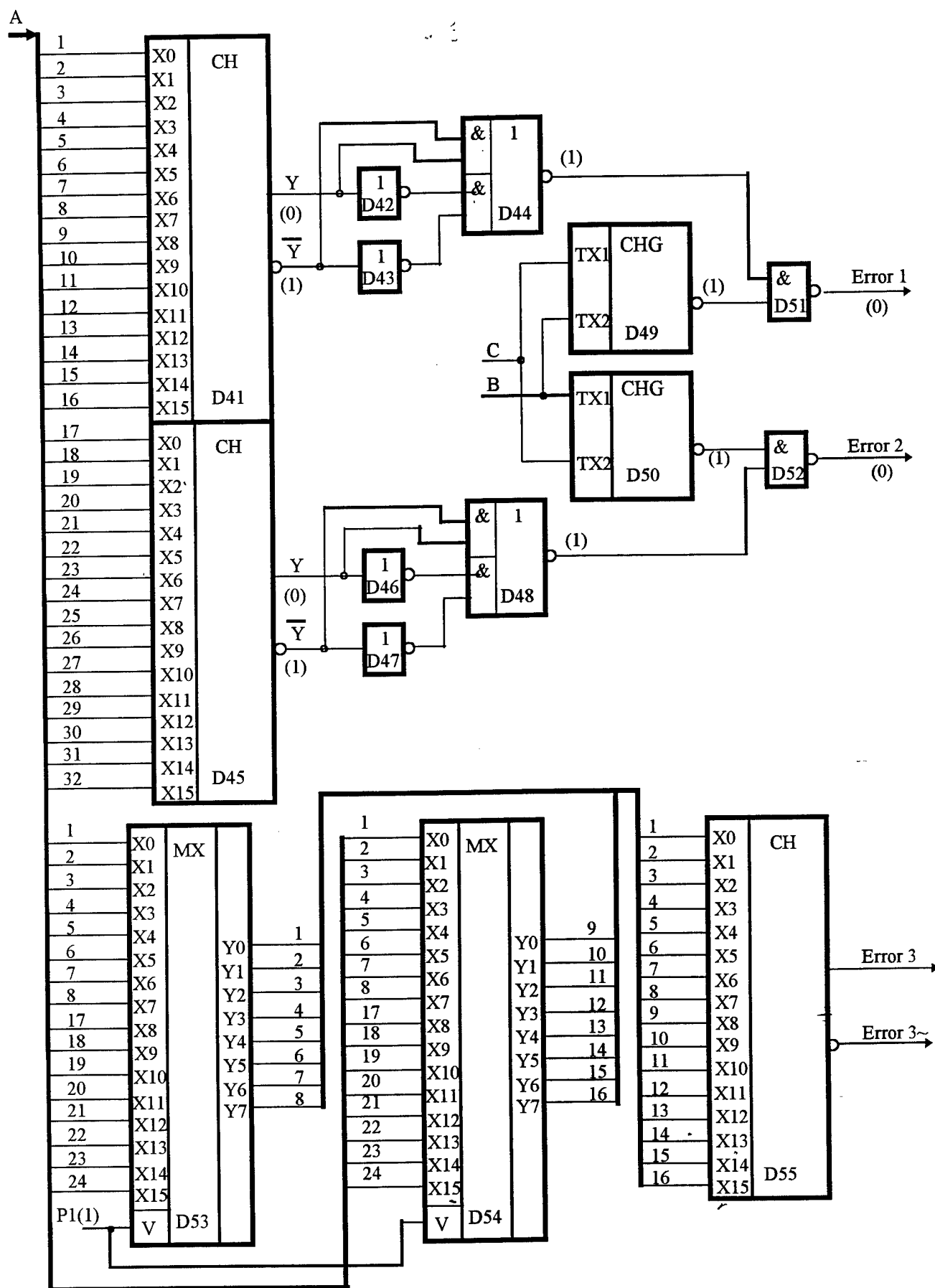
a) Functional diagram (sheet 2 of 2)

Fig. 1.74 (sheet 2 of 5). **Synchronous fault-tolerant converter S_SR8-FT**
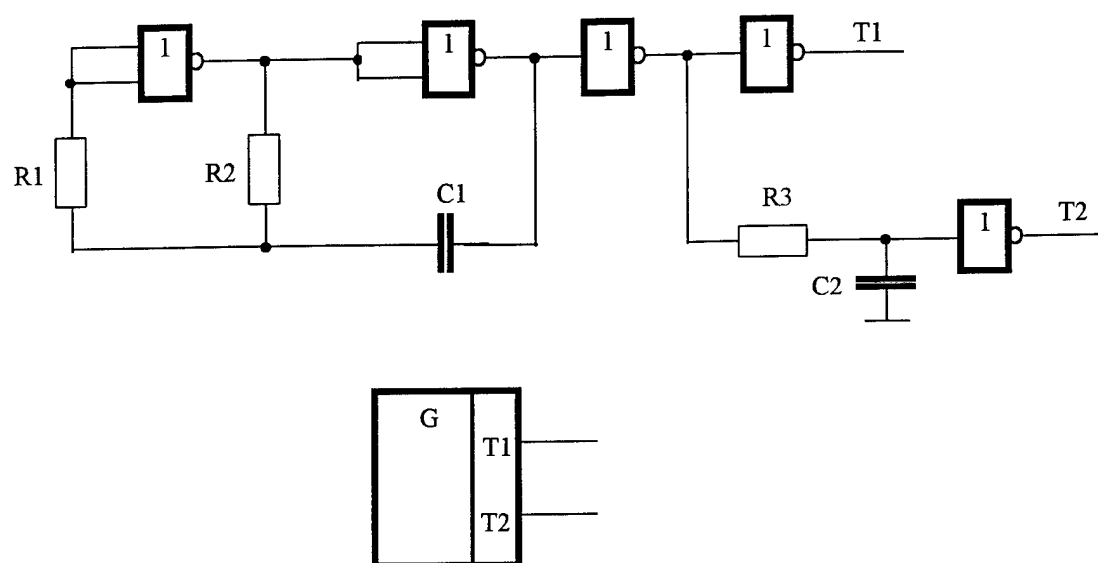
b) Generator

Fig. 1.74 (sheet 3 of 5). Synchronous fault-tolerant converter S_SR8-FT
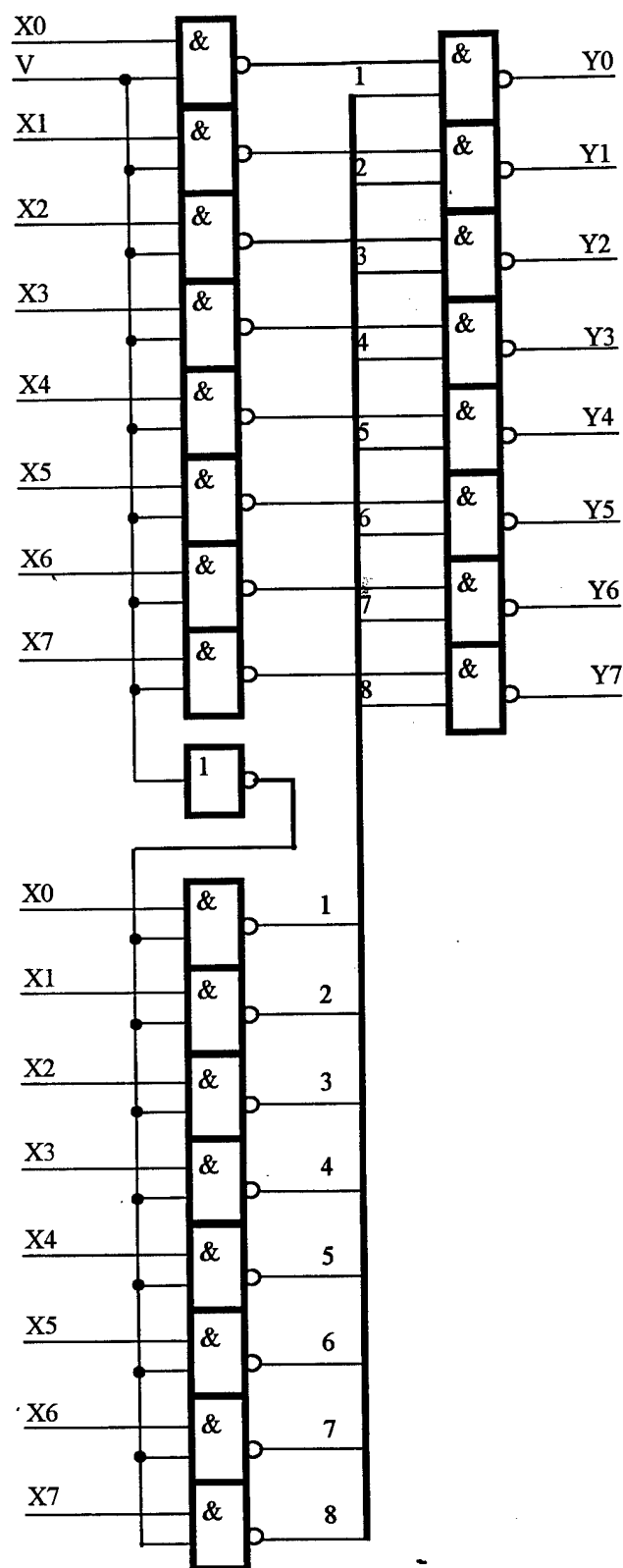
c) Two-input multiplexer
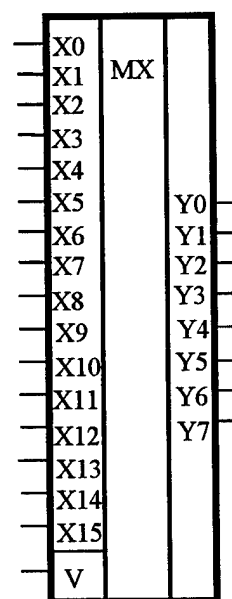
Fig. 1.74 (sheet 4 of 5). Synchronous fault-tolerant converter S_SR8-FT
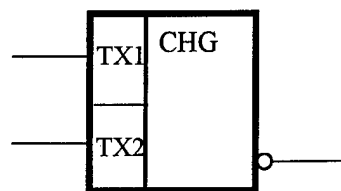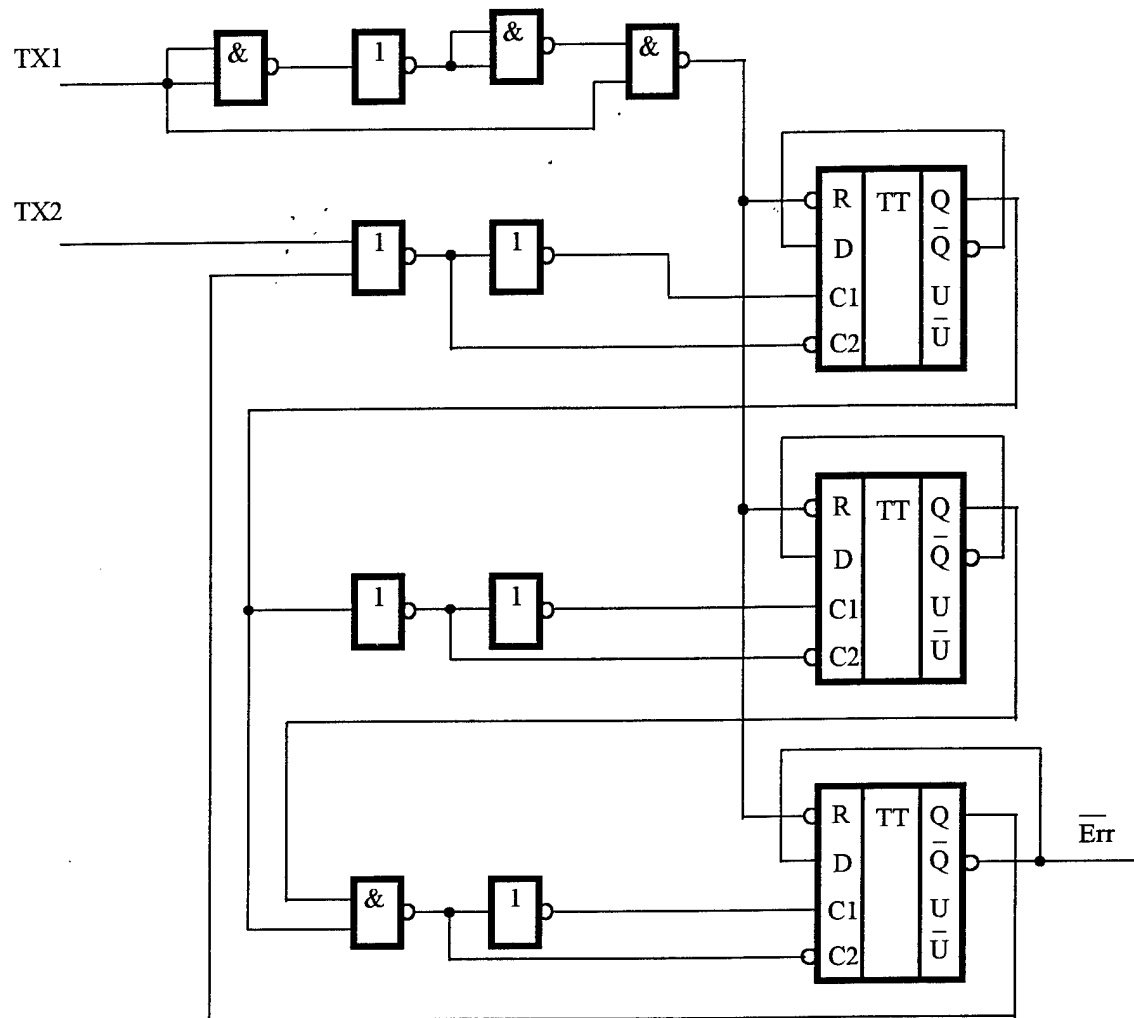
d) G-checker

Fig. 1.74 (sheet 5 of 5). Synchronous fault-tolerant converter S_SR8-FT
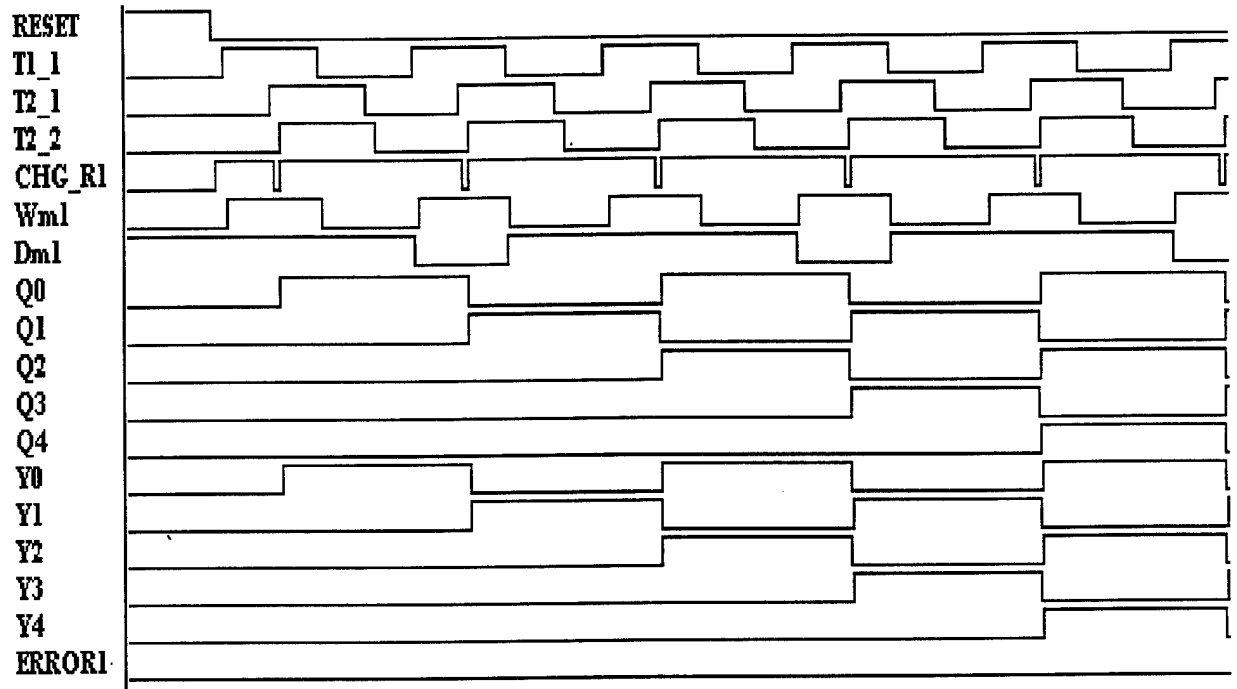
Fig. 1.75. Timing diagram of correct operation of S_SR8-FT
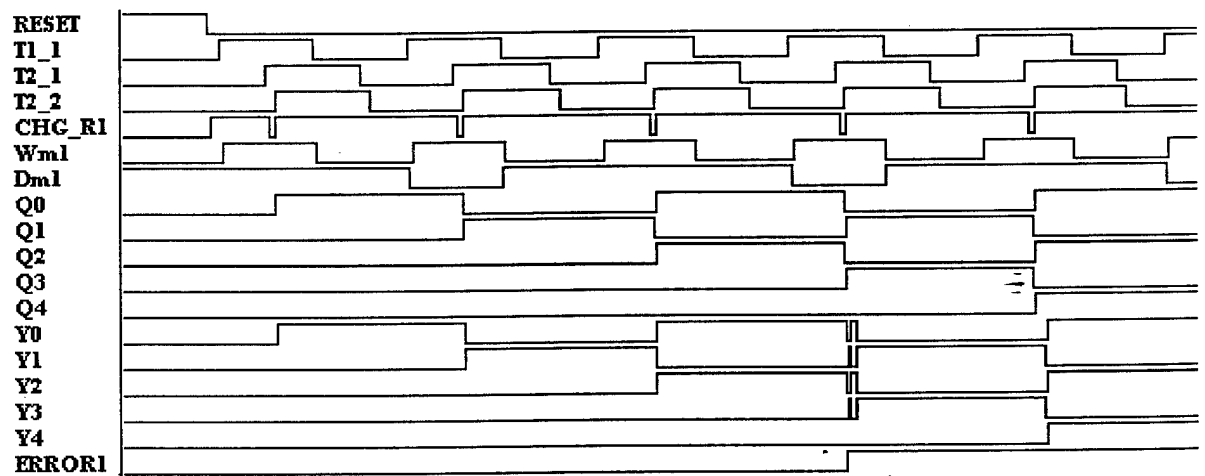
Fig. 1.76. Timing diagram of incorrect operation of S_SR8-FT

1.3.5 The self-synchronous fault-tolerant code converter

The principle of fault-tolerance is implemented in any device by sequential accomplishment of the following steps:

1) diagnosing failures, i.e. revealing errors in device operation

2) localizing faults, i.e. determining error points

3) repairing, i.e. replacing a faulty unit by a reserve one.

The 1st step is made in S^3 circuits automatically that allows them to be classified as *intrinsically self-checkable*. An S^3 circuit can not be not self-checkable and ensures revealing of 100 % of any constant faults, multiple faults included. The competing synchronous self-checkable circuit S_SR8-SC (see the subsection 1.3.2) ensures revealing of single constant faults only.

The 2nd step is implemented in S^3 circuits easily since each indicator exposes any fault in the indicated circuit fragment. Corresponding indicator signals are selected proceeding from requirements of a reservation level. In the variants of the TFD considered earlier the fault localization is provided within a separate converter bit and a specific Muller's C-element that enables reservation at a sufficiently low level. In this case, a reserve unit might be of minimal complexity and hardware expenses of reservation — reduced significantly.

For the synchronous fault-tolerant variant, chosen was a higher reservation level — doubling of a whole TFD. For correctness of comparison of two implementations, synchronous and self-synchronous, the same reservation level was accepted, i.e. an entire TFD was doubled.

This decision is shown in *fig. 1.77*. The circuit consists of two S^3 registers RG8, one 8-bit multiplexer MX8, and two circuits of commutation of indicator signals. Any of S^3 implementations described above are acceptable for RG8. A special control unit keeps an eye on operation of these two registers (the control unit, implemented by hardware or software, is assumed to be outside the circuits under consideration for both synchronous and self-synchronous variants).

A failure in one of the registers RG8 results in no changes of the indicator signal I during some predefined time interval starting since a moment of initiation, by the Master, of a Bus exchange. Let us designate this time-out as T_{t0}. A lower boundary of T_{t0} is chosen at a system level proceeding from the condition: it must not be less than the maximum TFD switching time at upper limit ambient conditions (T = +125° C, V_{CC} = 3 V, N_d = 15).
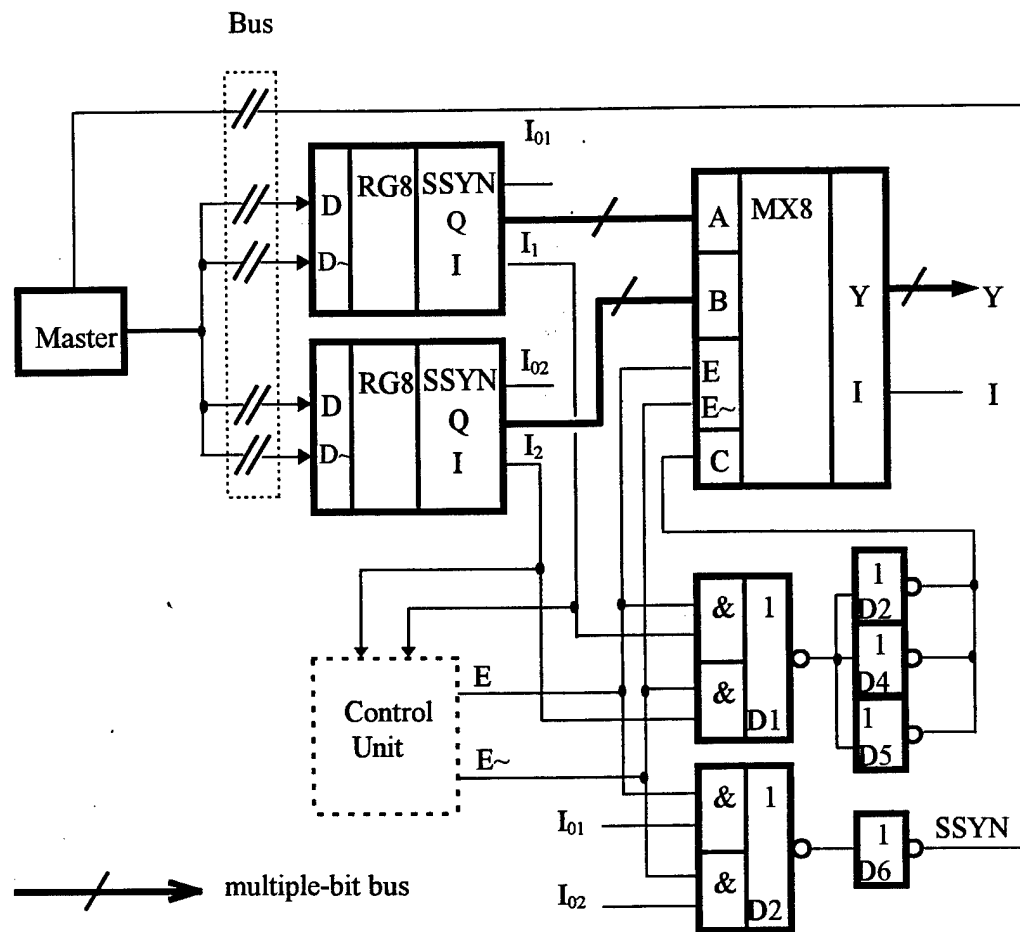
Fig. 1.77. **Self-synchronous fault-tolerant converter**

To reveal faults in S^3 circuits, an upper boundary of the interval T_{t0} must be fixed proceeding from ability of the circuits to provide normal functioning beyond limits of switching characteristics recommended for applied elements by their manufacturers (though with their actual performance degrading). Thus, the upper boundary of T_{t0} is actual for application where lifetime is more important than performance.

The register operates as follows. Resulting from analysis of the indicator signals from both registers, the control unit generates the paraphase signal E, $E\sim$ that replaces a faulty register by an intact one. The work codes 01 and 10 of the signal E, $E\sim$ are exchanged through the intermediate code 00.

The information register inputs are selected by the 8-bit multiplexer MX8. Its inputs A and B are connected to the information outputs of a corresponding shift register. The input E is controlled by an external reservation control circuit. A signal on the input C shows completion of

transition of the shift register to a next phase. The information multiplexer outputs Y repeat outputs of an intact shift register. The indicator output I shows multiplexer operation phase. The information read-out is permitted at $I = 0$ (the work phase). Description of operation of the S^3 multiplexer MX8 is given in the section 1.2.

Interactions with the Master are implemented with one of indicator signals I_{01} and I_{02} coming from entrance register bits after commutation. The general indication signal of the registers are also multiplexed.

Revealing of an error in an active shift register (which outputs are being selected by the multiplexer MX8) seizes its operation via the time-out mechanism. The control unit analyses the indicator outputs of the reserve register and, if the latter is efficient, changes a paraphase code on the selection inputs $E, E\sim$ of the multiplexer and connects the indicator output of the former reserve register to the control input of the multiplexer. If the reserve register is also unable to function, the control unit brings its paraphase control output in the spacer $E = E\sim = 0$ that shuts down logically both registers RG8.

Thus, the described circuit is also strictly self-synchronous and, consequently, dependable in respect to constant faults in not reserved commutation circuits and the multiplexer. This feature is not proper to the synchronous fault-tolerant variant of the TFD S_SRn-FT discussed earlier.

Table 1.4 summarizes technical characteristics of multiplexing equipment (MX8 and D1 ÷ D6) for various circuit bit lengths. The delay of a common indicator can be reduced by using of a multi-input or combined indicator element.

Table 1.4.

Technical characteristics of multiplexing equipment
(@ T = 27° C, V_{CC} = 5 V, N_d =6)

The number of bits in the multiplexer	8	16	32
Additional delay at reading out a parallel code, ns	68	77	91
Additional actual delay of writing one bit, ns	9	5	3
Per bit hardware overheads, transistors	37	36	35

1.4 Conclusions

1) Some different implementation methods for combinational S^3 circuits (the most critical regarding hardware expenses) have been compared. The proposals concern the question how to choose the best implementation method, dependently on the requirements of reliability characteristics of combinational S^3 circuits, the form of logic function being implemented, hardware expenses and the requirements of operational speed.

These proposals will be implemented in the CAD system for S^3 circuits RONIS that is at present under design.

2) Library elements for diverse implementation variants of a test S^3 circuit have been developed. They have higher operational speed and require less transistors then the known schemotechnique solutions of the test element. The improvements have been obtained due to:

— extensive use of tri-state circuits and "weak" inverter in various indicator elements (Muller's C-elements), in the indicator flip-flops (TC-elements), and in the service C-elements (SC-elements);

— efficient implementation of initial reset in the flip-flop cells (there are no reduction in speed of in-line read and write operations);

— implementation of a fast complementary reset of all types of indicator elements.

3) An original approach to the implementation of S^3 interaction between units has been proposed. Based on intracyclic concurrence, the approach extracts the external and internal half-cycles from a general exchange cycle and provides their maximal possible concurrence. The approach:

— compensates, partially or completely (this depends on the exchange conditions), losses of time on implementation of the «request-reply» S^3 exchange;

— accelerates the S^3 exchange between units.

The authors are convinced that a «formal» approach to the S^3 exchange implementation in a number of projects that have been performed abroad was one of the main reasons of failures in increasing performance of S^3 devices as compared with their synchronous analogs. And this important quality has «disappeared» from the list if merits of the S^3 schemotechnique in their projects.

4) A specific test circuit — the serial-to-parallel converter — has been designed in some variants: a self-checkable self-synchronous and a not self-checkable synchronous (S_SR8) implementations. A comparison of the obtained results has been performed.

The comparison on operational speed is based on the "actual speed" — a characteristic that allows an exact and unprejudiced estimation of S^3 circuits operation with respect to real components delay which depend on the real ambient conditions (temperature, voltage, fan-out) and real type of information processed. It has been demonstrated that the actual speed of the S^3 circuits can vary in a wide range. For example, the transfer time for one information bit in the test circuit S^3_SR8-3 varied within from 79 ns (min) to 471 ns (max) that is, respectively, three times better and two times worse than corresponding time in an «ideal» synchronous circuit. Naturally, the concept of the actual speed leads to a concept of actual performance of a computer system which will also have a range of possible values.

The following generalized results of the comparison have been obtained.

a) For the «real» strictly self-synchronous (S^3_SR8-3) and the "ideal" synchronous (S_SR8) circuits, simulation showed:

— on typical operational speed — the S^3 circuit is 1.5 times faster;

— on the number of transistors — the synchronous circuit is 2 times better.

b) For the «real» strictly self-synchronous (S^3_SR8-3) and a «real» synchronous circuits (estimation):

— on typical operational speed — the S^3 circuit is 2 times faster;

— on the number of transistors — the synchronous circuit is 1.9 times better.

c) For the «real» quasi-self-synchronous (QSS_SR8) using the of quasi-self-synchronous flip-flop cell and multi-input composite Γ-flip-flop and the «real» synchronous circuits (estimation):

— on typical operational speed — the quasi-self-synchronous circuit is 2.1 times faster;

— on the number of transistors — the synchronous circuit is 1.5 times better.

Hence, the simulation results and estimations of various schemotechniques demonstrate the S^3 schemotechnique provides noticeably higher operational speed. Therefore, even in application areas where the requirements of reliability are not critical but the high operational speed is required, using of the S^3 schemotechnique may be reasonable.

5) The results of simulation of two functionally similar self-checkable circuits — the «ideal» synchronous (S_SR8-SC) and «real» strictly self-synchronous (S^3_SR8-3) — showed that:

— on typical operational speed — the S^3 circuit is 2.5 times faster;

— on the number of transistors — the S^3 circuit is 1.3 times better.

Hence, using of the S^3 schemotechnique in highly reliable computer systems with possibility of service (breaks for repairing is allowed) is preferable and economically reasonable.
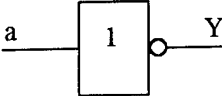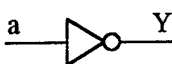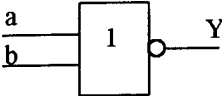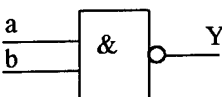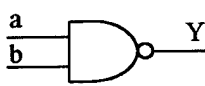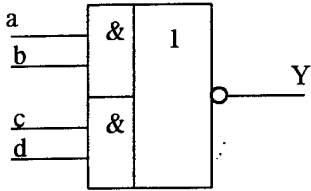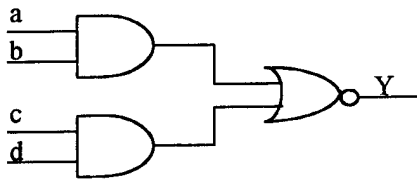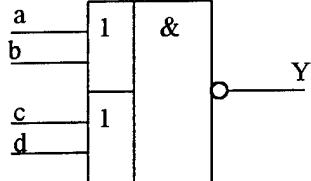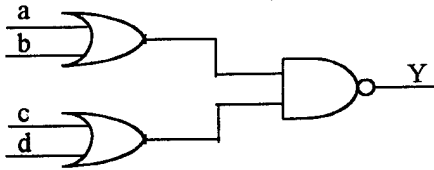
6) The results of simulation of two functionally similar fault-tolerant circuits — the «ideal» synchronous (S_SR8-FT) and the «real» strictly self-synchronous (S^3_SR8-3-FT) — showed:

— on typical operational speed— the S^3 circuit is 2.3 times faster;

— on the number of transistors — the S^3 circuit is 1.3 times better.

Hence, using of the S^3 schemotechnique in highly reliable fault-tolerant real time computer systems is extremely reasonable and much more preferable.

Appendix 1.1

THE GRAPHICAL RULES AND DESIGNATIONS IN THE DIAGRAMS

Used	Conventional	Function

$Y = \overline{a}$

$Y = \overline{a+b}$

$Y = \overline{ab}$

$Y = \overline{ab+cd}$

$Y = \overline{(a+b)(c+d)}$

Appendix 1.2

Table 1.

Register hardware overheads in CMOS transistors (per an information bit)

Register type	Bit length		
	8	**16**	**32**
S_SR*	22	21	21
A_SR*	23	22	21
S_SR*-SC	60	59	58
S_SR*-FT	170	158	151
S^3_SR*-0	48	48	48
S^3_SR*-1	51	50	50
S^3_SR*-1'	51	50	50
S^3_SR*-2	49	49	49
S^3_SR*-3	46	45	45
QSS_SR*-4	43	42	42
S^3_SR*-3-FT	129	126	125
S^3_FIFO*-1	63	63	63
S^3_FIFO*-2	54	54	54
S^3_FIFO*-3	48	48	48
S^3_FIFO*-4	42	42	42

Table 2.

Actual write time in a register bit @ $T = 27^\circ C$, $V_{CC} = 5$ V

The number of Bus loads	Register type	Bit length		
		8	16	32
0	S_SR*	240	243	247
	A_SR*	168	170	173
	S_SR*-SC	408	412	417
	S_SR*-FT	397	401	406
	S^3_SR*-0	244	269	307
	S^3_SR*-1	227	---	---
	S^3_SR*-1'	208	256	265
	S^3_SR*-2	200	241	255
	S^3_SR*-3	184	236	251
	QSS_SR*4	182	222	230
	S^3_SR*-3-FT	184	236	251
	S^3_FIFO*-1	174	174	174
	S^3_FIFO*-2	115	115	115
	S^3_FIFO*-3	294	294	294
	S^3_FIFO*-4	152	152	152
3	S_SR	240	243	247
	A_SR*	267	269	275
	S_SR*-SC	408	412	417
	S_SR*-FT	397	401	406
	S^3_SR*-0	372	396	435
	S^3_SR*-1	274	---	---
	S^3_SR*-1'	267	284	293
	S^3_SR*-2	225	241	255
	S^3_SR*-3	184	236	251
	QSS_SR*-4	182	222	230
	S^3_SR*-3-FT	231	236	251
	S^3_FIFO*-1	285	285	285
	S^3_FIFO*-2	245	245	245
	S^3_FIFO*-3	348	348	348
	FIFO*-4	252	252	252

Table 2 (continued)

The number of Bus loads	Register type	Bit length		
		8	16	32
6	S_SR*	240	243	247
	A_SR*	279	283	288
	S_SR*-SC	408	412	417
	S_SR*-FT	397	401	406
	S^3_SR*-0	382	406	445
	S^3_SR*-1	280	---	---
	S^3_SR*-1'	269	292	298
	S^3_SR*-2	235	241	255
	S^3_SR*-3	191	236	251
	QSS_SR*-4	190	222	230
	S^3_SR*-3-FT	197	236	251
	S^3_FIFO*-1	296	296	296
	S^3_FIFO*-2	255	255	255
	S^3_FIFO*-3	360	360	360
	S^3_FIFO*-4	263	263	263
9	S_SR*	240	243	247
	A_SR*	289	293	298
	S_SR*-SC	408	412	417
	S_SR*-FT	397	401	406
	S^3_SR*-0	392	417	456
	S^3_SR*-1	287	---	---
	S^3_SR*-1'	272	296	302
	S^3_SR*-2	245	244	255
	S^3_SR*-3	199	236	251
	QSS_SR*-4	199	222	230
	S^3_SR*-3-FT	205	236	251
	S^3_FIFO*-1	307	307	307
	S^3_FIFO*-2	266	266	266
	S^3_FIFO*-3	371	371	371
	S^3_FIFO*-4	274	274	274

Table 2 (continued)

The number of Bus loads	Register type	Bit length		
		8	16	32
12	S_SR*	240	243	247
	A_SR*	301	305	400
	S_SR*-SC	408	412	417
	S_SR*-FT	397	401	406
	S^3_SR*-0	402	426	465
	S^3_SR*-1	296	---	---
	S^3_SR*-1'	279	301	308
	S^3_SR*-2	255	256	256
	S^3_SR*-3	207	236	251
	QSS_SR*-4	207	222	230
	S^3_SR*-3-FT	213	236	251
	S^3_FIFO*-1	318	318	318
	S^3_FIFO*-2	276	276	276
	S^3_FIFO*-3	382	382	382
	S^3_FIFO*-4	285	285	285
15	S_SR*	240	243	247
	A_SR*	313	317	324
	S_SR*-SC	408	412	417
	S_SR*-FT	397	401	406
	S^3_SR*-0	412	437	475
	S^3_SR*-1	302	---	---
	S^3_SR*-1'	287	305	311
	S^3_SR*-2	266	266	266
	S^3_SR*-3	215	236	251
	QSS_SR*-4	215	222	230
	S^3_SR*-3-FT	221	236	251
	S^3_FIFO*-1	330	330	330
	S^3_FIFO*-2	289	289	289
	S^3_FIFO*-3	394	394	394
	S^3_FIFO*-4	297	297	297

Table 3.

Actual time of accepting a serial code in a register @ $T = 27^\circ C$, $V_{CC} = 5$ V

The number of Bus loads	Register type	Bit length		
		8	16	32
0	S_SR*	1920	3928	7904
	A_SR*	1344	2720	5536
	S_SR*SC	3264	6592	13344
	S_SR*-FT	3176	5728	10928
	S^3_SR*-0	1770	4094	9580
	S^3_SR*-1	1637	---	---
	S^3_SR*-1'	1490	3928	8341
	S^3_SR*-2	1440	3681	7836
	S^3_SR*-3	1320	3685	8000
	QSS_SR*-4	1306	3461	7328
	S^3_SR*-3-FT	1388	3762	8091
	S^3_FIFO*-1	980	2080	4275
	S^3_FIFO*-2	765	1600	3270
	S^3_FIFO*-3	2055	4285	8745
	S^3_FIFO*-4	1070	2265	4650
3	S_SR*	1920	3928	7904
	A_SR*	2136	4304	8700
	S_SR*-SC	3264	6592	13344
	S_SR*-FT	3176	5728	10928
	S^3_SR*-0	2658	5998	13516
	S^3_SR*-1	1971	---	---
	S^3_SR*-1'	1887	4387	9107
	S^3_SR*-2	1632	3660	7908
	S^3_SR*-3	1255	3500	7940
	QSS_SR*-4	1241	3290	7289
	S^3_SR*-3-FT	1323	3577	8031
	S^3_FIFO*-1	1770	3780	7785
	S^3_FIFO*-2	1660	3515	7230
	S^3_FIFO*-3	2470	5135	10465
	S^3_FIFO*-4	1720	3650	7510

Table 3 (continued)

The number of Bus loads	Register type	Bit length		
		8	16	32
6	S_SR*	1920	3928	7904
	A_SR*	2232	4528	9216
	S_SR*-SC	3264	6592	13344
	S_SR*-FT	3176	5728	10928
	S^3_SR*-0	2730	6152	13834
	S^3_SR*-1	2051	---	---
	S^3_SR*-1'	1905	4426	9246
	S^3_SR*-2	1699	3708	7924
	S^3_SR*-3	1310	3500	7935
	QSS_SR*-4	1303	3290	7289
	S^3_SR*-3-FT	1378	3577	8026
	S^3_FIFO*-1	1840	3935	8105
	S^3_FIFO*-2	1730	3670	7550
	S^3_FIFO*-3	2545	5290	10785
	S^3_FIFO*-4	1790	3805	7830
9	S_SR*	1920	3928	7904
	A_SR*	2312	4688	9536
	S_SR*-SC	3264	6592	13344
	S_SR*-FT	3176	5728	10928
	S^3_SR*-0	2802	6306	14151
	S^3_SR*-1	2146	---	---
	S^3_SR*-1'	1934	4516	9488
	S^3_SR*-2	1765	3756	7940
	S^3_SR*-3	1365	3500	7930
	QSS_SR*-4	1365	3290	7279
	S^3_SR*-3-FT	1433	3667	8021
	S^3_FIFO*-1	1915	4085	8420
	S^3_FIFO*-2	1800	3825	7865
	S^3_FIFO*-3	2620	5440	11105
	S^3_FIFO*-4	1865	3960	8146

Table 3 (continued)

The number of Bus loads	Register type	Bit length		
		8	16	32
12	S_SR*	1920	3928	7904
	A_SR*	2408	4816	9632
	S_SR*-SC	3264	6592	13344
	S_SR*-FT	3176	5728	10928
	S^3_SR*-0	2874	6460	14469
	S^3_SR*-1	2256	---	---
	S^3_SR*-1'	2020	4552	9656
	S^3_SR*-2	1837	3900	8125
	S^3_SR*-3	1425	3505	7925
	QSS_SR*-4	1425	3295	7274
	S^3_SR*-3-FT	1593	3582	8016
	S^3_FIFO*-1	2035	4240	8740
	S^3_FIFO*-2	1875	3975	8185
	S^3_FIFO*-3	2695	5605	11430
	S^3_FIFO*-4	1935	4115	8465
15	S_SR*	1920	3928	7904
	A_SR*	2504	5072	10368
	S_SR*-SC	3264	6592	13344
	S_SR*-FT	3176	5728	10928
	S^3_SR*-0	2945	6613	14786
	S^3_SR*-1	2337	---	---
	S^3_SR*-1'	2119	4673	9733
	S^3_SR*-2	1908	4043	8309
	S^3_SR*-3	1485	3510	7920
	QSS_SR*-4	1485	3300	7269
	S^3_SR*-3-FT	1553	3587	8011
	S^3_FIFO*-1	2159	4390	9055
	S^3_FIFO*-2	1945	4130	8500
	S^3_FIFO*-3	2770	5440	11750
	S^3_FIFO*-4	2008	4265	8780

2 ALGORITHMS OF ANALYSIS AND SYNTHESIS OF S^3 CIRCUITS

This chapter is dedicated to description of methods and implementation algorithms based on the Muller's theory and change diagrams (CDs) [1.2].

The concept was implemented in the S^3 VLSI CAD system *FORCAGE* 3.0 and its further evolution BTRAN. The proprietor of FORCAGE and BTRAN is IPI RAN.

The detailed description of algorithms taken as a principle of both CAD systems is given below.

A main distinctive feature of these algorithms is strict compliance with the theory of self-synchronous circuits and their fundamental character. This represents a solid basis for further development of both algorithmic base and software implementation.

This chapter formulates a theoretical concept and ways to pass, in description of S^3 circuits, from the Muller's transition diagrams (TDs) to CDs. The concept enables analysis and synthesis of S^3 circuits in the TD basis to avoid exponential growing of computation complexity and create a powerful S^3 VLSI CAD tool.

Essential advancement in this direction had been made by the Varshavsky's group [2.1], many questions were however left unsolved, specifically:

• development of algorithms and programs of circuit analysis on semi-modularity, on base of CD description;

• development of an interface with VLSI designers (input languages, exception diagnostics, representation and display of results).

The exposed material reflects only a general «ideology» of S^3 circuit analysis and synthesis, it does not touch upon the known methods as well as specific details of the used algorithms. Not considered is also a more general class of application of the given theory to analysis of parallel processes and organization of parallel computing structures.

2.1   Possible approaches to analysis of S³ circuits

The notion of analysis of S³ circuits permits two approaches: "narrow" and "broad".

The "narrow" understanding is reduced to analysis of a system of Boolean equations describing a specific schemotechnique solution of a self-synchronous unit for the purpose of checking its belonging to the class of semi-modular circuits (see subsection 2.1.1 and section 2.2).

The "broad" understanding comprises analysis of behavior model describing a self-synchronous process. This approach requires a specific set of language means for description of semi-modular processes and enables for synthesis of particular schemotechnique solutions derivative from analysis of S³ processes (see subsection 2.1.2 and section 2.3).

2.1.1   Analysis of S³ circuits on base of Muller's diagrams

2.1.1.1 · The Muller's model

The Muller's model was suggested for description of switched circuits and processes in them. It represents an abstraction of a logic circuit, in which each element consists of a functional transformer "evaluating" switching conditions of the element and a delay simulating a time interval between appearance of these conditions on element inputs and changing of the element output. The delay must be of arbitrary but finite value.

The delays in an element and its output wire up to a branching point are considered to be reduced (brought) to the element output. Delay differences in wires after the branching point are regarded as negligible (in practice, sufficient is to consider them to be equal in all wires). If necessary, the delays in wires are simulated by insertion of additional repeaters.

Switching of any element is asynchronous and controlled only by its inputs. Accordingly, diverse elements can switch simultaneously, and a circuit may be considered as a parallel process of multiple switchings of elements represented in a description by corresponding variables. This explains why any S³ circuit can be treated, in the Muller's model, as a formalization of a real parallel process. A more strictly formalized description of the Muller's model was given in our first report.

The parallel process must be controllable, with the controllability assuming any of permissible sequences to result in the same final state from the same initial state. In other words, the behavior of a system must not depend on relative durations of "work phases" of its

components, including also components functioning in parallel. Refusal from explicit presence of time in the description of system behavior underlies semantics of partial order in parallelism. The Muller's model uses this partial order for describing behavior of circuits, that do not dependent on element switching speed, by the transition diagrams.

Behavior of a circuit is reflected in the TDs by a sequence of changes of full states. Each state (or pair of states) brings information about those variables that are excited (are able to switch) in this state. Therefore, one of basic problems of parallelism — the concurrency problem (i.e. revealing variables that can switch in parallel) — is here trivial and localized within each state. This incontestable advantage allows the TDs to describe easily parallel behavior. The number of excited variables in each of states reflects the number of components functioning in parallel.

In the TDs, nothing is known a priori about time durations of changing of an excited variable. Therefore, in TDs there are present all states generated by various distributions of those durations, and a TD defines all permissible sequences for a circuit.

The choice of a model representation of parallelism is decisive for the "resolving capability" of all subsequent researches. Expressive possibilities of the model depend on detailing degree of a description. The more it is detailed (and, hence, more bulky), the finer properties can be exposed.

Cumulative diagrams built on TDs enabled revealing of TD algebraic properties and classification of behavior of circuits in dependence on kinds and degree of parallelism (semi-modular, distributive, parallel-sequential, and sequential TDs).

2.1.1.2 Properties of S^3 circuits

One of most essential outcomes of the theory of self-synchronous circuits is the fact that the semi-modularity of corresponding TDs is a sufficient condition for such circuits to ensure their ability to function properly. Further, all main merits of self-synchronous circuits are consequent upon their ability to function properly independently of delay values of elements consisting in the circuits. Checking of circuit properties providing such a functioning is one of key tasks of analysis on self-synchronousity.

At this approach, analysis is performed "from a circuit". An autonomous circuit is chosen as a source object, and information on its properties is a result of the analysis (circuits with external inputs are made autonomous artificially by embedding in a model environment). Here, it is possible to separate the following tasks of analysis on correctness of time behavior for a circuit:

- check on independence of element delays

- check on independence of delays in wires

- classification analysis (whether the circuit belongs to the semi-modular, distributive, or sequential classes).

All incorrectnesses in the behavior bringing a circuit out of one or another subclass of semi-modular circuits can be localized in specific states. For example, all violations of the semi-modularity can be localized in "collision states", in which switching of a set of elements removes excitation from some other element. Moreover, sufficient is to consider only single-collision states, in which switching of a single element removes excitation from some another.

Thus, analysis on belonging to one or another semi-modularity subclass is divided in two stages:

1. Check of incorrectness (disposition to collisions, detonation, etc.).

2. Solution of achievability tasks.

The following statements simplify essentially the incorrectness check procedure:

- a circuit is semi-modular with regard to a state *s*, if and only if no one of single-collision states is achievable, on neighbors, from *s*

- a circuit is distributive with regard to a state *s*, if and only if no single-collision and detonant states are achievable, on neighbors, from *s*.

Building of a multitude of achievable states for a circuit is convenient to perform by building of its TD. The TD representations may differ in this case. For example, a functional representation of TD definition may be used when the multitude of TD states is defined by characteristic Boolean functions.

At this approach, the iterative building of a TD consists in using of a functional operator of the direct achievability which, for a considered multitude of states given by a characteristic function, defines a characteristic function of a multitude of immediate successors.

Conventional building of a TD as a graph is also possible under circuit analysis. In this case, generation of TD vertices (i.e. circuit states) is implemented iteratively, beginning from an initial state, by switching excited variables.

The violation states (collision, detonant, and others) can be calculated in the functional form as well. Then, the task of checking of a circuit on correctness is reduced to confirming inachievability of any of "bad" states. In practice, this means checking on absence of intersections

of the characteristic function of a multitude of achievable states with a multitude of incorrect states. At the iterative solution of the achievability task, it is advisable to alternate increments of the multitude of work states with checking of collisions in them.

The most time-consuming operation of the analysis of a work multitude is determination of calculation termination conditions, i.e. determination the fact that all multitude of work states is formed up.

Since the multitude of layers is put in order and the number of circuit states is finite, the layers begin to repeat (multiple layers) from some step of building of the multitude of work states. Appearance of a multiple layer can signal termination of the process of building of a work multitude.

Some part of the work states can appear, at functioning of a circuit, only once (an initial TD section), all other states are repeated cyclically (a work TD cycle). The problem of selection of a candidate on the role of a *check-layer* (that can be surely predicted to be repeated further) is solved at creation of automated analysis systems with empirical methods and means of interactions with the user that enables the system to store only three TD layers: the check-layer, a current layer, and a layer being built.

Transition from a "full" analysis to the analysis of the work multitude and the layer organization of algorithms allows softening of the "dimension damnation" and application of the TD analysis system for checking of real circuits (up to hundreds of thousands of states in a work multitude).

However, and this is important to stress, all TD analysis methods do not avoid the exponential growth of computation complexity versus the number of states checked.

These algorithms are implemented in the system TRANAL designed by the Varshavsky's group.

2.1.2   Possible approaches to CD-based analysis of S^3 circuits

The CDs have been proposed in [1.2] as a language means to describe parallel processes for analysis of semi-modular circuits behavior. As distinct to the TDs, they are not models in full states and describe cause-consequence relationships between events (switchings) in a circuit. This allows computations to avoid the exponential complexity of process models but complicates essentially the procedure of parallelism analysis since influence of detonant zones becomes "spread" along a whole CD.

Change diagram is named a structure <A, I, →, ⊢, M, L> where:

- A — a multitude of changes (events) including fictitious ones
- I — initial events from an events multitude A
- → — relationship of "strong" precedence on the events multitude A
- ⊢ — relationship of "weak" precedence on the events multitude A
- M — a multitude of initially active links (arcs) between events of the events multitude A
- L — a multitude of disconnectable links (arcs) between events of the events multitude A
- $M \cap L = 0$.

It has been mentioned that a TD is constructed for a closed circuit with regard to interactions with its environment. Likewise, a corresponding CD describes a cyclic process with regard to reaction of the environment.

2.1.2.2 Interpretation of functioning of cyclic CDs

For description of CD functioning, the arcs activity is convenient to consider as variable.

The event AND comes first time if all its input arcs have a positive activity. The subsequent event AND is possible if all its input undisconnectable arcs have a positive activity.

The event OR comes if at least one of its input arcs has a positive activity.

The coming of any event decrements the activity of input arcs by 1 and increments by 1 activity of output arcs.

Note.   The stated mark-out is descriptive and not followed by necessity to build up a "mark-out tree" at analysis of CDs.

2.1.2.3 Additional CD restrictions

Since the CD model was announced as a specification of circuit processes, for description of the semi-modularity there are required additional restrictions on the structure and properties of CDs intended for describing real processes in VLSIs.

1. For the relationship of equation between events, apart from conventional properties of reflexivity, symmetry, and transitivity, added are the following restrictions:

- equated are changes of the same sign
- within one CD, equated can be only changes of the same signal
- events of two different CDs can be equated if they relate to the same signal.

2. A disconnectable arc can not be initially active.

3. Disconnectable arcs are not permitted on inputs of an OR vertex.

4. Disconnectable arcs must be put in order (i.e. the CD must be well-formed):

- if an arc (a, b) is disconnectable, then a is a one-time event (an event occurring once)

- if changes $a \to c, b \to c$ occur, and a is a one-time event, then either b is a one-time event or the arc (a, c) is disconnectable

- if a change $a \vdash b$ occurs, and a is a one-time event, then b is a one-time event too.

2.1.2.4 Connection of the CDs and TDs

A basic achievement of Varshavsky [1.2] is the statement of a mutually simple relationship between the semi-modular Muller's TD and correct CD.

Any semi-modular connected TD may be corresponded by a correct CD satisfying the following conditions:

1. For any pair of events $a1$ and $a2$ of changing a signal A of the same sign $a1 \to a2$, always found is an event $a3$ of changing of the same signal A of the opposite sign such that $a1 \to a3 \to a2$ (correctness on switchings).

2. CDs contains no concurrent changes of any signals (nonautorecurrence).

Right is also the opposite statement that any CD with the listed properties is corresponded by a semi-modular TD. This statement enables using of available methods related to CDs for analysis on semi-modularity of switching processes and schemotechnique solutions.

2.1.2.5 Acyclic CDs

A next important property of the CDs necessary for analysis of schemotechnique solutions is conformity of a cyclic CD to its infinite deconvolution (acyclic CD). In [2.1] there are formulated sufficiently simple rules of building both acyclic CDs on cyclic ones (i.e. deconvolution) and cyclic CDs on acyclic ones (i.e. convolution). The structure of an acyclic CD is a submultitude of the structure of cyclic CDs and contains no initially active and disconnectable arcs. Instead, a description of an initial signal state is used. The acyclic CD is built easily on a system of Boolean equations describing schemotechnique solutions and is convenient for subsequent analysis.

2.1.2.6 Analysis of precedence and concurrence

Namely for acyclic CDs, the terms of precedence and concurrence underlying correctness analysis have a strict sense. This determinism is paid for by obligatory analysis on achievability of all events of acyclic CDs limitedness and connectivity.

The central statement in the acyclic CD analysis enables for settlement of connection of the relationship of strong precedence and concurrence with the notion of the event achievability from an initial state:

- an event a precedes strongly an event b then and only then when both a and b are achievable, with b being not achievable without a

- events a and b are concurrent then and only then when a and b are achievable one without another.

The event achievability area is determined by the following properties of cyclic CDs:

- deconvolution periods begin to repeat not later than since $(n+1)^{st}$ period, where n is the number of CD events

- an arc (a, b) is limited then and only then when the event a of 1st deconvolution cycle is not concurrent with the event b of $(m+1)^{st}$ deconvolution cycle, where m is the number of CD signals

- a CD is connective then and only then when, for any event a, a multitude of achievable events of $(m+1)^{st}$ deconvolution cycle without events a is empty, where m is the number of CD signals.

This means that a full analysis of a cyclic CD can be accomplished on not more than $(n+1)^{st}$ deconvolution period of a corresponding acyclic CD. The described CD properties consist in a theoretical base of analysis on semi-modularity for a system of Boolean equations with regard to an initial state.

2.1.2.7 Algorithms of checking on semi-modularity of a system of Boolean equations with regard to an initial state

The algorithm below represents only a general solution scheme for the task of analysis of self-synchronous circuits by means of CDs.

1. Building, on a system of Boolean equations and a given initial state, of an acyclic CD containing $(n+1)$ deconvolution period of a cyclic CD.

2. Analysis of achievability of all events from the initial state (not achievable event and arcs incedent to them are excluded from consideration).

3. Analysis of relationships of strong precedence and concurrence of events (including the checking of the properties of nonautoconcurrence and correctness on switchings).

4. Determination of CD limitedness degree and analysis of CD connectivity.

Complexity of the described algorithm of analysis on semi-modularity of a system of Boolean equations with regard to an initial state for a closed self-synchronous system does not exceed $O(n^4)$ where n is the number of CD events.

For practical implementations, it is required to detail some specific questions and investigate a number of supplementary properties of constructed acyclic CDs that is capable to influence perceptibly computation overheads.

2.1.2.8   Practical approbation of the algorithm

In spite of the available theoretical base, no full-scale software realization of this algorithm is known, and we plan to make efforts for its implementation.

At present, there exists the program TRASPEC [2.1] for analysis of distributive circuits, but it covers a limited subclass of self-synchronous circuits, not to mention that its available version has too many errors.

Difficulties arising at realization of the analysis concept in respect to large self-synchronous circuits are connected with high (though polynomial) complexity of the described algorithm. The basic solution directions are following:

- decomposition of CDs in fragments prior to analysis

- specification of algorithms for fast computations of typical tasks on separated fragments.

2.2   Analysis algorithms applied in the system BTRAN

The subsection 2.1.1 is dedicated to algorithms of analysis of S^3 circuits on semi-modularity based on the Muller's TDs. Those algorithms consist in a theoretical base of analysis on semi-modularity as relying immediately upon its definition and involving into analysis all possible work states of a circuit. This feature makes necessary availability of a fundamental program of analysis on semi-modularity in any CAD system for S^3 VLSIs. It may be used as a tool of analysis and checking of complicated circuits exceeding the bounds of limitations of the synthesis methods.

Immediately as analysis means, such a program is effective for small circuits (to 80 variables) and low parallelism degrees (to 10). For large circuits, the exponential algorithm complexity makes the existing programs (e.g. TRANAL) inefficient. Meanwhile, such extreme characteristics of investigated S^3 circuit fragments are encountered frequently enough. Therefore, we consider the expansion of the number of variables to 250 and the parallelism degree to 14 in the system BTRAN as an essential contribution to increase of the analysis system efficiency.

In order to check self-synchronousity of a circuit via semi-modularity, it is needed:

• to determine a work multitude of TD states for a circuit being checked (this is the multitude achievable from an initial state)

• to check conditions of semi-modularity violations on all states of the work multitude.

The 1st task is most labor consuming. Its complexity can reach 2^n where n is the dimension of a system of logic equations.

Building of a work multitude is performed on the layer-by-layer basis. States included in $(i+1)^{st}$ layer differ from corresponding states of ith layer only by a value of a single variable, while building of a next layer from a previous one is performed by correct switchings of variables and formation of their new values for the next state layer. Some of states of a next layer are repeated multiple times but included in the layer only once, i.e. any layer does not contain repeated states.

The fact of termination of examination of work states is determined by matching or consuming state layers with different indices. In a general case, not each layer is multiple, and algorithm can diverge. Its convergence is provided by a technique of coercive selection of a *check-layer* if the match did not occur within a predefined number of layers.

Any semi-modularity violation is fixed immediately during building of a next layer as an incorrect rejection of switching of a state variable. The building of a next layer is terminated with a special message reporting conditions of the violation.

This method enables requirements of memory for storing states to be lowered by an order of magnitude, with algorithm execution speed being correspondingly raised, due to cutting down the number of comparisons.

Basic time expenses of the algorithm are determined by permanent comparisons of newly formed states with already existing in a layer under construction. It is evident that the building time for a state layer has, in this case, the quadratic dependence on layer measurements, i.e. the number of states.

A real way to shorten essentially the computation time is division of a layer states array in a number of blocks and distribution of the states among those blocks in compliance with some selected characteristics. Then the search for a specific state could be performed within a definite states block rather than the whole layer array. For organization of such a layer building process, it was proposed:

- to involve a virtual mode of operation with all states arrays but one being built, that leaves about 80 % of memory to the latter

- to store layer states in a hashed form for abrupt reducing the search time within the layer.

Division effectiveness depends on method of choosing the hash-function determining the region of search for and filling in the states layer. The function must meet the following requirements:

- load evenness for all regions allocated for a states layer

- short calculation time of the function itself (i.e. finally, of the region address for a state being searched for).

The following algorithm was designed for calculation of the hash-function:

- under layer building, accumulated is statistics on switchings of variables and performed is a test analysis of several states layers

- selected are variables that are switched most frequently and alternate 0 and 1 states most evenly.

The first condition guarantees subsequent changes of the selected variables during circuit analysis while the second provides the even division of a whole layer in separate regions. The hash-function itself represents a binary address of the region of searching for and forming up a states layer.

During building of states layers, the hash-function parameters are updated permanently on results of division of preceding layers that allows more even and optimal using of memory.

Depending on a specific set of logic functions and his desire, the system BTRAN enables the user to control indirectly the memory usage and analysis time.

2.3 Algorithms of the subsystem TRASYN

The subsystem TRASYN is intended for automation of design process of self-synchronous circuits and enables for synthesis of S^3 circuits on closed cyclic CDs.

The system input is a description of switchings of an S^3 circuit being designed. The system output is a set of Boolean equations in the antitonous basis supplemented by initial conditions of the S^3 circuit implementation.

The functional portion of the system comprises two basic blocks:

- of analysis on semi-modularity of a source description in the CD language

- of synthesis of an antitonous schemotechnique solution of the S^3 circuit on base of CD deconvolution of a semi-modular process.

2.3.1   The block of analysis of source CD descriptions

The block of analysis on semi-modularity of source CD process descriptions is implemented on base of the algorithm described in the subsection 2.1.2. However, its input is a description of switching processes rather than of a circuit, and building of an acyclic CD is performed on the deconvolution rules for source cyclic CDs. In the rest, the analysis process conforms to the algorithm in the subsection 2.1.2 (2, 3, 4).

2.3.2   The block of synthesis

The block of synthesis was conceived as a tool of automatic generation of S^3 circuits (that resembles a conventional silicon compiler). However, the authors of the subsystem TRASYN did not manage to find a uniform, universal, and efficient algorithm of "translation" of a semi-modular CD into an S^3 circuit (and there is no evidence that this is principally feasible). The developer's efforts were concentrated therefore on creation of a set of algorithms efficient for solution of specific tasks.

Note.   Emphasis should be made on an important feature of all proposed synthesis methods: their strict implementation of a source CD. This means any implemented S^3 circuit to function in strict compliance with the event sequence defined by the source CD. This feature distinguishes advantageously the subsystem from other analogous programs that usually permit some "deviations" in the event sequences.

The S^3 circuits synthesis assumes the following stages:

1. Redetermination and correction of a source CD to obtain the solution in the antitonous basis.

2. CD generation.

3. Forming the truth tables on CD states to derive Boolean function of elements.

4. Minimization and search for an optimal solution of the S^3 circuit.

The stages 3 and 4 are described in detail in the available literature on self-synchronous circuits and do not require essential "interventions" of the circuit designer. The main attention is paid therefore to methods of redetrmination and correction of a semi-modular CD for generation of final S^3 circuits in the antitonous basis. Namely these methods determine amounts of intellectual expenditures and qualitative characteristics of the solution accepted.

The proposed methods are conditionally divided in two basic classes:

● methods of "direct translation" ensuring a schemotechnique solution conforming to a given CD

● interactive methods of S^3 circuits synthesis aimed to obtain best schemotechnique solutions.

The practice of work with the subsystem confirms necessity to use methods of both classes.

2.3.2.1 Methods of direct translation

The task of direct translation of a semi-modular process CD to a schemotechnique solution supposes searching for an optimal variant of a circuit in a limited antitonous basis. Theoretically, the task may have no solutions because of the general character of CDs. Therefore, the search for an acceptable circuit solution is performed for an "equivalent" process extended by adding of extra variables. Just choosing a specific way to add extra variables, with the event equivalence being preserved, defines the method of direct translation.

The following direct translation methods are known:

● RS-implementation

● two-phase RS-implementation

● implementation on distributors.

2.3.2.1.1   RS-implementation

The concept of the RS-implementation method consists in a paraphase representation of each circuit variable on base of synthesis with RS flip-flops. It assumes doubling of process variables and additional restrictions on the kinds of both R- and S-functions:

● orthogonalization of R and S

● implementation of R and S in the reduced disjunctive normal form (DNF).

Actually, the method is used for small circuits as it requires building of a TD, and its complexity grows exponentially versus the number of variables. In the subsystem TRASYN, it was proposed an original solution of this method related to building of a TD for a source not modified process. It allows the program to preserve the source (not doubled) number of variables at TD building and to accelerate the synthesis procedure. Nevertheless, complexity of the method remains exponential in respect to the number of variables, while the obtained solutions are, as a rule, essentially redundant.

2.3.2.1.2   Two-phase RS-implementation

The method of two-phase RS-implementation applies the principle of separation of switching phases of a circuit in a *work phase* and a *reset phase*, for significant simplification of the R- and S-functions. The following artificial technique is used at its implementation:

- 1st deconvolution period of an acyclic CD is considered as the work phase while 2nd as the reset one

- a value of a new signal is assigned to any event within two first periods of a cyclic CD.

In a thus modified CD simulating a circuit block, signal changes within 1st cycle are considered as positive (the work phase) while within 2nd cycle — negative (reset phase). Source signals can be restored (a semantic portion of a circuit) by adding, modulo 2, of modified CD signals belonging to corresponding source signals (in compliance with the CD modification algorithm).

The obtained schemotechnique solution represents an RS-implementation of the simulating and semantic portions, and the control circuit ordering the switching of basic control variables (i.e. signals of the source CD).

The proposed method of logical translation is universal and implemented in TRASYN for processes described by signal graphs. Implementation complexity of this algorithm is consequent upon implementation complexity of the circuit simulating portion and is linear versus the number of events in the modified CD.

2.3.2.1.3   Implementation on distributors

The main concept of the implementation method on distributors (e.g. David's cells) consists in localization of the two-phase discipline at the level of each event of a source CD. By this, a

distributor cell simulating an event is excited after input information is received and relaxes after output information is delivered to subsequent distributor cells (a direct analog to "neuron networks").

The general structure of the schemotechnique solution remains analogous to the two-phase RS-implementation concerning the simulating, semantic, and control portions. However, the circuit control portion becomes implemented within the structure of the distributor cell while the semantic portion is simplified since a distributor cell switching simulates a source CD event and can be used immediately for restoring the signal state. Like the previous method, this algorithm is implemented in TRASYN for semi-modular processes described by signal graphs. The algorithm complexity is determined by the simulating portion of the schemotechnique solution.

2.3.2.2 Interactive methods of synthesis

The interactive methods of synthesis of S^3 circuits are based on the CD decomposition principle. Here are used the properties of composition semi-modularity preservation and rules of CD assembling from CD fragments (projections).

High qualification of S^3 circuits designers is necessary for correct CD decomposition of a whole circuit. The decomposition criteria are:

- completeness of covering source CDs

- projection "undiscrepancy"

- minimum complexity of each projection.

Building of an S^3 circuit for each CD projection may be accomplished by one of direct translation methods. In many cases, the solution turns out significantly more compact and efficient than resulted from the direct translation of the whole source CD.

Each of the synthesis methods has its advantages and disadvantages, its application area where it is most efficacious. The final decision is accepted by the designer.

Some comparative estimations of the described algorithms [2.1] are gathered in the following table.

Methods Features	RS-implementation	Two-phase RS-implementation	Implementation on distributors	Implementation by decomposition
The number of inputs n — the number of signals m — the number of events q — vertex degree on input	$2 \times n \times L$ L — complexity of R- and S-functions (L is large)	$4 \times m \times q$ (at m >> n)	$3 \times m \times q$ (at m >> n)	Depends on the kind of projection
Event simulation time, τ — gate delay	2τ	5τ	8τ	4τ
Possibility of implementation in limited bases	poor	good	good	good

2.4   Conclusions

The CD-based extended and modified S^3 VLSI CAD system FORCAGE 3.0 allows development of S^3 circuits. It is a reliable base for further algorithmic developments.

Design of complicated circuits can be implemented by decomposition in simpler fragments with subsequent using of CAD systems FORCAGE 3.0 or RONIS, with FORCAGE 3.0 performing analysis and synthesis at the logic level and RONIS accomplishing synthesis of S^3 fragments in transistors.

The subsystem BTRAN can be used for "manual perfection". Combination of these software means enables efficient design of S^3 circuits at present that will be demonstrated in following chapters of this report.

The theoretical materials stated in this chapter promise prospectiveness of the theoretical and algorithmic investigations in the area of S^3 circuits analysis and synthesis on base of CDs, with creation of algorithms and software for CD-based analysis of schemotechnique solutions on semi-modularity being regarded as most urgent. Solution of this task removes the "dimension damnation" from analysis on semi-modularity as having the polynomial character of computation overheads versus circuit dimensions. Supplemented by efficient diagnostics, such a software is able to become a main tool of qualified S^3 circuits designers.

Though a powerful and efficient solution regarding development of S^3 VLSI CAD systems is seen in the domain of analysis, the available algorithms are obviously neither sufficient nor efficient in the domain of synthesis. This makes often inexperienced designers to reject application of S^3 circuits. Therefore, issues of creation of new algorithms for CD-based synthesis of S^3 circuits remain rather actual.

Three basic research directions are distinguished explicitly in the domain of CD-based synthesis:

- creation of synthesis tools for quasi-self-synchronous circuits (widely applied now due to lower hardware implementation overheads)

- development of new synthesis algorithms for S^3 circuits not involving Muller's TDs

- development of new efficient algorithms of "immersion" of semi-modular processes into the antitonous schemotechnique basis.

The sequence of works on creation of a powerful CAD system for S^3 VLSIs enabling a full-scale design of complicated S^3 circuits can be as follows:

1st stage   Creation of a CD-based program of S^3 circuits analysis on semi-modularity.

2nd stage   Development of analysis and synthesis tools for quasi-self-synchronous circuits.

3rd stage   Development of synthesis algorithms not using the TDs.

Development of new algorithms of schemotechnique implementation of semi-modular processes.

3   Program Subsystems VHDL - BTRAN and TRASIN

3.1   Functionality and characteristics of the subsystem BTRAN/VHDL

The subsystem BTRAN/VHDL is intended for checking of S^3 circuits on self-synchronousity (semimodularity) and evaluation of some their behavior characteristics. The main facility of the subsystem, as compared with the subsystem TRANAL, is a fast analysis procedure derivative from involvement of all accessible memory and application of hash algorithms for access to data during generation of a stratum. Comparing with the subsystem TRANAL, $20 \div 50$-fold increase of the number of checked states and $20 \div 50$-fold decrease of circuit analysis time are obtained.

The subsystem consists of a basic part performing analysis of logic functions on self-synchronousity — the BTRAN itself — and a supplementary program converting descriptions of logic circuits from a dedicated BTRAN input language into a subset of the VHDL language.

A circuit to be checked is represented by a system of Boolean equations (by the Muller's model conforming to the rules of the input system language) and a given initial state. Circuit elements are simulated by model equations, with element output being written in their left part and its inputs — in the right part. Analysis results in either confirming semimodularity of the circuit, for the given initial state, or detecting violations of semimodularity with indication of ambiguous states and conflicting variables, or revealing a deadlock state.

In the normal mode of analysis on semimodularity, the circuit is checked with regard to the given initial state. For another initial state, the circuit description should be corrected and analyzed again. In the browsing mode, the circuit inputs to be browsed should be explicitly listed.

During and after analysis, the user is reported about most essential characteristics of the analyzed circuit: number of checked states, parallelism degree, etc. At a syntax unconformity or violation of restrictions on circuit parameters, analysis is terminated with appropriate error reports.

The user is allowed to control the analysis process by presetting definite parameters. Relevant information is displayed during analysis.

Restrictions imposed to circuit elements:

- maximal number of analyzed elements — 256;

- maximal number of inputs — 8;

- maximal fan-out — 32;

- maximal number of simultaneously excited elements — 32;

- maximal number of browsed inputs — 32;

- maximal element name length — 24 characters.

The BTRAN converter program provides an automated interface for transferring results of circuit designs to circuit simulation systems. The converter forms up an S^3 circuit model description in a subset of the VHDL language. The obtained description comprises an architectural body consisted of a process operator built from logic equations describing the circuit.

Aside from text conversion, the converter program removes, if necessary, elements simulating the «environment» from the source description (i.e. transforms closed-type circuits into the open type). If a source circuit is of the closed type and does not have neither inputs nor outputs, no corrections of variable types is needed at conversion to the VHDL description. If a source circuit is of the open type and has inputs and/or outputs connecting the circuit to its environment, types of circuit input and output variables set by default should be changed prior to forming up the VHDL description. Sometimes, it may occur desirable to «unclose» closed-type circuits by removing some equations from the source description. Such an «unclosing» can be executed by the converter as well.

The described converter version assumes assignment of types for external variables to be done in a user dialog. The dialog is based on a sequence of questions accompanied by helps and protected from incorrect user replies. Under correction, some partial analysis of the circuit connections is performed, warnings are issued on incorrect and ambiguous situations, some automatic corrections are made depending on circuit peculiarities.

The converter processes all language constructions proper to the subsystem BTRAN. The permissible number of analyzed elements is up to 500.

3.2 Functionality and characteristics of the subsystem TRASYN

The subsystem TRASYN is intended for obtaining logic circuits on a given behavior description of the circuit operation process. It provides:

– analysis the input behavior description given in the CD language;

– synthesis, if possible, Boolean equations of logic elements of the circuit implementing the input CD.

A CD represents an event model, in which a switching of a circuit element corresponds to an event. The CD is considered being correct when the process of S^3 circuit elements switching can be directly connected to it. The following properties of CD correctness are analyzed in the subsystem TRASYN:

1. Correctness on switchings —signs of changes of the same signal should alternate, or, in the opposite case, a sequence of changes cannot be implemented by the circuit.

2. Nonautoconcurrentness of signals — there should not be simultaneous changes of signals with the same name in a CD.

3. CD limitation —limitless of any arc carries on to necessity of use of unlimited memory for storing the number of markers that does not allow CD implementation by the circuit with a finite number of elements.

A result of an analysis stage is either confirming of the input description correctness or localization and classification of incorrectness points.

At a synthesis stage, restoring of global states of a transition diagram from a changes diagram is performed. A condition of existence of a system of Boolean equations is absence of contradictory states on circuit elements inputs. In the case of CD inconsistency, the subsystem displays contradictory states codes and information on the implementation trajectories resulting in a contradictory situation.

At a minimization stage, the subsystem creates Boolean functions of circuit elements in the DNF, and the control of the synthesized function type is stipulated.

The main characteristics of the subsystem TRASYN are:

– synthesis of any subset of circuit variables;

– possibility of synthesis, on the partial definitions, of circuits as contradictory CDs;

– bilingual interface of dialogues (Russian and English);

— possibility of construction of elements functions from CDs on basis of information on functional dependencies.

Limitations accepted in the subsystem TRASYN are:

— the number of circuit variables at analysis ≤ 255;

— the number of vertices in a diagram ≤ 512;

— the number of active vertices< 127;

— the number of arcs in a graph ≤ 32000;

— the number of vertices in a linear fragment < 128;

— the number of includings of a variable name < 128;

— the number of variables in a circuit at synthesis without indication of dependencies between variables ≤ 16;

— the same at synthesis with indication of dependencies ≤ 255;

— branchings coefficients on element inputs and outputs ≤ 64;

— number of switchings of one variable within a running cycle < 64;

— number of variables in a circuit at minimization ≤ 64 (≤ 32 in the case of the RS-implementation).

4 THE S³ CIRCUIT DESIGN CONCEPT

In order to determine the position of the proposed approach to the S³ circuit design among many existing, it is reasonable to analyze its particularities in comparison with other approaches.

The main specificity of the proposed approach consists in accepting a *strict self-synchronousity* definition on base of the Muller's model.

Two groups of methods can be discriminated in conventional *not strict* approaches. In the 1st group, a circuit compilation synthesizes circuits with the interblock interactions of a *request-response* type. The blocks themselves are not *delay-insensitive*. Examples of this group are represented in [4.1, 4.2]. Methods of the 2nd group are based on the *conditional*, i.e. satisfied on some certain conditions, self-synchronousity and assume introducing into a circuit some internal timing or *pipe-line* operation, e.g. as in [4.3]. Such circuits are *delay-sensitive* and should be classified as *quasi-self-synchronous*.

The *non strict* approaches are "halved". Their advantage is that the synthesized circuits possess, within definite limitations, self-synchronousity properties at acceptable hardware expenses. Their disadvantage is self-synchronousity implementation incompleteness that prevents obtaining some important properties inherent to circuits meeting the *strict* self-synchronousity principles at levels of gates, blocks, and further, the reliability properties primarily.

The available design experience with respect to the specificity of the S³ circuits shows the existing general-purpose CAD tools to be inadequate at the lower levels of logic (gates), transistors, and topology. The general-purpose simulation tools can be used only for the final check of a designed circuit on an operation correctness.

The situation is consequent upon the following requirements that are to be met under design:

a) rather specific requirements of self-synchronousity on Muller must be provided for a set of logic functions describing a circuit;

b) S³ circuit implementability is defined by a set of basis functions describing a circuit;

c) an S³ circuit implemented at the transistor level, like any other circuit, has to possess a definite *schemotechnique quality*, i.e. must be balanced on delays, optimized on speed, number of transistors, chip area, etc.;

d) under topology design, on the routing stage, the self-synchronousity conditions must be provided also in branches.

Apart from these necessary requirements, other problems associated with enhancement of S^3 circuit quality should be minded under design: consideration of input breakdowns after branchings, increase of reliability of topological structures, etc.

The known existing CAD systems specially created for the S^3 circuits do not solve, and are not assigned to solve, all these tasks.

Methods and systems being developed by foreign authors provide only the request-response interblock interaction and/or quasi-self-synchronousity. Specifically, a special self-synchronous coding is not applied, and the conditions of strict self-synchronousity (semimodularity) are not satisfied.

At present, the only CAD system assigned to design strictly self-synchronous circuits is the system FORCAGE developed by the group of Dr. Varshavsky[1]. But even this system meets only the requirement (a) of the design requirements listed above. Derivative from this incompleteness, its application area is restricted to investigations of self-synchronous, on Muller, equation sets of not large dimensions.

Below, the approach to S^3 circuits design implemented in the CAD system FORCAGE will be observed in detail.

4.1 Design facilities in the CAD system FORCAGE (F-approach)

Merits of the design methods at the F-approach are the following.

For the first time, regular methods of analysis and synthesis at the functional logic level have been created for the self-synchronous (semimodular) circuits. The methods are universal in respect to ways of information coding, presence or absence of memory, set change discipline, and other initial characteristics. Basing on the developed methods, the pioneer software system of S^3 circuits analysis and synthesis FORCAGE was created.

However, this approach is not adequate namely for the schemotechnique design as not meeting satisfactorily none of the listed design requirements and having significant drawbacks.

Implementation of the analysis and synthesis algorithms at the F-approach is connected with transforms of global states that provokes an exponential dependence of system resources needed

---

[1] The CAD system FORCAGE is property of IPI RAN.

for analysis and synthesis versus the number of variables. Other methods and representations (reference diagrams, change diagrams, and others) are able to decrease the needed resources but the exponential character of the dependence still takes place. Claims of the authors of the F-approach regarding the polynomial complexity of algorithms are unconvincing as all methods are based eventually on analysis of state changes in a whole circuit or its part.

The synthesis methods use, for description of source data, change diagrams prepared by a user. In case of a weakly defined circuits, these descriptions are not complicated. But for near entirely defined circuits, the description becomes "sorting out" and, as a result, cumbersome and unimplementable. This regards specifically to combination circuits.

At the F-approach, a designer composing a source description, for subsequent synthesis, has to solve two tasks providing:

- necessary functionality

- process aperiodicity conditions (correctness conditions).

These conditions are exactly the conditions of self-synchronousity expressed in other terms. Therefore, both the source description, in changes, and the synthesized Muller's model are equivalent in the sense of self-synchronousity. The synthesis scheme does not add to a circuit any new quality: the source data that anyway have to be prepared by the designer must already possess self-synchronous properties. This is followed by "distributivity" limitation, i.e. exclusion of the "OR" operation from the source description. It is a grave limitation complicating the S^3 circuit design and not understandable to designers of conventional circuits.

Another drawback also consequent upon the aperiodicity conditions of a source description is the absence of circuit inputs and outputs both in source data for synthesis and in obtained solution. The designer gets constrained to add some artificial environment to the source description before synthesis and to remove corresponding extra elements and connections afterwards. All these are pure overheads.

A considerable disadvantage of the F-approach is that its source descriptions are not modular (non hierarchical). This interferes with embedding fragments designed earlier into later developments and confines design possibilities.

Regarding the S^3 circuit implementation basis at synthesis[2], the F-approach takes into account only requirements of element function monotonousity and limitations on the number of inputs and outputs. These requirements are insufficient since there exist another schemotechnique

---

[2] Here the *physical* basis, i.e. minimal elements desribed by one Boolean equation each, is implied rather than the *logical* basis, e.g. RS-implementation type.

limitations, at the transistor level, on the base elements.

Regarding performance of synthesized circuits, the F-approach neither raises nor, accordingly, solves this task. A standard DNF minimization can not be considered as sufficient for circuit optimization. In fact, the applied criterion of the number of variable re-entering influences the number of transistors under implementation only indirectly. The circuit performance and other characteristics are of no concern at all.

Deriving from two latter particularities of the F-approach, it is possible to affirm that an actual object and result of the design is a system of Boolean equations rather than a circuit itself. Accordingly, the design level at the F-approach could be defined as a *functional-logical*.

From the viewpoint of the user interface, the CAD system FORCAGE has graphical input and text input and output descriptions. Languages of the text interfaces, both input and output, differ inconveniently from standard ones (e.g. VHDL) and do not meet widely accepted notation rules for the logic formulae.

4.2   Design facilities in the CAD system RONIS (R-approach)

As opposed to the preeminently theoretical and investigation-oriented F-approach, a more practical R-approach enables a user to design a conventional description — automaton logic equations — with no respect to properties of self-synchronousity or aperiodicity. The functioning has to be described in a static regime with no respect to the circuit timing. We will name such a description *monophase functional prototype* (MFP). At the MFP design, conventional existing CAD systems may be used for synthesis, simulation, etc. As usual, an MFP is represented in a form of macroelements (blocks). In parallel, the system RONIS synthesizes automatically, also blockwise, S^3 macroelements carrying out thus an S^3 project.

Description in the form of event (or time) representations, for example as change diagrams is more convenient sometimes than description in the form of logic functions. Therefore, it is supposed in the future to develop a special subsystem for conversion of monophase event or time descriptions into logical ones. In contrast to the F-approach, no requirements of aperiodicity (self-synchronousity) are imposed to those source descriptions, this may significantly simplify its preparation. Synthesis of S^3 circuits can be performed further on the obtained monophase logical descriptions.

Synthesis by the system RONIS will result in an S^3 device with paraphase information coding and dual phase (with a *spacer*) functioning discipline. Such choice is explained by the fact that, due to available experience and estimates, other modifications of devices have very restricted application areas.

A functional representation of source data and synthesis results underlies the synthesis methods at the R-approach. Synthesis is carried out by portions — "S^3 units" of moderate size, about 10 gates on the average. The exponential dependence of algorithm complexity takes place only within such units. The number of the S^3 units in a circuit is practically not limited as their composition increases the algorithm complexity only linearly.

The R-approach has a number of advantages comparing with the F-approach.

1) The designer solves customary tasks of functional design facilitated by absence of any timings.

2) The habitual VHDL language is used for describing a prototype.

3) Possibility of embedding circuits designed earlier supports hierarchical design mode and removes limitations on the project size and complexity.

4) Physically, the approach is based on CMOS and BiCMOS S^3 base elements that meets both the requirements of the Muller's model and the design requirements listed above (schemotechnique restrictions). These elements will be supported by the CAD tools at the transistor and microtopology levels.

5) On all stages of synthesis the optimization is performed on criteria of performance and/or number of transistors.

6) Design at the R-approach encompasses the following levels:

- functional-logical
- gate
- transistor

7) The R-approach applies transforms of logic functions. This more compact representation of functional dependencies enables the synthesis system to process a circuit $1 \div 2$ orders of magnitude greater as compared with the F-approach.

4.3   Structure the CAD system RONIS and design tasks

The basic task of the system RONIS is the automated design of S^3 VLSIs at several levels.

1) At the functional-logical level — design of MFPs and their preparation to synthesis.

2) At the gate level — synthesis of S^3 circuits, in base functions (gates).

3) At the transistor level — generation of electrical circuits of base elements (gates), in transistors.

Besides, supplementary tasks comprise support of system data bases, project libraries, and libraries of base elements.

The structure of the system RONIS with its major information links is shown in *Fig. 4.1*.

The CAD system consists of four parts:

1. **Subsystem of S^3 and quasi SS circuits analysis — ASPECT**:

- analysis of S^3 and quasi SS circuits with complexity up to some hundreds gates

2. **Subsystem of functional-logic design of S^3 circuits — MIRAGE:**

- preparation of functional non-selfsynchronous prototypes;

- synthesis of S^3 circuits (combinational, circuits with memory, indicators, pipelines, etc.) in the base of SS-functions;

- support of project libraries (prototype macroelement descriptions, synthesized macroelements, basic functions).

3. **Subsystem of prototype synthesis — SOLON**:

- synthesis of functional non-SS-prototypes on event descriptions;

- support of project libraries for synthesis.

4. **Subsystem of S^3 circuits design on gate and transistor levels — VAVIL**:

- analysis of S^3 circuits on implementation correctness

- synthesis of the base library elements in transistors

- support of the system base elements library

- calculation and evaluation of S^3 circuits parameters (number of transistors, delays, etc.)
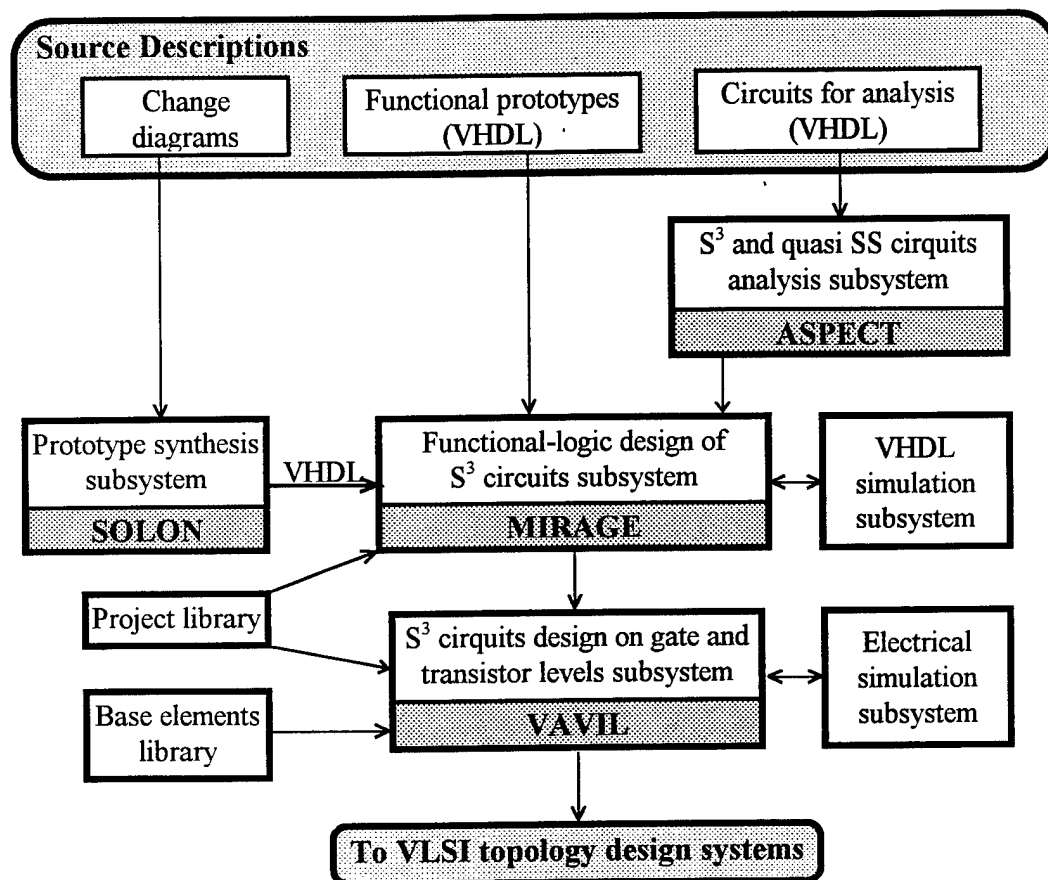
- interface to VLSI topology design systems.

Fig. 4.1   **Structure Of the S³ Circuit CAD System RONIS**

The design with the system RONIS permits to involve the existing systems for VHDL simulation and for electrical simulation (of PSPICE type).

The system ASPECT is advisable to be involved for synthesis of small S³ circuits (cells, macroelements) in special cases, for example, when a sophisticated circuit is designed "manually" because a standard synthesis procedure with the subsystem MIRAGE can result in a non-optimal solution.

4.4 Specificity of S^3 circuits topology implementation

VLSI CAD systems available at present incorporate flexible and powerful means of topology design. In many cases, those can be used for S^3 circuits topology design. However, design outcomes require a special check and possible correction. Application of standard means results usually in both extra design time and extra silicon area.

4.4.1 Equichronous zone of an S^3 element

The Muller's hypothesis considers a switching delay of any active element being arbitrary but finite while a delay difference in output wires after a branching point — negligible. That is, delays in wires are reduced to the element output, and the output signal appears on inputs of subsequent active structures simultaneously. This rule is formulated in practice, in a restricted form, as limitation of the delay difference value in wires after a branching point by a switching delay of a gate element. This value is constant for a specific production technology and determines geometrical applicability area for the Muller's hypothesis, referred to also as *equichronous zone*.

Geometrical measurements of the equichronous zone are determined by R-, L-, C-parameters of connection routes and shrink rapidly along with diminution of design norms, with the delay nature changing gradually from aperiodic to wave one (LC parameters dominating). For modern technologies $0.15 \div 0.5$ µm, the size of an equichronous zone is hundreds microns that is comparable with lengths of some connections. Outside the equichronous zone where Muller's conditions are broken, the notion of self-synchronousity becomes senseless.

Solution methods:

1) Rerouting to reduce lengths of some critical connections. The method is not always efficient, since reduction of some connections is able to produce new critical ones.

2) Application of additional circuitries (output indication) leveling delays. This method solves the task, but it assumes extra hardware and increases circuit complexity.

3) Diminishing of measurements of self-synchronous units to the size of the equichronous zone (by additional division of S^3 units by separate S^3 blocks and subsequent reassembling). This method, like the previous one, augments the circuit complexity though ensures meeting the Muller's conditions.

Usually, the most efficient solutions combine several methods and depend on design ways and features of S^3 circuits.

4.4.2 Structure and schemotechnique peculiarities of topological implementation of base logic elements

Quality of VLSI topology design depends immediately on a priori information on structure and interconnections of topology blocks. The more information is accessible to the designer, the higher is intelligence of CAD tools, flexibility of their interfaces, and quality of chip topology. This relates directly to S^3 VLSIs topology design. Here, depending on the method, there are predesigned libraries of standard fragments and automatically generated topological structures of nonstandard units.

In the CAD system RONIS, synthesis of circuits and base elements is accomplished in that way that the obtained topology solutions do not have feedbacks. The libraries of standard elements comprise weak inverters which topological connection is implemented at assembling of a whole S^3 unit. The basic schemotechnique features of a base logic function are:

- monotonousity
- restriction to a single output
- limitation on the number of implicants
- limitation on the number of variables in an implicant.

Consequent upon such a structure of the base logic function are the following topology implementation features:

- any base topological structure is situated within an equichronous zone

- CMOS implementation has an interconnections routing graph near to planar (the planarity is broken only at routing of input circuitries).

Since troubles with delay differences appear only in input circuitries, the internal and external circuitries can be routed in different layers. In the CMOS technology with planar internal interconnections of active structures, those can be implemented in a diffusion layer without involving metallization. This enables a silicon compiler to produce a very compact and efficient topology.

4.5   Conclusions

1) Design of S^3 circuits has a number of specific features caused by the requirements of the Muller's model. That does not allow using of traditional VLSI CAD systems. Some special design tools are necessary for the S^3 circuits design.

2) All known special design systems for such a type of circuits do not provide their complete independentness of delays at the gate level (the quasi-self-synchronous approach). As a result, the circuits being synthesized do not possess such a high level of tolerancy that is intrinsic for S^3 circuits.

3) The only existing experimental S^3 circuits CAD system — FORCAGE — has a number of essential drawbacks that do not permit its using for design of circuits with complexity of more than 100 gates.

4) The proposed S^3 circuits CAD system RONIS is free of most of FORCAGE's disadvantages. The CAD system RONIS is being developed as a system that solves tasks specific for S^3 circuits design only, it is compatible with general purpose VLSI CAD systems via special embedded interfaces.

he system RONIS will enable design of circuits of practically unlimited complexity at the function-logic, gate, and transistor levels. The topological level will be provided by the interface to any existing general purpose CAD systems.

A special program in the system RONIS which can automatically generate topology of library basic cells (LBC), with respect to their peculiarities, allows for increasing of efficacy of existent general purpose topology CAD systems for S^3 circuits (more dense topology of S^3 circuits can be achieved).

The system RONIS possesses an ability to support also design of quasy-self-synchronous circuits in necessary cases.

5 The Self-synchronization as a Natural Base of Real-time Computers

5.1   Introduction

A project on Highly Reliable Emergency Computer (HREC) — a perspective computer for real-time systems — is being carried out about 2 years at IPI RAN. A main objective of the project was to develop a balanced system of architectural principles enabling creation of HRECs for automated control of particularly crucial (critical) objects and processes. The HREC architecture (and, accordingly, derivative computers) are to feature: controllable performance, dependability, self-synchronousity, recurrent nature, parallelism, and ability to redevelopment. Implementation of these features seems to provide efficient (high-quality, accident-free, long-term) operation of an HREC as a kernel of crucial control systems.

1st stage of the HREC project had finished at the moment of conclusion of the contract with EOARD (August, 1996). The main research direction on the 1st stage were:

a) selecting and developing a schemotechnique meeting implementation requirements of not synchronous WSI/CMOS systems;

b) searching for an architectural organization for the HREC ensuring reliable, maximum fast (due to parallelism) real-time computations relied upon the selected schemotechnique;

c) determining a structure and functions of design support tools for HREC WSI/CMOS circuits.

On the 1st stage:

— developed is the S^3 schemotechnique, to which design and application features this report is dedicated to a significant extent (1.a);

— proposed are organization principles of a recurrent computer architecture oriented at provision of steadiness of fast parallel real-time computations in HREC application domains (1.b);

— launched is development of a commercial CAD system RONIS of self-synchronous CMOS ICs for the HREC of various integration levels (from base cells to the WSI level), on which the 2nd part of this report is concentrated (1.c).

Below, some obtained at IPI RAN results regarding the qualitatively new HREC architecture are described briefly.

Note.   It should be emphasized that delivering of materials represented in this section has not been planned by the contract with EOARD. Those are added solely to complete representation about significance of the self-synchronization principles not only for hardware design at the element base level but also for a computer architecture as a whole. Universality of the self-synchronization principle enables its application at all organization levels of a computer system resulted in both a natural development and uniform operation.

5.2   Possible HREC application areas

Provision of fast highly reliable computations due to architectural orientation at parallelism was a central point of investigations on the direction (1.b). The reason was that, among many requirements to real-time computers, most rapidly growing are requirements of computational power, with reliability requirements being preserved stable high.

Most of requirements to computer characteristics are derivative from application areas. In our case, the following potential application domains were considered as possible for HREC systems:

— control of plant technological processes and expensive aggregates;

— support of optimal and safe functions of mobile objects (cars, vessels, planes, space apparatus, robots, and others);

— accident-free long-term control of local specific technological processes (e.g. mobile nuclear power plants on vehicles, vessels, satellites, etc.);

— monitoring of ecologically dangerous objects (gas and oil rigs and pump stations, chemical and medical reactors, etc.);

— control of weapons, troops, communications;

— long-term on-location investigations, tests, and experiments.

Objects of the listed areas have much in common. They require to apply high-reliable check and control means, operate in real time and heavy exploitation conditions, do not permit even short control intermissions. Accordingly, computers for those objects must ensure a sufficient level of computation power throughout all mission (life) time, including emergency conditions.

5.3   Requirements to real-time HRECs

To be applied in the areas listed above, HRECs are to possess the following properties (in the order of lowering importance):

— utmost speed (for a guaranteed performance reserve in occasional emergency situations);

— safety, both internal and external (i.e. "don't harm to neither yourself nor another");

— vitality throughout all mission (life) time;

— lifetime of the WSI system electronic portion not less than lifetime of a controlled technological equipment;

— "green tincture" (meeting requirements of the ecology standard *EPA Energy Star*);

— ability to redevelopment (expansion of performance range not less than 3 orders of magnitude versus a minimal configuration);

— fast economical recompensation.

Performance for the crucial real-time computers is a most important quality. Though it is put above the reliability, the latter (combining safety, fault tolerance, and vitality) may be considered also as a property contributing to increase of an integral (throughout all lifetime) performance.

Implementation of the listed most important properties allows the HREC to acquire a promising quality of dependability (in the sense of [5.1]) characterizing its ability to be applied in control systems for crucial object and processes.

All available on-line control computer systems possess only some of the listed properties that makes obvious the necessity to improve control computer architectures for crucial applications.

5.4   Investigations at IPI RAN in the area of perspective real-time computers

Works on high-performance and high-reliability computers are in progress all over the world. In this respect, indicative is the project URES ("Ultrareliable Electronic Systems", 1990-2000) of US Aerospace Industries Association Systems. The main goal of the project is to create, to beginning of XXI century, real-time computers able to function faultlessly during all their lifetime [5.2]. Both projects, URES and HREC, have the same final aim but differ in methods of its attainment.

The fault tolerant computers available on the market confirmed in practice the promised high reliability but for excessively high cost. Practically all systems designed for the real-time control use the same way of reliability enhancement — reservation (doubling, doubling by pairs, majorization, etc.). However, any reservation bears inherent difficulties related to checking of circuits controlling the checking, reservation of circuits controlling the reservation, etc. The systems, both reserving and reserved, are built in a conventional synchronous schemotechnique, with about 80 % failures being consequent upon signal deskews (especially in respect of clock and arbitration signals).

The IPI RAN's investigations concerned particularly specificity of crucial computer application areas and "weak points" of the known parallel approaches: Dataflow and Reduction Machines, Controlflow computers (CISC, RISC, and VLIW), Vector processors, and others. As a result, it were formulated requirements to a computing system and principles of its construction enabling for improvement of the computing stability in real-time applications. *Fig. 5.1* shows principal differences of some compared architectures; *fig. 5.2* clarifies distinctions in instruction execution for them.
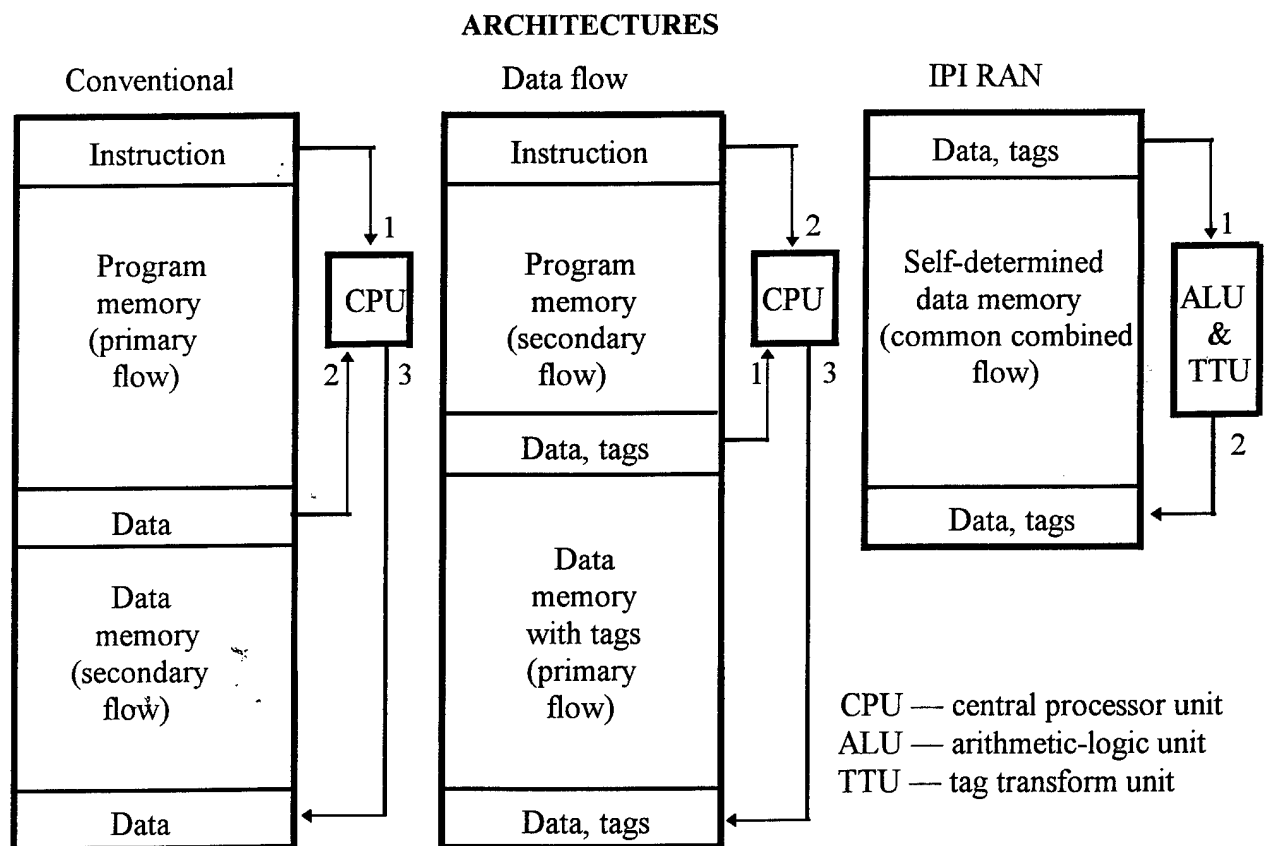
ARCHITECTURES

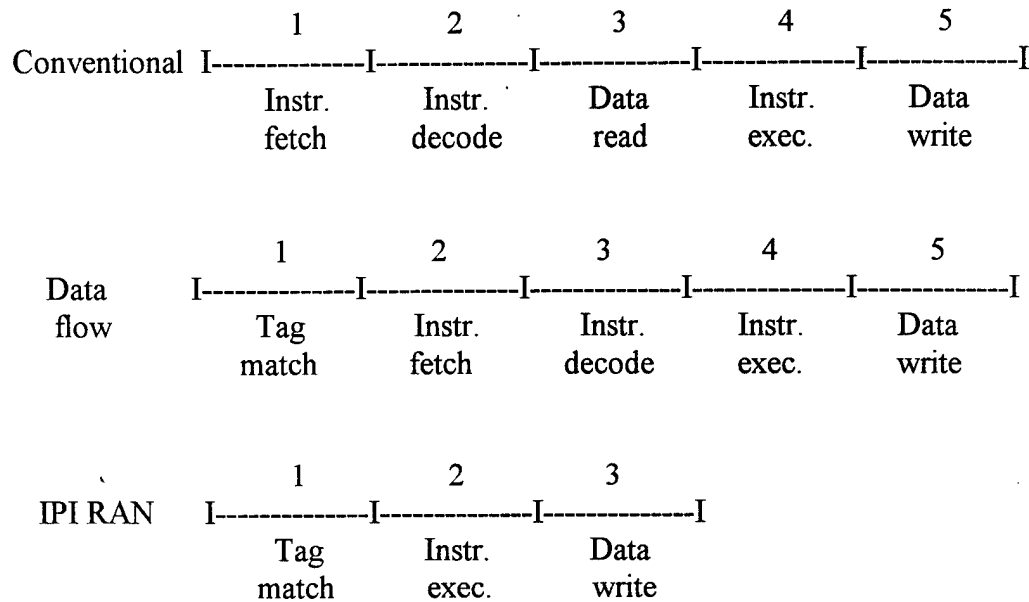*Fig. 5.1.* **Principal differences of the compared architectures**

ARCHITECTURES

```
                        1              2              3              4              5
      Conventional  I-------------I-------------I-------------I-------------I-------------I
                        Instr.         Instr.         Data           Instr.         Data
                        fetch          decode         read           exec.          write


                        1              2              3              4              5
      Data           I-------------I-------------I-------------I-------------I-------------I
      flow             Tag            Instr.         Instr.         Instr.         Data
                       match          fetch          decode         exec.          write


                        1              2              3
      IPI RAN        I-------------I-------------I-------------I
                       Tag            Instr.         Data
                       match          exec.          write
```

Fig. 5.2. Instruction execution in the compared architectures

The key features of the newly proposed architecture are: event orientation, self-synchronousity, self-determined data structures, and computation recurrence. All these together permits us to define it as a nonconventional architecture differing essentially from all known types of architectures.

5.5   Features of the IPI RAN's architecture

Not going here into details of the IPI RAN's architecture, relevant is to list basic principles distinguishing it from other known computer architectures (DFM, RM, CISC, RISC, VLIV, and others).

The proposed architecture is distinguished by:

— recurrence (recurrent algorithm coding, recurrent internal machine language, recurrent self-evolvement of the computation process);

— a single combined flow of machine words (with no division in instructions and data);

— computation process control based on self-determined data;

— self-synchronous organization of interactions.

The listed principles, mutually complementary, consist in a conceptual base of the IPI RAN's architecture.

Recurrence is a backbone of the HREC architecture. It is tightly connected with the self-synchronization. The recurrent architecture is intrinsically self-synchronous as the computation process is synchronized by data readiness (like in DFMs) according to recurrent self-determination of data. This allows using of all advantages of self-synchronous interaction (see Chapter 2 of Part 1 of this report) and makes natural the development of computation processes and system parallelism, due to the following:

— recurrent continuity of the computation process free of external constraining interventions;

— parallelism of development of the computation process as a multitude of concurrent and independent events;

— interactions asynchronousity free of limitations on events duration;

— self-coordination of interactions at local levels, without any imposed rigid timing.

5.6   Advantages of recurrent architectures

The recurrent architecture organization, that implies a definite functional and applied orientation, generates parallelism in its most efficient dynamic form and provides efficacious processing of both scalar and vector data. Simultaneously, performance increases consequently upon elimination of instruction fetch and decoding, overheads are reduced, probability of collisions diminishes, and support of modern programming technologies is facilitated.

The recurrent architecture introduces new facilities that enable the HRECs:

— to become a highly parallel event-oriented (self-synchronous) system supporting the dynamic parallelism at lowered overheads;

— to cut down hardware expenses for control of executive elements resulting from a compressed program representation form;

— to save time on moving, loading, and storing programs and data as combined compressed items;

— to enhance protectability of information and algorithms owing to recurrent encoding.

5.7   Factors stimulating transition to the self-synchronization

The «self-synchronization» is a natural way of distributed control of parallel processes, both static and dynamic. Moreover, it is a universal principle of interactions encountered everywhere in natural phenomena and processes. This universality shows promise that application of the self-synchronization in the HRECs at all levels, from architecture to schemotechnique, would yield an extra positive effect if the HREC project should be accomplished uniformly (see Section 2.3 of Part 1 of this report).

Rejection of conventional synchronous and asynchronous principles and transition to the self-synchronous approach is an obliging step. The reasons to make it were described earlier. At the HREC concept level, necessity of transition to the self-synchronization was imposed by three factors:

— advancement of architecture organization to full support of parallel data processing;

— increase of hardware and software complexity, with reliability preserved;

— adequacy of the IC schemotechnique to the organization levels of the architecture and software.

5.7.1   Structural parallelism and self-synchronization

Researches on new architectural solutions are concentrated on parallel data processing at diverse hardware levels. Growing complexity of parallel systems and comprised devices requires implicitly some alternative approaches to their synchronization. Inadequacy of a «rigid» synchronization scheme to the parallel processing, concerning both hardware and software, stimulated interest to asynchronous interaction principles. This interest itself should be considered as a step ahead to solution of efficient coordination of numerous functional elements within a parallel system. However, there is still neither «rich experience» nor approved design techniques in the asynchronous VLSI developments. This fact may be regarded as one of facilitating stimuli for abrupt transition directly to the self-synchronization as a more cardinal and perspective solution.

The self-synchronization is, in fact, the most consistent and comprehensive way of implementation of general asynchronous interactions. At a self-synchronous interaction, a next phase of element operation is initiated internally and inherently by natural completion of transition processes associated with a previous operation phase in the element rather than by general clocks,

5.7.3   Element base, architecture and self-synchronization

An adequate schemotechnique support for complicated parallel asynchronous VLSI, GSI, and WSI systems is the S^3 schemotechnique. This statement follows from Part 1 of this report and is indirectly confirmed by the fact that more and more often used practically in all international scientific congresses and conferences is the term «event-controlled systems», that, at the schemotechnique level, is a synonym of the term «self-synchronization».

Concept of the self-synchronization is based on alternation of two operation phases. At the bottom (schemotechnique) level, phase changes occur on completion of transition processes that are specially indicated and may be regarded as «advancing events». «Facts of events» and «Waits for events» are alternated. This model is entirely applicable to the top (architecture) level. Thus, the theoretical and model bases of asynchronous systems and S^3 schemotechnique coincide «ideologically» that enables for a methodologically uniform system design-through and makes control of parallel asynchronous processes most efficient.

Thus, persistent at all system levels application of principles of process development control on base of self-evolved data and self-synchronous interactions must provide to an HREC not only integrity and uniformity but also a higher degree of system organization.

5.8   Development trends of real-time systems and self-synchronization

Despite evident advantages, the self-synchronousity continues to remain outside the sphere of interests of leading computer manufacturers. Modern computers are developed on the conventional synchronous base. Two main trends-stimulators should be emphasized in their development:

1) enhancement of performance and throughput;

2) increase of operation reliability.

The first trend promoted creation of various on performance and throughput nonredundant computers. Further enhancement of performance and throughput of theses systems is associated with processing parallelism, and the parallelism degree will grow persistently.

The second trend is aspiration for exploitation perfection. It promoted creation of a class of fault tolerant computers able to function permanently dozens and hundreds thousands of hours at

the expense of hardware redundancy, with the amount of reserve functional devices and systems implemented as VLSI circuits having been essentially enlarged.

Recent years, computer developments showed explicitly tendency of merging both trends to obtain qualitatively new properties. The principle of the global self-synchronization meets both trends and their integration being universal and treating performance and reliability problems as «two sides of one medal» and providing thus a successful their solution.

5.9   Economics and self-synchronization

Indeed, the HREC project has a definite scientific significance but its economical aspects are difficult to estimate, in particular because of absence of analogs. The economical effect will depend on many factors, with the achieved dependability being dominating.

The dependability of a system is understood as a property enabling a user to rely on unconditional rendering services that the system is intended for [5.1]. The dependability can be derived from:

— essential prolongation, in several times, of the lifetime of HREC electronics, under operation in an extended range of ambient exploitation conditions, due to insensibility of S^3 circuits to deterioration and parametric failures;

— larger amount of computations executed due to prolongation of the lifetime;

— smaller number of reserve HRECs and lower replacement expenses due to leveling lifetimes of the computer and technological equipment;

— less wastes due to elimination of breakdowns and downtimes through computer's faults.

Thus, the self-synchronization is not only a basic constituent of the dependability implementation but also can contribute to HREC recompensation.

5.10   Conclusions

1. The principle of self-synchronous interactions, in respect to real-time computers, is widely applicable, universal, and productive. Its realization enables real-time computer designers to overcome almost «automatically» many obstacles and difficulties.

2. A practical attempt to realize this principle in the design of an experimental real-time WSI computer, undertaken at IPI RAN within the HREC project, briefly, resulted in:

a) based is the S^3 schemotechnique for design of WSI circuits;

b) formulated are principles of architecture organization for the HRECs;

c) outlined is the structure of a commercial CAD system RONIS, and its development is launched;

d) found are potential application areas for the HRECs;

e) investigated and estimated are requirements to properties of real-time HREC computers: performance, application and ecological safety, vitality, prolonged lifetime, ability to redevelopment, economical recompensation (encompassed by an integral quality of dependability);

f) revealed is necessity of transition to self-synchronous interactions in computers of next generations at ULSI, GSI, and WSI integration levels and, particularly, in the IPI RAN's HRECs.

Note.   The necessity of transition to self-synchronous interactions is derivative from the following trends:
   • involving a higher architecture organization with a structural parallelism
   • complicating hardware and software with preserving reliability
   • introducing a new element base adequate to the new architecture organization
   • complying with existing trends in development of computing systems
   • searching for other sources of economical recompensation of complicated WSI systems.

3. The proposed by IPI RAN new nonconventional recurrent HREC architecture is distinguished by several original features: processes recurrence, single combined information flow of self-determined data, and self-synchronous interactions. These features are interlinked one to others as mutually complementary and consist in a conceptual base of the architecture.

The self-synchronization meets requirements of building WSI systems and provides conceptual, information, hardware, and software implementation integrity.

6   Software Demo Products

6.1   Demo version of the system FORCAGE 3.1

The demo version of the S^3 circuit CAD system FORCAGE 3.1 comprises two subsystems:

- for analysis of S^3 circuits — BTRAN, revision 1.0
- for synthesis of S^3 circuits —TRASYN, revision 3.0.

The demo version of the system FORCAGE is placed in the directory FORCDEMO, with three subdirectories:

- the subdirectory *BIN* contains auxiliary components of both subsystems
- the subdirectory *BTRAN* contains BTRAN application examples
- the subdirectory *TRASYN* contains TRASYN application examples.

The system FORCAGE is designed to operate in a DOS environment. The subsystem BTRAN applies, during analysis, an extended DOS mode. Each subdirectory has a command batch file to involve a corresponding subsystem: *btran.bat* and *trasyn.bat*, respectively. An example for the subsystem BTRAN is described below in the subsection 6.3.1. An example for the subsystem TRASYN is described in the subsection 6.3.2.

6.2   Demo roll of the CAD system RONIS

The demo roll *ronis.ppt* represents a future S^3 circuit CAD system. It shows:

- assignment of the CAD system RONIS
- basic features of the system
- structure of a system under design
- tasks being solved
- S^3 circuits design ways.

6.3   S³ circuit design with the CAD system FORCAGE

6.3.1. Example of analysis of S³ circuits with the subsystem BTRAN

The subsystem BTRAN enables checking of a circuit being designed on self-synchronousity (semi-modularity). As an example, let consider analysis of the shift register S³-SR8-3.

The source circuit description is given in the file *FORCDEMO\BTRAN\sr8_3.cir*, in the circuit description language of the subsystem BTRAN. Signal names in the description correspond to *fig. 1.61*. A 5-bit implementation of the register S³-SR8-3 is analyzed as having all two specific register bits: entrance and intermediate bits.

The subsystem BTRAN analyzes a circuit only for a single given initial state. Therefore, for comprehensive check of a circuit, the BTRAN subsystem must run multiple times with all possible legal initial states of the circuit defined in the source circuit description file. *Fig. 6.1* shows results of a single running of the subsystem BTRAN on Pentium-100, with the source description file *sr8_3.cir*.

Analysed states: 2318, strats: 187 by time 15.82
Circuit is semimodular
The Circuit Parameters: go through operational cycles - 1
In the operational cycle: states - 232 layers - 24
Layer length in the cycle: min - 4 max - 16

Fig. 6.1. Analysis results with BTRAN for S³-SR8-3

The analysis of the source description of the register S³-SR8-3 with the subsystem BTRAN confirms semi-modularity of the circuit at diverse initial states.

6.3.2. Example of synthesis of S³ circuits with the subsystem TRASYN

In this example, the task is set as follows. Synthesized is a control circuit for a hypothetical ring railway having six spans. Two trains move along the railway in the same direction. No limitation are put on their speed and load/unload time. At least one empty span must be always left between the trains for safety.

With durations of controlled processes being arbitrary, a correct control circuit must be invariant to delays of elements consisting in the system, i.e. to belong to the class of self-synchronous circuits.

A control circuit providing the described operation mode is given in the file *FORCDEMO\TRASYN\s1_tr6.di* in the CD description language of the subsystem TRASYN.

In the CD description, the signals T = {t1, ... , t6} characterize positions of the trains on the spans 1 ÷ 6 (ti = 1 if a train is on i-th span and ti = 0 otherwise) while the signals S = {s1, ... , s6} characterize states of semaphores on the spans (si = 1 if a semaphore is open and si = 0 otherwise) For the control circuit under consideration, the signals t1, ... , t6 are input signals from the environment, and the signals s1, ... , s6 are its outputs.

Arrival of a train to i-th span is a complex event assuming assertion of the signal +ti and negation of the signal –t(i-1). The train transits to (i+1)th span, an event +t(i+1), if it has arrived to i-th span, a complex event +ti and –t(i-1), and the semaphore s(i+1) is open. Negation of the signal +ti is conditioned on assertion of +t(i+1).

Opening of a semaphore on i-th span (+si) is permissible only if the train has left (i+1)th span an event -t(i+1)*-s(i+2). Its closing occurs just after the train appears on i-th span (+ti).

Synthesis of the CD described above with the TRASYN subsystem confirms its correctness and forms a set of logic functions for the control circuit (see the file *s1_tr6.cir*).

Processing of the given CD is demonstrated in the roll *forcdemo.ppt* transmitted earlier. The graphical representation of the CD is seen on the slides 25 and 26, the process of checking and building of logic circuit description — on the slides 33 and 40, the graphical image of the obtained control circuit — on the slide 41.

7 Global Conclusion

1) The principle of self-synchronous interaction is rather fruitful and universal when applied to computer systems design. It is in a great degree applicable to organization of hardware and software operations in multisystems with total parallelism. It allows designers to overcome a lot of problems concerned with WSI circuits design.

2) The S^3 schemotechnique transfers automatically VLSIs to a "New VLSI Generation". Though the main goals of various programs of VLSI design for prospective real time computers are increase of their speed and packaging density, we consider some other requirements to be not less important. A set of these requirements can be combined by a concept of "exploitability" (i.e. exploitation perfection) that assumes necessity to have, within VLSIs, embedded circuits for detection and localization of internal defects, reserve circuits with possibility of automatic replacenent for self-repairing. The S^3 schemotechnique meets those requirements in a maximum degree.

3) The results of simulation tests and engineering estimations of competing schemotechniques (self-synchronous and synchronous) enables concluding that the S^3 schemotechnique provides noticeably higher real operational speed. Therefore, even in application areas where the operational reliability is not decisive the S^3 schemotechnique may happen preferable. The speed increase (maximumal is three times higher comparing with the synchronous schemotechnique) is achieved by some additional hardware expenses (in transistors — $1.5 \div 2$ times), with S^3 circuits possessing the following features that are not available in traditional synchronous analogs (qualitative estimations):

— higher certainty of information processing due to getting rid of timing and arbitration problems;

- guaranteed operational safety due to the self-checking and ability to shutdown in the case of any malfunction;

— extended life-time due to parametric fault-tolerance.

4) Application of the S^3 schemotechnique in order to increase the readiness coefficient of available computer systems (permitting breaks for repairing) is more preferable and economically advisable: speed of the self-checkable S^3 circuits is higher then of their self-checkable synchronous

analogs while hardware expenses are lower. Besides features listed above, the S^3 circuits possess some more facilities not available in the synchronous self-checkable analogs, such as:

— lower level of noise generation because of elimination of high frequency clocking system and decreasing possibility of simultaneous switching of a large number of elements;

— longer battery lifetime due to lower power consumption under operation (less hardware) and during idle periods for inactive elements of a computer system (no element switchings), elimination of power-consuming clocking system, and more even distribution of dissipated power (power consumption).

5) Application of the S^3 schemotechnique in fault-tolerant real time systems is most preferable in comparison eith the synchronous schemotechnique: the fault-tolerant S^3 circuits are characterized by higher operational speed and lower hardware overheads comparing with their fault-tolerant synchronous analogs. The S^3 schemotechnique opens a way to creation of fault-tolerant WSI circuits.

Besides, the fault-tolerant S^3 circuits possess some features not available in synchronous fault-tolerant analogs (qualitative estimations):

— increased production output of operational circuits due to ability to parry technological defects in wafers (a simpler implementation of a reconfiguration) and softer technological requirements of manufacturing;

— simplicity and less hardware expenses of reservation at any level;

— easier testing of WSI circuits and wafers.

The investigations results allow us to hope that the S^3 schemotechnique will provide creation of dependable computing systems which surpass modern systems in respect to basic operational characteristics: performance, reliability, safety, viability, size, and power consumption.

6) The process of S^3 circuits design has some specific features caused by necessity to meet the requirements of the Muller's model. This makes impossible developments using only conventional VLSI CAD systems. Some dedicated tools are required for the S^3 circuits design.

The CAD system FORCAGE used at IPI RAN, despite restrictions on the size of circuits being designed, gives nevertheless possibility to design S^3 circuits at the functional-logical level. The design of complicated circuits have been performed by dividing them in a number of simpler subcircuits. The "hand-made" completion have been performed using the new subsystem BTRAN for S^3 circuits analysis.

7) The modeling results available in the 2nd part of this report are obtained by logic simulation of a simple test circuit. More unprejudiced and comprehensive estimations can be obtained as a result of design of a functionally more complicated device or, even better, in the case of design of some complete microcomputer system involving all stages up to its production.

The experience and S^3 circuits design tools available at IPI RAN allow statement of readiness to accomplish such a work, with some conventional VLSI CAD system being also involved.

8) The listed earlier merits of S^3 circuits predetermine expansion of possible areas where they may be used. This leads to the necessity of development of a commercial CAD system comprising efficient tools for description, analysis, and synthesis of S^3 circuits. With this fact taken into account, proposed was a new S^3 circuits design concept with the CAD system RONIS that is intended to solve only the tasks specific for S^3 circuits being nevertheless compatible with conventional VLSI CAD systems via embedded interfaces.

The system RONIS will enable design of circuits of practically unlimited complexity at functional-logical, gate, and transistor levels, with the topology level being supported by existing conventional topological CAD systems.

The system RONIS will also include the possibility to support designing quasi-self-synchronous circuits (if necessary).

The S^3 analysis algorithms implemented in the system RONIS which use the change diagrams (the subsystem ASPECT) will allow significant increase of the size of analysed circuits, comparing to the system FORCAGE.

9) Our group at IPI RAN made an attempt to investigate the merits of the self-synchronization principle of interaction by an experimental design of a real time WSI computer, within the HREC project. The results of this research are as follows:

a) stated was the choice of the S^3 schemotechnique as a basis for WSI circuits design;

b) formulated were the basic principles of an HREC architecture;

c) determined were the main areas of HREC applications (control of critical objects and processes in real time), formulated were the requirements to the HREC facilities for operation in real-time systems (performance, application safety, viability, extended lifetime, ecological safety, expandability, economical justification), with all listed features being implemented in the HRECs as a generalized feature — dependability;

d) revealed were some factors indicating on expediency of transition to the self-synchronous interaction principle in computers of a new generation approaching the ULSI-, GSI-, WSI-integration levels, especially in real-time systems:

— transition to more highly organized architectures (structural parallelism),

— growth of hardware and software expenses without lowering the level of reliability (performance, computation stability, etc.),

— adequacy of the element base to the levels of architecture organisation of hardware and software,

— concordance with existing trends in evolution of computing systems,

— necessity of new sources of economical justification of complicated WSI systems;

10) proposed was a non-conventional recurrent architecture for the HREC based on some new principles (processes recurrence, single information flow, self-determined data, and self-synchronization of interaction), with all these principles being connected to each other as mutually complementary and consisting in a conceptual base of the architecture; the self-synchronization corresponds to the requirements to WSI systems development and provides conceptual, informational, hardware and software integrity of their implementation;

11) the features of the HREC architecture listed in the previous paragraph, the self-synchronization especially, make real its «on wafer» implementation (at the WSI integration level) eliminating completely synchronization problems, which confine, in synchronous VLSIs (VLSI-computers), development of full-scale on-chip cell structures and interfaces for them (the WSI/CMOS technology has been selected as allowing, at early phases of development process, adaptation of an architecture to all possible variants of its physical implementation as VLSI, ULSI, or WSI systems optimal on performance, reliability, power consumption and cost; althogh implementation of the WSI system is the matter of nearest future, bearing their features in mind at early stages of development will allow a lot of mistakes to be avoided in future);

12) the analysis algorithms described in the section 3 and based on CD mathematics are universal for analysis of concurrent processes and efficiently solve the following problems:

— testing of self-synchronized concurrent processes (with both the «transparent» CAD RONIS and logic level only CAD system FORCAGE),

— analysis of concurrent processes in hardware as well as in software (exposition of parallelism of algorithms and programs, for example, in «parallel» compilers).

There are three factors which can really provide, in our opinion, the computer performance increase, namely:

— recurrent self-determined data,

— the self-synchronization of processes in hardware and software,

— the parallel processes analysis subsystem on base of change diagrams.

The authors hope that the original solutions in architecture, schemotechnique, and technology, being embeded in an architecture, will provide to the HREC a number of new advantages as compared with competiting computers, developed on both the traditional base and known non-conventional approaches.

In the literature, as far as the authors know, there were no publications on projects of prospective computers similar to the one represented in this report.

Literature

1.1. Апериодические автоматы / Под ред. В.И. Варшавского - М.: Наука, 1976, 424 с.

1.2. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах / Под ред. В.И. Варшавского - М.: Наука, 1986, 398 с.

1.3. Плеханов Л.П. Базовые элементы самосинхронных схем. Ежегодник ИПИ РАН "Системы и средства информатики", вып. 7. - М.: Наука, 1994.

1.4. Peter A. Beeret and Tereza H.-Y. Meng, Semi-modularity and testability of speed-independent circuits // Integration, the VLSI journal, 1992, N 13, p.p. 301-322.

1.5. Поспелов Д.А. Логические методы анализа и синтеза схем / Изд. 3-е, перераб. и доп., М., "Энергия", 1974, 368 с.

1.6. John Theus. The Futurebus+ Handbook. First Edition. TheUs Group. VFEA International Trade Association Scottsdate - Zaltbommel. A VITA Publication, 1993, 382 pp.

1.7. Nigel C.Paver. The Design and Implementation of an Asynchronous Microprocessor. A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Science. Department of Computer Science, 1994, 162 pp.

1.8. Варшавский и др. Отчет о НИР по теме: "Теория, методы и программная поддержка проектирования самосинхронных схем" (промежуточный). М., ИПИ РАН, 1990

2.1. Mih. Kishinevsky, Al. Kondratyev, Alec. Taubin, Vik. Varshavsky. "Concurent hardware: the theory and practice of timing design". J. Wiley & Sons. "In Parallel computing", 1994, 368pp.

4.1. The VLSI-Programming Language Tangram and its Translation into Handshake Circuits. K. van Berkel, J. Kessels, M. Ronchen and oth. Proc. of the EDAC-91. Los Alamitos, IEEE Comp. Ass. Press, 1991, p.384-389.

4.2. Martin A.J. A Synthesis Method for Self-Timed VLSI Circuits. - IEEE Intern. Conf. on Comp. Design: VLSI in Computers & Processors, 1987, p.224-229.

4.3. Compton J., Albicki A. Self-Timed Pipeline with Adder. Proceeding GLSV'92, Kalamazoo, MI, 1992, p.109-113.

5.1. A. Avizienis and J.C. Laprie. Dependable Computing: From Concepts to Design Diversity. Proc. IEEE, vol. 74, No 5, May 1986, pp. 629-638

5.2. B. Robinson. An ultrareliable future. Electronic Engineering Times, 1991, February 25, p. 32