

# On the Delay-Sensitivity of Gate Networks

Janusz A. Brzozowski, *Member, IEEE*, and Jo C. Ebergen

**Abstract**—In classical switching theory, asynchronous sequential circuits are operated in the *fundamental mode*. In this mode, a circuit is started in a stable state, and then the inputs are changed to cause a transition to another stable state. The inputs are not allowed to change again until the entire circuit has stabilized. In contrast to this, delay-insensitive circuits—the correctness of which is insensitive to delays in their components and wires—use the *input-output mode*. Here, it is assumed that inputs may change again, in response to an output change, even before the entire circuit has stabilized. In this paper, we show that such commonly used behaviors as those of the set-reset latch and Muller's C-ELEMENT do not have delay-insensitive realizations, if gates are used as the basic components. In fact, we prove that no nontrivial sequential behavior with one binary input possesses a delay-insensitive realization using gates only. Our proof makes use of the equivalence between ternary simulation and the General Multiple Winner model of circuit behavior.

**Index Terms**—Asynchronous circuits, delay-insensitive circuits, fundamental mode, gate networks, input-output mode, realizability.

## I. INTRODUCTION

ASYNCHRONOUS circuits have witnessed a remarkable revival during the past decade. This revival came about on two fronts. On the one hand, important results were found concerning the analysis of asynchronous circuits using the “classical” approach [3], [4], [18]. On the other hand, new formal approaches have been developed and applied in the design of special types of asynchronous circuits, such as speed-independent circuits [7], [16], delay-insensitive circuits [2], [8], [15], quasi delay-insensitive circuits [11], and self-timed systems [6], [19]. Successful applications of these new approaches have been demonstrated by A. J. Martin [10], C. van Berkel *et al.* [21] and, most recently, by I. E. Sutherland in his Turing Award lecture [20].

There are some important differences between the “classical” and the “new” approaches. One of these differences lies in the “mode of operation” of a network, which prescribes how the environment should interact with a network in order to obtain the desired input-output behavior. In the classical approach, one applies the so-called fundamental mode operation [13] in which the environment changes the inputs and holds them fixed until the network has stabilized completely.

Manuscript received September 12, 1990; revised April 16, 1991. This work was supported by the Dutch organization for scientific research, NWO, and by the Natural Sciences and Engineering Research Council of Canada under Grants A0871 and OGP0041920.

The authors are with the Department of Computer Science, University of Waterloo, Waterloo, Ont., Canada, N2L 3G1. This work was done while J. A. Brzozowski was on sabbatical leave during 1989–1990 at the Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

IEEE Log Number 9200207.

Only after the network has reached a stable state, is the environment allowed to give the next input change. (Although some authors include the single-output-change assumption in the definition of fundamental mode operation, we do not do so here, following instead McCluskey's original definition.) In the new approaches, one applies the so-called input-output mode of operation [2], [15] in which the environment does not have to wait until the network has stabilized completely to give the next input change: an input change may be made as soon as the network has given an appropriate response to a previous input change.

Another difference between the classical and the new approaches lies in the basic formalism. In classical asynchronous circuit design, the basic formalism is Boolean algebra, where Boolean functions are the essential objects. In the new approaches, one applies an event-based formalism, where sequences of events are the essential objects. Examples of event-based formalisms are trace theory [7], [8], [17] and Petri nets [6], [14], [22]. The different formalisms lead also to the use of different sets of primitive elements. In the classical approach, the primitive elements are the logic gates, which correspond to the primitive Boolean functions. In the new approaches, the primitive elements are such elements as the Muller C-ELEMENT, TOGGLE, MERGE, and ARBITER, which correspond to basic sequences of events.

The input-output mode of operation and primitive elements, like C-ELEMENTS and TOGGLES, are used in delay-insensitive circuits. Informally, a delay-insensitive circuit is one whose correctness is insensitive to delays in the wires and primitive elements. A fundamental question is whether a different set of primitive elements for the design of these circuits is really necessary. In other words, can every delay-insensitive circuit be realized by a network using gates only? For example, does the C-ELEMENT or TOGGLE have a delay-insensitive gate realization? Although it is generally assumed that such realizations do not exist, no proof of this has been given so far. We provide such a proof in this paper. In fact, we prove a more general result: no nontrivial sequential behavior with one binary input has a delay-insensitive gate realization. In doing so, we explain the differences between the fundamental mode and the input-output mode using one formal framework that combines the event-based and state-based approach.

## II. SPECIFICATION OF INPUT-OUTPUT BEHAVIOR

When specifying the behavior of a network to be designed, we need to describe how the proposed network is to communicate with its environment through its input and output terminals. In this paper, we formally specify an *input-output behavior* by a 5-tuple  $B = \langle h, k, S, T, s_0 \rangle$ , where

- $h \geq 0$ , is the number of input variables,
- $k \geq 0$ , is the number of output variables,
- $S \subseteq \{0, 1\}^{h+k}$ , is the state set,
- $T \subseteq (S \times S) - \{(s, s) | s \in S\}$ , is the transition set, and
- $s_0 \in S$  is the initial state.

The vector  $u = u_1, \dots, u_h$  will be used to represent the input variables, and  $v = v_1, \dots, v_k$  will denote the output variables. The vector  $u_1, \dots, u_h, v_1, \dots, v_k$  represents the state of  $B$ . A state  $c \in \{0, 1\}^{h+k}$  is sometimes written  $c = ab$ , where  $a \in \{0, 1\}^h$  and  $b \in \{0, 1\}^k$ ; this permits us to identify the input and output components of  $c$ . A transition  $(ab, a'b')$  is said to be an *input transition*, if only the inputs have changed, i.e., if  $a \neq a'$  and  $b = b'$ ; it is an *output transition*, if only the outputs have changed, i.e., if  $a = a'$  and  $b \neq b'$ . Note that a transition may be neither an input transition nor an output transition, namely, when  $a \neq a'$  and  $b \neq b'$ . A state of the input-output behavior is said to be *stable*, if it has only input transitions leaving it or has no transitions at all; otherwise, it is *unstable*.

To illustrate the definitions given above, we present a number of examples.

**Example 1:** An input-output behavior of the set-reset latch, under the assumption that the set and reset inputs are never 1 at the same time, is given in Fig. 1. The latch has binary inputs  $u_1$  ("set") and  $u_2$  ("reset") and a binary output  $v$ . The state set consists of all binary triples  $u_1 u_2 v$ , except 110 and 111, and the transitions are shown in Fig. 1. For convenience, we label each transition with the variable that changes during the transition. The initial state is 000. (For brevity, we often write tuples without commas; for example, 110 represents (1,1,0).) □

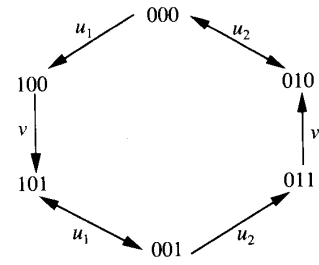


Fig. 1. Input-output behavior of latch with  $u_1 u_2 = 11$  disallowed.

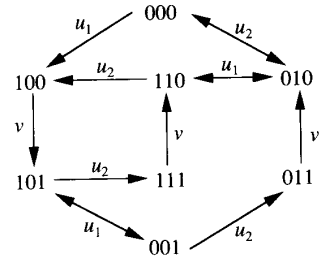


Fig. 2. Behavior of latch with  $u_1 u_2 = 11$  allowed.

**Example 2:** The graph of Fig. 2 shows an input-output behavior of the latch where  $u_1 u_2 = 11$  is permitted, but where  $u_1$  and  $u_2$  never change simultaneously. Notice that Fig. 1 is symmetrical, but Fig. 2 is not. This asymmetry is introduced in the states where  $u_1 u_2 = 11$  by giving the "reset" priority over the "set." □

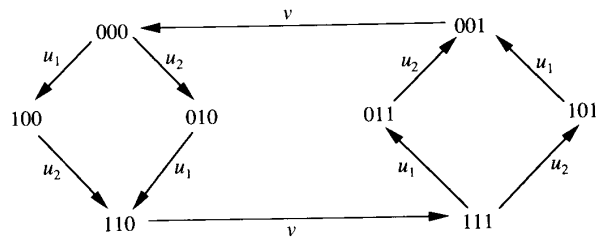


Fig. 3. Input-output behavior of C-ELEMENT.

**Example 3:** In Fig. 3 we show an input-output behavior of the C-ELEMENT with inputs  $u_1$  and  $u_2$  and output  $v$ . □

**Example 4:** Fig. 4 shows an input-output behavior of a TOGGLE with input  $u$  and outputs  $v_1$  and  $v_2$ . If we count the input changes starting from the initial state 000, each odd input change causes a change in output  $v_1$  and each even input change causes a change in output  $v_2$ . □

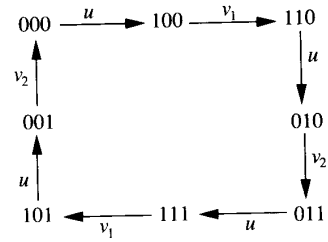


Fig. 4. Input-output behavior of TOGGLE.

In this paper we study a restricted class of input-output behaviors. We consider only those input-output behaviors where a) the initial state is stable; b) each unstable state has exactly one outgoing transition, this is an output transition, and the state reached by this transition is stable; c) exactly one (input) variable changes in every input transition; and d) exactly one (output) variable changes in every output transition. We call this class of input-output behaviors *simple deterministic*. Note that the behaviors in Figs. 1, 2, 3, and 4 are simple deterministic.

The motivation for the assumptions above is the following. First, it would be unreasonable to start a network in an unstable state and expect it to operate properly; in fact, most practical designs provide for special "reset" inputs to make sure that

the network is properly initialized. Second, we are interested in deterministic behavior in many common circuits, such as counters, latches or C-ELEMENTS. Thus the final outcome of a transition should be a unique stable state. More will be said about this later, in connection with oscillations. We also point out that the first two assumptions are very much in line with the classical theory of fundamental-mode operation. The last two assumptions—that a single signal changes during any transition—are not restrictive, in view of the fact that we are interested in delay-insensitive networks. For such networks, one necessarily takes into account wire delays. In the presence

of wire delays, it would be difficult to argue strongly in favor of a model where simultaneous changes play a significant role: a small deviation in a wire delay is sufficient to change a simultaneous occurrence of two signals into a sequential occurrence. Finally, the assumptions simplify a number of proofs.

### III. THE GENERAL MULTIPLE WINNER MODEL

In order to address the question whether a gate network realizes a certain input-output behavior, we first discuss the analysis of gate networks. We view a *gate network* [3] as a directed labeled graph  $N = \langle n, m, V, E, g \rangle$ , where

- $n \geq 0$  is the number of input variables;
- $m \geq 0$  is the number of gate variables associated with the gates of  $N$ ;
- $V = \{1, \dots, n + m\}$  is the set of vertices of  $N$  corresponding to the  $n$  input terminals and  $m$  gates;
- $E \subseteq V \times V$ , is the set of edges of  $N$ , corresponding to the wires connecting the inputs and gates;
- $g = g_1, \dots, g_m$  is the vector of Boolean functions associated with the gates of  $N$ .

The vertices of  $N$  are classified as follows. The first  $n$  vertices, numbered  $1, \dots, n$ , correspond to the input terminals and are all of indegree 0. Associate with input vertex  $i$  the variable  $x_i$ . The remaining vertices are all of indegree greater than 0; they correspond to the gates of the network and are numbered  $n + 1, \dots, n + m$ . Associate with gate vertex  $n + i$  the variable  $y_i$ . Let the vectors  $x = x_1, \dots, x_n$  and  $y = y_1, \dots, y_m$  denote the network inputs and the gate outputs, respectively. The vector  $xy$  represents the *state* of the network.

The function  $g_i$  is the *excitation* of gate  $i$ , mapping  $\{0, 1\}^{d_i}$  to  $\{0, 1\}$ , where  $d_i$  is the indegree of vertex  $i$ . The variables on which  $g_i$  may depend are shown by the edges in  $E$  that lead to vertex  $n + i$ . It is often convenient to view  $g_i$  as a function of the entire state, i.e., to treat it as a mapping  $g_i(xy)$  from  $\{0, 1\}^{n+m}$  to  $\{0, 1\}$ . Notice that gates are not confined to simple AND and OR gates, but may implement any Boolean function.

In any given state  $ab$  the output  $b_i$  of gate  $i$  may differ from its excitation  $g_i(ab)$ . Gate  $i$  is then said to be *unstable*; otherwise it is *stable*. A state of the network is *stable* if all the gates are stable in that state.

To analyze a given network in any state we use the *General Multiple Winner (GMW)* model [5]. In this model the relation  $R_a$  defines the possible transitions among the network states on the basis of the unstable gates in each state. The relation  $R_a$  is defined for any (fixed) input vector  $a \in \{0, 1\}^n$ . Let

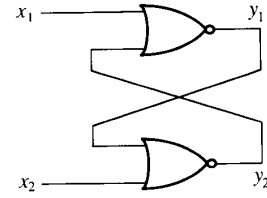
$$D_a = \{ab \mid b \in \{0, 1\}^m\},$$

i.e., let  $D_a$  represent all the total network states that have the input component fixed at  $x = a$ . For any  $d$  in  $D_a$  define the set of unstable gate indexes to be

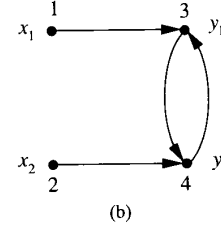
$$U(d) = \{i \mid d_{n+i} \neq g_i(d)\}.$$

Next, define  $R_a$  as the smallest relation on  $D_a$  with

$$d R_a d, \text{ if } U(d) = \emptyset, \text{ i.e., if } d \text{ is stable,}$$



(a)



(b)

Fig. 5.(a). NOR latch: Gate network. (b) NOR latch: Graph.

$d R_a d^W$ , if  $W$  is a nonempty subset of  $U(d)$ ,

where the notation  $d^W$  stands for the vector  $d$  in which all components with subscripts in  $W$  are complemented. Thus the relation  $R_a$  relates each stable state to itself and each unstable state to any state obtained by complementing a nonempty subset of the unstable gate variables. (A state with two or more unstable variables is said to have a *race*, and the gate variables with subscripts in  $W$  are the *multiple winners* of the race. The reader should also note that two equivalent notations are used in connection with relations, namely:  $aRb$  and  $(a, b) \in R$ .)

*Example 5:* To illustrate the GMW model, consider the network of the NOR latch and the corresponding graph in Fig. 5.

We have

$$V = \{1, 2, 3, 4\},$$

$$E = \{(1, 3), (2, 4), (3, 4), (4, 3)\},$$

$$x = x_1 x_2, \quad y = y_1 y_2,$$

$$g_1(xy) = \overline{(x_1 + y_2)} \text{ and } g_2(xy) = \overline{(x_2 + y_1)}.$$

The graph of the relation  $R_{00}$  is shown in Fig. 6.

Note that, after state 0000 is reached, the network may end up in stable state 0001 or in the stable state 0010 (this represents a *critical race*), or it may oscillate between the states 0000 and 0011. We consider such oscillations unacceptable and indicative of undesirable behaviors in a physical circuit. □

In the GMW analysis of a network state, a cycle must be reached eventually in the graph of the relation  $R_a$ , since the number of network states is finite. Recall that every stable network state forms a cycle of length one. Let

$$\text{cycl}(R_a, d) = \{e \in D_a \mid d R_a^* e \text{ and } e R_a^+ d\}$$

be the set of all the states reachable from state  $d$  that appear in cycles of  $R_a$ . Here,  $R_a^+$  and  $R_a^*$  denote the transitive and the reflexive-and-transitive closures of  $R_a$ , respectively.

The transient states occurring during a transition from a given state are often of no interest. A cycle is *transient* if there exists a gate which is unstable in every state of the

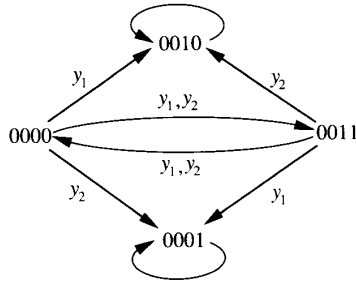


Fig. 6. Relation  $R_{00}$ , partial network behavior of NOR latch started in state 0000.

cycle and has the same value in all these states; otherwise it is *nontransient*. Let  $trans(R_a, d)$  be the set of all the states in  $cycl(R_a, d)$  that appear only in transient cycles of the graph of  $R_a$ , and let

$$out(R_a, d) = cycl(R_a, d) - trans(R_a, d).$$

We interpret  $out(R_a, d)$  as the final outcome of the GMW analysis of the network started in state  $d$ , where the inputs are kept fixed at  $a$ . The case of most interest is that in which the outcome consists of a single state. This state is necessarily stable and is reached from  $d$  in a finite time. Consequently, if the outcome consists of a single state, the network does not exhibit nontransient oscillatory behavior. As we stated before, nontransient oscillatory behaviors are indicative of undesirable behaviors in a physical circuit and are therefore unacceptable.

From the construction of the relation  $R_a$  it follows that an arbitrary finite delay is associated with each gate, but that wires have no delays. If we wish to represent wire delays, we do so by treating them as one-input, one-output gates performing the identity function. In fact, this is exactly what we do in Section V, when we examine delay-insensitive realizations of input-output behaviors. There, a wire delay is included for each connection wire in a gate network.

#### IV. FUNDAMENTAL MODE REALIZATION

In order to realize a specified input-output behavior, we need not only an appropriate network, but also a co-operating environment. Roughly speaking, the environment is expected to produce the "correct" input changes at the "correct" times. For example, consider the input-output behavior of the latch of Fig. 1, started in state  $u_1 u_2 v = 000$ . When a "set" input is applied to a network that should realize this input-output behavior, we expect the network to respond by changing output  $v$  and moving to a state whose input-output component is  $u_1 u_2 v = 101$ . If the "set" input is then changed again, the network should move to input-output state 001, "remembering" that a "set"-pulse had been received. If the "set" input changes too soon to 0, however, some networks may fail to "remember" the "set" and may change back to state 000. In the fundamental mode operation, the environment is allowed to change the "set" input again only when the complete network has stabilized. The reader should note that the fundamental mode of operation requires that the environment must know

somehow when the network is ready to receive the next input; however, it is irrelevant to the designer of a fundamental mode circuit how this environment restriction is implemented. In this section we make the concept of fundamental mode realization [13] precise.

Before giving a formal definition of fundamental mode realization, we need to define the restriction of a network state to a subset of its state variables. For a given input-output behavior  $B$  and a network  $N$ , we want to match each input of  $B$  with exactly one input of  $N$  and each output of  $B$  with exactly one gate variable of  $N$ . One could also view this as a 1-to-1 mapping of a subset of the input variables of  $N$  to the input variables of  $B$ , and of a subset of the gate variables of  $N$  to the output variables of  $B$ . In case such a mapping exists, we say that there is a *restriction of  $N$  to  $B$* . We extend this restriction to a mapping of the states of  $N$  to states of  $B$  by simply removing all state variables in  $N$  that are not representatives of state variables in  $B$ . The restriction of network state  $q$  to  $B$  is denoted by  $q \downarrow B$ . We also generalize this notion to the restriction of any binary relation  $R_N$  on the states of  $N$  to a binary relation  $R_N \downarrow B$  on the states of  $B$  in the obvious way:

$$s (R_N \downarrow B) s' \text{ iff there exist states } q, q' \text{ of } N \text{ such that} \\ q \downarrow B = s, q' \downarrow B = s', \text{ and } q R_N q'.$$

Given a restriction of a network  $N$  to a behavior  $B$ , we will simulate the network behavior step by step with the environment behavior as given by  $B$ , in order to check whether  $N$  is a proper fundamental mode realization of  $B$ . First, if  $N$  is to realize  $B$ , there must exist an initial stable state  $q_0$  of  $N$  whose restriction is  $s_0$ . Thus we will consider an initialized network  $(N, q_0)$  as a possible realization of  $B$ . After that,  $N$  should imitate  $B$  as follows. If, in  $B$ , an input change is followed by an output change, then, in  $N$ , the corresponding input change *must* eventually result in the corresponding output change. Thus phenomena such as deadlock and livelock [9] in  $N$  are not tolerated. We will take appropriate precautions to make sure that our definition of realization does not permit such phenomena to occur. Furthermore, if an output of  $B$  is to change once, then the corresponding output of  $N$  should not change more than once; thus dynamic hazards are not acceptable. Similarly, if an output of  $B$  is *not* supposed to change, then neither is the corresponding output of  $N$ ; consequently, static hazards are also excluded. We will show that our definition of realization is correct from this point of view. All the concepts mentioned above will be made precise in this section.

The details of the simulation are as follows. The set of network states that can be reached from  $q_0$  is denoted by  $Q_F$ , and the possible transitions among these states are recorded in a relation  $R_F$ . Let  $B = \langle h, k, S, T, s_0 \rangle$  be a simple deterministic behavior, let  $N = \langle n, m, V, E, g \rangle$  be a network, let  $\downarrow$  be a restriction of  $N$  to  $B$ , and let  $q_0$  be an initial state of  $N$  such that  $q_0 \downarrow B = s_0$ . The set  $Q_F \subseteq \{0, 1\}^{n+m}$  and binary relation  $R_F \subseteq Q_F \times Q_F$  are defined inductively as follows:

$$\text{Basis: } Q_F = \{q_0\}, R_F = \emptyset.$$

Induction Step: For each  $q \in Q_F$ , let  $a \in \{0, 1\}^n$  denote the input vector of  $q$ . We have the following rules:

- *Rule 1:*  
If  $q$  is unstable and  $qR_a q'$  for some  $q'$ , then add  $q'$  to  $Q_F$  and  $(q, q')$  to  $R_F$ .
- *Rule 2:*  
If  $q$  is stable,  $s = q \downarrow B$ ,  $sTs^{(i)}$  is a transition in  $B$ , and  $u_i = x_j$  in the restriction of  $N$  to  $B$ , then add  $q^{(j)}$  to  $Q_F$  and  $(q, q^{(j)})$  to  $R_F$ .

(Recall that  $q^{(j)}$  denotes state  $q$  with gate variable  $j$  complemented.) The induction step is applied until  $Q_F$  and  $R_F$  can no longer be enlarged.

Note that, when  $B$  is simple deterministic and  $s$  is unstable,  $s$  has only one output transition. Consequently, for an unstable  $s$  there exists no input  $u_i$  of  $B$  such that  $sTs^{(i)}$ , and any stable state  $q \in Q_F$  with  $q \downarrow B = s$  will not be related to any state by  $R_F$ .

A tentative definition of “ $(N, q_0)$  is a fundamental-mode realization (F-realization for short) of  $B$ ” might be that

$$R_F \downarrow B - I = T$$

must hold, where  $I$  is the identity relation on  $S$  and “ $\downarrow$ ” has a higher priority than “ $-$ ”. (Notice that self-loops may be created when  $R_F$  is restricted to  $B$ . Since such self-loops are not allowed in  $T$ , we subtract them from the restriction.) This definition is not quite satisfactory, however, since phenomena similar to deadlock and livelock may occur in the realization. These phenomena are illustrated in the following examples.

*Example 6:* Let behavior  $B_1$  have input  $u$ , output  $v$ , initial state 00, and  $S_1$  and  $T_1$  be defined by

$$B_1 : 00 \xrightarrow{u} 10 \xrightarrow{v} 11 \xrightarrow{u} 01 \xrightarrow{v} 00,$$

where the last state is the same as the initial state, so the behavior repeats. Let network  $N_1$  have input  $x$  and two gates  $g_1, g_2$  with outputs  $y_1, y_2$ . The functions of the gates are given by

$$g_1(xy) = y_1 + y_2 \cdot \bar{x} \text{ and } g_2(xy) = x \cdot \bar{y}_1 + \bar{y}_1 \cdot y_2,$$

where  $\cdot$  denotes the AND operation. See Fig. 7.

Let the restriction be defined by  $u = x$  and  $v = y_2$ . Furthermore, let  $q_0 = 000$ ; this is a stable network state. Applying the above construction for  $Q_F$  and  $R_F$ , we find the following. Since  $q_0 = 000$  is stable, Rule 2 can be applied. There is only one input transition  $(00, 10)$  in  $B_1$ , which gives rise to  $(000, 100)$  in  $R_F$ . State 100 is unstable, and Rule 1 yields  $(100, 101)$  in  $R_F$ . State 101 is again stable, and Rule 2 with input transition  $(11, 01)$  from  $B_1$  gives  $(101, 001)$  in  $R_F$ . In state 001,  $y_1$  is unstable, so Rule 1 gives  $(001, 011)$  in  $R_F$ . State 011 is also unstable, and Rule 1 yields  $(011, 010)$  in  $R_F$ . Next, state 010 is stable, and Rule 2 now gives  $(010, 110)$  in  $R_F$ . The last state 110 is also stable, and the simulation ends here. In summary, we have

$$000 \xrightarrow{u} 100 \xrightarrow{v} 101 \xrightarrow{u} 001 \xrightarrow{y_1} 011 \xrightarrow{v} 010 \xrightarrow{u} 110.$$

Although the condition  $R_F \downarrow B_1 - I = T_1$  is satisfied, a phenomenon similar to deadlock occurs: as soon as state 110 is reached, no new output can ever be produced.  $\square$

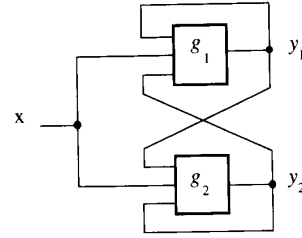


Fig. 7. Network  $N_1$ .

The previous example illustrated that, although a network may stabilize, there is no guarantee that it will produce the required output. The following example illustrates that a network may even fail to stabilize.

*Example 7:* Consider the behavior  $B_2$  with input  $u$ , output  $v$ ,  $s_0 = 10$ , and  $S_2$  and  $T_2$  given by

$$B_2 : 10 \xrightarrow{u} 00 \xrightarrow{v} 01.$$

The network of Fig. 8 shows the NOR latch of Fig. 5, but now with the inputs tied together and the outputs of the two NOR gates connected to a XOR gate.

The restriction of this network  $N_2$  to  $B_2$  is given by  $u = x$  and  $v = y_3$ . The initial state of  $N_2$  is 1000. We find  $R_F$  as shown in Fig. 9, and  $R_F \downarrow B_2 = \{(10, 00), (00, 00), (00, 01)\}$ .

Consequently,  $R_F \downarrow B_2 - I = T_2$ . We also observe, however, that the network may keep oscillating between states 0000 and 0110, and thus never reach state 0011 or 0101. In other words, if the network fails to stabilize, the required output  $v = 1$  may never be produced. This phenomenon is similar to livelock.  $\square$

In order to avoid realizations like that in Example 6, we introduce the notion of F-consistency. We say that  $(N, q_0)$  with restriction  $\downarrow$  is *F-consistent with respect to B* iff, for all  $q \in Q_F$ ,

$$q \downarrow B \text{ is unstable} \Rightarrow q \text{ is unstable.}$$

The F-consistency condition requires that, as long as an output change is expected with respect to  $B$ , network  $N$  will remain unstable. Notice that, in Example 6 above, 100 and 110 both map onto the unstable state 10; state 100 is unstable, but 110 is not. Consequently,  $(N_1, 000)$  is not F-consistent with respect to  $B_1$ .

In order to guarantee stabilization, we require the absence of nontransient oscillations. For this purpose we introduce the notion of F-determinism as follows. We say that  $(N, q_0)$  with restriction  $\downarrow$  is *F-deterministic with respect to B* iff for all  $q \in Q_F$ , the result  $out(R_a, q)$  of the GMW analysis consists of only one state, where  $a$  denotes the input vector of  $q$ . The F-determinism condition requires that the GMW analysis of any state in  $Q_F$  is guaranteed to stabilize in exactly one state. Obviously, if there exists a nontransient oscillation in a GMW analysis of a state  $q$  in  $Q_F$ , then  $out(R_a, q)$  contains more than one state. Consequently, if the F-determinism condition is satisfied, then there are no nontransient oscillations. Notice that, in Example 7 above,  $out(R_0, 0000) =$

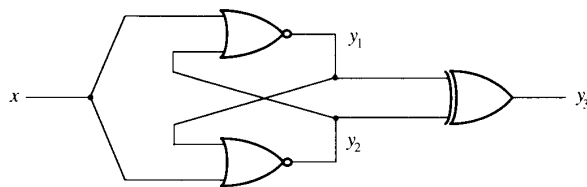


Fig. 8. Network  $N_2$ .

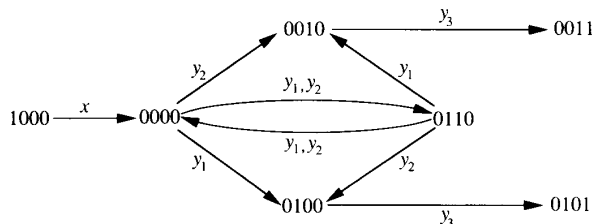


Fig. 9. Relation  $R_F$  for  $N_2$ .

{0011, 0101, 0000, 0110}. Consequently,  $(N_2, 1000)$  is not F-deterministic with respect to  $B_2$ .

It was proved in [3] that, if a delay is included in the circuit model in each connection wire (i.e., if the gate-and-wire-delay model is used), then there exists a nontransient oscillation iff  $out(R_a, q)$  consists of more than one state. Consequently, for the gate-and-wire delay model, the absence of nontransient oscillations is equivalent to F-determinism. The definition of F-realization now reads as follows:

*Definition 1:* An initialized network  $(N, q_0)$  with restriction  $\downarrow$  is an F-realization of behavior  $B$  iff  $(N, q_0)$  is F-consistent and F-deterministic with respect to  $B$ , and  $R_F$  satisfies the condition:

$$R_F \downarrow B - I = T,$$

where  $I$  is the identity relation on the set of states of  $B$ .  $\square$

We illustrate this definition by additional examples.

*Example 8:* Consider the behavior  $B_3$  with input  $u$ , output  $v$ , initial state  $s_0 = 00$ , and  $S_3$  and  $T_3$  given below.

$$B_3 : 00 \xrightarrow{u} 10 \xrightarrow{v} 11 \xrightarrow{u} 01.$$

For the network  $N_3$ , we consider the NOR latch of Fig. 5 again, now started in state 0010. Let the restriction of  $N_3$  to  $B_3$  be defined by  $u = x_1$  and  $v = y_2$ . Performing the construction for  $Q_F$  and  $R_F$ , we obtain

$$R_F = \{(0010, 1010), (1010, 1000), (1000, 1001), (1001, 0001)\}$$

$$R_F \downarrow B_3 = \{(00, 10), (10, 10), (10, 11), (11, 01)\}.$$

One verifies that, for every state  $q \in Q_F$  for which  $q \downarrow B_3$  is unstable in  $B_3$ ,  $q$  is also unstable in  $N_3$ . Consequently,  $(N_3, 0010)$  is F-consistent with respect to  $B_3$ . Furthermore,  $(N_3, 0010)$  is F-deterministic with respect to  $B_3$ , and  $R_F \downarrow B_3 - I = T_3$ , i.e.,  $(N_3, 0010)$  is an F-realization of  $B_3$ .  $\square$

*Example 9:* As a second example, consider behavior  $B_3$  with the network  $N_4$  of Fig. 10.

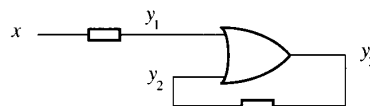


Fig. 10. Network  $N_4$ .

The network has input  $x$ , gate variables  $y_1, y_2, y_3$ , and excitation functions  $g_1 = x, g_2 = y_3$  and  $g_3 = y_1 + y_2$ . Assuming the network starts in state 0000, and that the restriction is defined by  $u = x$  and  $v = y_3$ , we obtain

$$R_F = \{(0000, 1000), (1000, 1100), (1100, 1101), (1101, 1111), (1111, 0111), (0111, 0011)\}.$$

Again, one verifies that  $(N_4, 0000)$  is F-consistent and F-deterministic with respect to  $B_3$ , and that  $R_F \downarrow B_3 - I = T_3$ , i.e.,  $N_4$  is an F-realization of  $B_3$ .  $\square$

For simple deterministic behaviors, F-realization satisfies Property 2 below with respect to static and dynamic hazards. A *static hazard* is said to occur on an output of a network during a sequence of state transitions in the relation  $R_a$ , if that output has the same value in the initial and final states of the sequence, but the complementary value in at least one intermediate state. Here the output is not supposed to change, but does change (at least twice). A *dynamic hazard* is defined similarly, except that now the output has different values in the initial and final states. Here the output is supposed to change once, but changes more than once. Before formulating the property, we introduce some terminology. A sequence  $q_0, \dots, q_k, k \geq 0$ , of network states with input component fixed at  $a$  is an  $R_a$ -sequence if  $q_i R_a q_{i+1}$  for  $i = 0, \dots, k-1$ . The restriction of an  $R_a$ -sequence to  $B$  will be called an  $(R_a \downarrow B)$ -sequence.

*Property 2:* Let network  $(N, q_0)$  with restriction  $\downarrow$  be an F-realization of simple deterministic behavior  $B$ , and let  $s$  be any state of  $B$ .

- $s$  is stable iff for all states  $q \in Q_F$ , such that  $q \downarrow B = s$ , and for any  $R_a$ -sequence from  $q$  to  $q' \in out(R_a, q)$ , there

is no change in the corresponding  $(R_a \downarrow B)$ -sequence (i.e., there are no static hazards).

- $s$  is unstable iff for all states  $q \in Q_F$  such that  $q \downarrow B = s$ , and for any  $R_a$ -sequence from  $q$  to  $q' \in \text{out}(R_a, q)$ , there is exactly one change in the corresponding  $(R_a \downarrow B)$ -sequence (i.e., the sequence has the form  $s, \dots, s, s', \dots, s'$ , where  $s' = q' \downarrow B$ , and there are no dynamic hazards).

*Proof:* Let  $(N, q_0)$  be an F-realization of  $B$  and let  $s$  be any state of  $B$ . For the first claim the condition

for all  $q \in Q_F$ , with  $q \downarrow B = s$ , and any  $R_a$ -sequence starting in  $q$ , the corresponding  $(R_a \downarrow B)$ -sequence has no changes

is equivalent to the condition state

$s$  has no output transitions,

because  $R_F \downarrow B - I = T$ . By the definition of stability and the fact that  $B$  is simple deterministic, this is equivalent to the condition that  $s$  is stable.

For the second claim we observe the following. Assume that, for all  $q \in Q_F$  with  $q \downarrow B = s$ , and any  $R_a$ -sequence starting in  $q$ , the corresponding  $(R_a \downarrow B)$ -sequence has exactly one change. Then there is an output transition from  $s$  in  $R_F \downarrow B$ . Thus, by the definition of unstable state,  $s$  is unstable. Conversely, assume that  $s$  is unstable. Then  $s$  has an output transition in  $T$  and, since  $R_F \downarrow B - I = T$ , also in  $R_F \downarrow B$ . Let  $q \in Q_F$ , with  $q \downarrow B = s$ , and take an arbitrary  $R_a$ -sequence that starts in  $q$  and ends in a stable network state. Such a stable network state exists, since  $(N, q_0)$  is F-deterministic with respect to  $B$ . Since  $(N, q_0)$  is also F-consistent with respect to  $B$  and  $s$  is unstable in  $B$ , the final stable network state of the  $R_a$ -sequence does not map onto  $s$ . Hence the  $(R_a \downarrow B)$ -sequence has at least one change.

Suppose further that the first state different from  $s$  in the  $(R_a \downarrow B)$ -sequence is  $s'$ . Since  $B$  is simple deterministic and, therefore,  $s'$  is a stable state, all successors of  $s'$  in the  $(R_F \downarrow B)$ -sequence are also  $s'$ . Consequently, there is exactly one change in any  $(R_a \downarrow B)$ -sequence. Thus, the second claim also holds.  $\square$

## V. INPUT-OUTPUT MODE REALIZATION

For the design of *delay-insensitive circuits* [15], one also starts with a specification of an input-output behavior, but now the network is operated in *input-output mode*. This mode of operation stipulates that the environment is allowed to change the inputs of the network before the complete network has stabilized, as long as this is done in accordance with the state transitions of the input-output behavior; the network must produce the outputs as specified in the input-output behavior. For example, for the input-output behavior of the TOGGLE of Fig. 4, the environment can start by changing the input  $u$ . As soon as the network for the TOGGLE responds with a change in output  $v_1$ , the environment is allowed to change the input  $u$  again, etc. The reader should observe that the input-output mode places less responsibility on the environment than does the fundamental mode: the environment can supply the next input change as soon as the network has produced an output change, and no additional information is needed.

When a network  $N$  is operated in input-output mode with respect to a behavior  $B$ , we construct the set of states  $Q_{IO}$  and the relation  $R_{IO}$ . The construction of these sets is similar to the construction of  $Q_F$  and  $R_F$  when a network  $N$  is operated in fundamental mode with respect to a behavior  $B$ . Again we assume that there exists a restriction  $\downarrow$  and a stable state  $q_0$  of  $N$  that maps onto the initial state  $s_0$  of  $B$ . The set  $Q_{IO}$  and relation  $R_{IO}$  are defined inductively as follows:

Basis:  $Q_{IO} = \{q_0\}$ ,  $R_{IO} = \emptyset$ .

Induction Step: For each  $q \in Q_{IO}$ , let  $a \in \{0, 1\}^n$  denote the input vector of  $q$ . We have the following rules:

- *Rule 1'*: If  $q \downarrow B$  is unstable and  $qR_a q'$  for some  $q'$ , then add  $q'$  to  $Q_{IO}$  and  $(q, q')$  to  $R_{IO}$ .
- *Rule 2'*: If  $q \downarrow B$  is stable,  $s = q \downarrow B$ ,  $sTs^{(i)}$  is a transition in  $B$ , and  $u_i = x_j$  in the restriction of  $N$  to  $B$ , then add for each nonempty subset  $W$  of  $(U(q) \cup \{j\})$   $q^W$  to  $Q_{IO}$  and  $(q, q^W)$  to  $R_{IO}$ .

The induction step is applied until  $Q_{IO}$  and  $R_{IO}$  can no longer be enlarged.

Notice that, for constructing  $Q_{IO}$  and  $R_{IO}$ , we examine the stability of  $q \downarrow B$ , instead of  $q$ . The definition of input-output-mode realization (IO-realization for short) now reads as follows:

*Definition 3:* An initialized network  $(N, q_0)$  with restriction  $\downarrow$  is an IO-realization of behavior  $B$  iff  $(N, q_0)$  is IO-consistent and IO-deterministic with respect to  $B$ , and  $R_{IO}$  satisfies the condition:

$$R_{IO} \downarrow B - I = T,$$

where  $I$  is the identity relation on the set of states of  $B$  and IO-consistency and IO-determinism are defined like F-consistency and F-determinism, but now with respect to  $Q_{IO}$ .  $\square$

The conditions that  $(N, q_0)$  be IO-consistent and IO-deterministic with respect to  $B$  are introduced for the same reasons as in the case of F-realization.

*Example 10:* We can apply the definition of IO-realization to the behaviors and networks given in Examples 6, 7, and 8. The construction of  $Q_{IO}$  and  $R_{IO}$  happens to be exactly the same as for  $Q_F$  and  $R_F$ , in these three cases. We obtain the following conclusions:

- $(N_1, 000)$  is not an IO-realization of  $B_1$ , since  $(N_1, 000)$  is not IO-consistent with respect to  $B_1$ .
- $(N_2, 1000)$  is not an IO-realization of  $B_2$ , since  $(N_2, 1000)$  is not IO-deterministic with respect to  $B_2$ .
- $(N_3, 0010)$  is an IO-realization of  $B_3$ , since  $(N_3, 0010)$  is IO-consistent and IO-deterministic with respect to  $B_3$ , and  $R_{IO} \downarrow B_3 - I = T_3$ .  $\square$

*Example 11:* The situation is different for  $B_3$  and  $N_4$  from Example 9. Although  $(N_4, 0000)$  is an F-realization of  $B_3$ , it is not an IO-realization of  $B_3$ . The following sequence of transitions can occur in  $R_{IO}$ :

$$\begin{array}{cccccccc} 0000 & \xrightarrow{u} & 1000 & \xrightarrow{y_1} & 1100 & \xrightarrow{v} & 1101 & \xrightarrow{u} & 0101 & \xrightarrow{y_1} & 0001 \\ & & & & & & & & & & \xrightarrow{v} & 0000. \end{array}$$

From this sequence it follows that  $R_{IO} \downarrow B_3 - I \neq T_3$ , since  $(01, 00) \notin T_3$ .  $\square$

The property with respect to static and dynamic hazards also holds for IO-realizations.

*Property 4:* Property 2 also holds when  $(N, q_0)$  with restriction  $\downarrow$  is an IO-realization of simple deterministic behavior  $B$ .

*Proof:* The proof of Property 2 still holds if each occurrence of F is replaced by IO.  $\square$

We prove below that, if a network IO-realizes a behavior, then it also F-realizes it. Consequently, the fundamental mode of operation can be seen as a special case of the input-output mode of operation. Thus, if an input-output behavior is not realizable by any network operated in fundamental mode, then certainly it is not realizable by any network operated in input-output mode.

*Theorem 5:* If  $(N, q_0)$  with restriction  $\downarrow$  IO-realizes  $B$ , then  $(N, q_0)$  with the same restriction also F-realizes  $B$ .

*Proof:* Let  $(N, q_0)$  be an IO-realization of  $B$ . First, we prove that  $R_F \subseteq R_{IO}$  and then that  $(R_{IO} \downarrow B - I) \subseteq (R_F \downarrow B - I)$ . From these two properties it follows that  $(R_{IO} \downarrow B - I) = (R_F \downarrow B - I)$ . Furthermore, if  $R_F \subseteq R_{IO}$  and  $(N, q_0)$  is IO-consistent and IO-deterministic with respect to  $B$ , it follows that  $(N, q_0)$  is F-consistent and F-deterministic with respect to  $B$  as well. Consequently, we conclude that  $(N, q_0)$  is also an F-realization of  $B$ .

We prove  $R_F \subseteq R_{IO}$  by induction on the construction of  $R_F$  and  $R_{IO}$ .

*Basis:*  $\{q_0\} = R_F$  and  $\{q_0\} = R_{IO}$ .

*Induction Step:*  $R_F$  can be enlarged by applying Rule 1 or 2 to a state  $q \in Q_F$ .  $R_{IO}$  can be enlarged by applying Rule 1' or 2' to state  $q$ . Rule 1 applies when  $q$  is unstable. Suppose we can add  $(q, q')$  to  $R_F$ , because  $qR_a q'$ . Then we can also add  $(q, q')$  to  $R_{IO}$  by using Rule 1', if  $q \downarrow B$  is unstable, or by using Rule 2', if  $q \downarrow B$  is stable. Therefore, if  $R_F$  can be enlarged by Rule 1, then  $R_{IO}$  can be enlarged similarly.

If Rule 2 is used, then  $q$  is stable. Because  $(N, q_0)$  is IO-consistent with respect to  $B$ , and  $q$  is in  $R_{IO}$  by the induction hypothesis,  $q \downarrow B$  is also stable. Hence Rule 2' applies, and  $R_{IO}$  can be enlarged like  $R_F$  by choosing  $W$  to be  $\{j\}$ . This completes the proof of the first claim, i.e., we have

$$Q_F \subseteq Q_{IO} \text{ and } R_F \subseteq R_{IO}.$$

The proof of the second claim, i.e., that  $R_{IO} \downarrow B - I \subseteq R_F \downarrow B - I$ , is also done by induction on the construction of  $R_{IO}$  and  $R_F$ .

*Basis:* Obviously, the second claim holds for  $Q_{IO} = \{q_0\}$ ,  $R_{IO} = \emptyset$  and  $Q_F = \{q_0\}$ ,  $R_F = \emptyset$ .

*Induction Step:* We show that, for all  $s$  and  $s'$ ,

$$\begin{aligned} s \in (Q_F \downarrow B) \wedge (s, s') \in (R_{IO} \downarrow B - I) \\ \Rightarrow (s, s') \in (R_F \downarrow B - I). \end{aligned}$$

Let  $s \in Q_F \downarrow B$  and  $(s, s') \in R_{IO} \downarrow B - I$ . Then there is a  $q \in Q_F$  such that  $q \downarrow B = s$ . Since  $R_F \subseteq R_{IO}$ , also  $q \in Q_{IO}$ . Since  $(N, q_0)$  is IO-deterministic with respect to  $B$ , there is a  $q'$  such that  $\{q'\} = \text{out}(R_a, q)$ , where  $a$  is the input vector of  $q$ .

Suppose  $s$  is stable. Then all of the  $R_a$ -successors of  $q$  are in  $Q_{IO}$  because of Rule 2', and in  $R_F$  because of Rule 1. It follows from Property 4 that no changes occur in any  $(R_a \downarrow B)$ -sequence from  $q$  to  $q'$ . Consequently,  $q' \downarrow B = s$ . Since  $s$  is stable and  $B$  is simple deterministic,  $(s, s')$  must be an input transition in which only one input variable changes. Since  $q'$  is stable, we can apply Rule 2 and add  $(q', q'^{(i)})$  to  $R_F$ , where  $i$  is the index of the input that changes, i.e.,  $q'^{(i)} \downarrow B = s'$ . Consequently,  $(s, s') \in R_F \downarrow B - I$ .

If  $s$  is unstable then it follows by Property 4 that, in any  $(R_a \downarrow B)$ -sequence from  $q$  to  $q'$ , there exist two consecutive states the restrictions of which yield the output transition  $(s, s')$ . Since  $s$  is unstable and  $(N, q_0)$  is IO-consistent with respect to  $B$ ,  $q$  must also be unstable. By Rule 1, all the  $R_a$ -successors of  $q$  are in  $Q_F$ . From this it follows that  $(s, s') \in R_F \downarrow B - I$ . This completes the proof of the second claim.  $\square$

We say that an input-output behavior  $B$  has a *delay-insensitive* IO-realization  $(N, q_0)$  iff all wire delays are included as gate components of  $N$  and  $(N, q_0)$  IO-realizes  $B$ . Similarly we define a delay-insensitive F-realization of  $B$  by network  $(N, q_0)$ . For example, in the network  $N_4$  of Fig. 10, all wire delays are included as gate components. Consequently,  $(N_4, 0000)$  is a delay-insensitive F-realization of  $B_3$ . For a recent survey on other formalizations of delay-insensitive circuits, we refer the reader to [2].

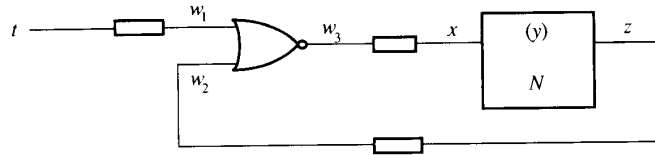
## VI. A BEHAVIOR NOT REALIZABLE IN THE INPUT-OUTPUT MODE

We will show in this section that the behavior  $B_3$  of Example 8 does not have a delay-insensitive realization operated in the input-output mode, although it has a delay-insensitive realization operated in the fundamental mode as we have seen in Example 9. We show that, if a delay-insensitive IO-realization  $(N, q_0)$  of  $B_3$  existed, then we could construct a network  $N'$  that would have contradictory properties when operated in fundamental mode. The proof uses a rather deep result concerning the equivalence of GMW analysis and ternary simulation. Ternary simulation uses the values 0, 1, and  $X$  and a partial order on these values such that  $X$  is greater than either one of the binary values. The concept of least upper bound is defined with respect to this partial order. Ternary simulation consists of two parts, called Algorithms A and B, which can be characterized as follows ([3], also [1] for Algorithm A):

*Theorem 6:* Let  $N$  be any network in which all wire delays are taken into account explicitly. Then the result of Algorithm A of the ternary simulation of a transition of  $N$  is equal to the least upper bound of the set of states that are reachable from the initial state in the GMW analysis of  $N$ . Furthermore, the result of Algorithm B of the ternary simulation is equal to the least upper bound of the set of states that appear in the outcome of the GMW analysis.  $\square$

For this section we assume that  $N$  has one input  $x$  and one gate variable  $z$  and we have a restriction with  $u = x$  and  $v = z$ . The remaining input variables of  $N$  are assumed to be constant, and will therefore be ignored. The remaining gate



Fig. 11. Network  $N'$ .

variables of  $N$  are represented by vector  $y$ , and we assume that the state of  $N$  can be represented by the vector  $xyz$ .

*Lemma 7:* The behavior  $B_3 = \langle 1, 1, S, T, 00 \rangle$ , where  $S$  and  $T$  are given by

$$B_3 : 00 \xrightarrow{u} 10 \xrightarrow{v} 11 \xrightarrow{u} 01,$$

does not have a delay-insensitive IO-realization.

*Proof:* If gate network  $(N, q_0)$  is a delay-insensitive IO-realization of behavior  $B_3$ , then it must have the following properties:

- $P_1$  There exists an initial stable state  $q_0 = 0b0$  for some vector  $b$ .
- $P_2$  The state  $1b0$  is unstable (since  $(N, q_0)$  is IO-consistent with respect to  $B_3$  and  $10$  is unstable in  $B_3$ ).
- $P_3$  In every  $R_1$ -sequence starting with  $1b0$  and ending with a state in  $out(R_1, 1b0)$ ,  $z$  changes exactly once (Property 4).
- $P_4$  Let  $1c1$  be any state that can be reached by an  $R_1$ -sequence from  $1b0$ . If the input  $x$  is changed to 0 again—some of the variables in  $y$  may also change at the same time—some state  $0d1$  is reached. If the GMW analysis is continued from state  $0d1$ , the gate variable  $z$  should not change. Note that state  $1c1$  does not have to be a stable network state.

Consider the network  $N'$  derived from  $N$  as shown in Fig. 11.

Notice that a delay element is introduced for every connection wire. Since network  $N$  also contains a delay element for each connection wire, Theorem 6 is applicable to network  $N'$ .

One verifies that the state  $twxyz = 11000b0$  is stable. We will operate  $N'$  in fundamental mode, causing a transition by changing  $t$  from 1 to 0 and then letting the network stabilize. We have the following  $R_0$ -sequence for  $N'$ :

$$01000b0 \xrightarrow{w_1} 00000b0 \xrightarrow{w_3} 00010b0 \xrightarrow{x} 00011b0.$$

Note that, until the last step above,  $N$  has been stable as guaranteed by  $P_1$ . In the last step,  $N$  becomes unstable as required by  $P_2$ . By  $P_3$ ,  $N$  eventually reaches a state  $1c1$ , for some vector  $c$ , by a subsequent  $R_0$ -sequence for  $N'$ , i.e.,

$$00011b0 \longrightarrow^* 00011c1,$$

where  $\longrightarrow^*$  is the transitive closure of  $\longrightarrow$ . From  $P_4$  it now follows that  $z$  cannot change any more, even if  $x$  becomes 0 again; this has to hold for all possible values that  $y$  may reach.

Thus, the  $y$ -component of the state of  $N'$  becomes irrelevant, and we replace it by “ $\_$ ” from now on. We have the following extension of the  $R_0$ -sequence for  $N'$ :

$$00011c1 \xrightarrow{w_2} 00111\_1 \xrightarrow{w_3} 00101\_1 \xrightarrow{x} 00100\_1.$$

In the last state, the variables  $w, x$ , and  $z$  are stable and will not become unstable again. From the above, it follows that the outcome of the GMW analysis of  $N'$  started in state  $01000b0$ , where the input is kept at 0, always yields states of the form  $00100\_1$ , i.e.,

$$q \in out(R_0, 01000b0) \Rightarrow \text{the } z \text{ component of } q \text{ is } 1.$$

Consequently, even in the presence of arbitrary gate and wire delays, the final outcome of the transition yields  $z = 1$ .

We also observe that, in the above analysis,  $N'$  is operated in fundamental mode with respect to behavior

$$B' : 00 \xrightarrow{t} 10 \xrightarrow{z} 11,$$

but  $N$  is operated in input-output mode with respect to behavior  $B_3$ .

Next we show that ternary simulation of  $N'$  contradicts the conclusion reached above. The reader unfamiliar with ternary simulation is referred to [3]; here we only give the results of the simulation.

Algorithm A of the simulation produces the following initial sequence:

$$X1000b0 \longrightarrow XX000b0 \longrightarrow XX0X0b0 \longrightarrow XX0XXb0.$$

As we have seen above,

$$01000b0 R_0^* 01000b0 \text{ and } 01000b0 R_0^* 00011c1,$$

i.e., both  $01000b0$  and  $00011c1$  are reachable from  $01000b0$  (in zero or more steps). Consequently, the output  $z$  can take the values 0 and 1 in the GMW analysis of the network. But then, by Theorem 6, Algorithm A of the ternary simulation must produce  $z = X$ . Subsequently,  $w_2$  becomes  $X$ , and the final result of Algorithm A has the form  $XXXXXeX$  for some vector  $e$  of ternary values.

Changing  $t$  from  $X$  to 0 now and applying Algorithm B to the state  $0XXXXXeX$ , we find that the algorithm terminates in the second step with the state  $00XXXXeX$ . Consequently, Algorithm B predicts that  $z$  has the value  $X$ . But then, by Theorem 6, there exists a state in the outcome of the GMW analysis where  $z = 0$ . This contradicts the GMW analysis

above. Therefore, the network  $N$  with the postulated properties cannot exist, and we have proved that behavior  $B_3$  does not have a delay-insensitive gate realization operated in the input-output mode.  $\square$

With the above lemma it is easy to verify that the input-output behaviors of the set-reset latch, the C-ELEMENT, and the TOGGLE of Figs. 1, 2, 3, and 4 do not have delay-insensitive realizations operated in input-output mode. Each of these behaviors contains behavior  $B_3$  of Lemma 7. The latch has the following subbehavior:

$$000 \xrightarrow{u_1} 100 \xrightarrow{v} 101 \xrightarrow{u_1} 001.$$

Thus, simply by ignoring the input  $u_2$ , we obtain the behavior  $B_3$ . Similarly, the C-ELEMENT has the subbehavior:

$$100 \xrightarrow{u_2} 110 \xrightarrow{v} 111 \xrightarrow{u_2} 101.$$

If we ignore the input  $u_1$ , we obtain behavior  $B_3$ . Finally, the TOGGLE contains the behavior

$$000 \xrightarrow{u} 100 \xrightarrow{v_1} 110 \xrightarrow{u} 010 \xrightarrow{v_2} 011$$

and we obtain  $B_3$  by ignoring the second output.

By means of slight modifications in the proof of Lemma 7, we can show that three other behaviors also lack delay-insensitive IO-realizations.

*Lemma 8:* Any behavior  $B = \langle 1, 1, S, T, s_0 \rangle$ , where  $S$  and  $T$  are given by

$$B : ab \xrightarrow{u} \bar{a}b \xrightarrow{v} \bar{a}\bar{b} \xrightarrow{u} a\bar{b}$$

with  $a, b \in \{0, 1\}$ , does not have a delay-insensitive gate realization operated in input-output mode.

*Proof:* In case  $ab = 10$ , repeat the arguments of Lemma 7, but with network  $N'$  modified as follows. Insert an inverter in series with a delay in the wire leading to the input  $x$  of network  $N$ . In case  $ab = 01$ , modify network  $N'$  by the addition of an inverter in series with a delay in the wire leaving output  $z$  of network  $N$ . In case  $ab = 11$ , modify network  $N'$  by the addition of two inverters with delays as indicated in the two cases above.  $\square$

In the following section we show that a larger class of behaviors does not have delay-insensitive gate realizations operating in the input-output mode.

## VII. NONTRIVIAL SEQUENTIAL BEHAVIORS

An example of a simple deterministic input-output behavior that does have a delay-insensitive IO-realization is shown below.

$$01 \xrightarrow{u} 11 \xrightarrow{v} 10 \xrightarrow{u} 00 \xrightarrow{v} 01.$$

This behavior is realizable by an inverter with input  $u$  and output  $v$ . The behavior is rather trivial, however, since every input vector uniquely determines the state of the network.

Another rather trivial input-output behavior that has a delay-insensitive IO-realization is given below, where either 00 or 01 is the initial state.

$$00 \xrightarrow{u} 10 \xrightarrow{u} 00 \quad 01 \xrightarrow{u} 11 \xrightarrow{u} 01.$$

This behavior can be realized by an output that is connected to a constant input and a “dangling” external input  $u$ . Here there are two different stable states possible for each input value; however, these two states are not connected.

In order to eliminate such trivial cases, we impose the following condition on *nontrivial* simple deterministic behaviors: if a behavior  $B = \langle h, k, S, T, s_0 \rangle$  is nontrivial, then there exists at least one input vector  $u = a$  for which there are at least two stable states  $ab$  and  $ab'$  where  $b \neq b'$  and  $ab'$  is reachable from  $ab$ . Notice that the two input-output behaviors above do not satisfy the above condition.

We have the following result:

*Theorem 9:* No nontrivial simple deterministic behavior  $B = \langle 1, k, S, T, s_0 \rangle$  with a binary input has a delay-insensitive gate realization operating in input-output mode.

*Proof:* If  $B$  is nontrivial, then there exist stable states  $ab$  and  $ab'$  such that  $ab'$  is reachable from  $ab$ . Since  $b \neq b'$ , they must differ in at least one component. Without loss of generality, assume that they differ in their last component, i.e., that  $b = cd$  and  $b' = c'd$ , where  $d \in \{0, 1\}$ .

In case  $ad = 00$ , we can apply the following reasoning. Start in state  $0c0$  which is stable. In order for  $0c'1$  to be reachable from  $0c0$ , we must change  $u$  to 1 and then back to 0 some finite number of times. At some point in this sequence we must have a state  $0e0$  where the output has not yet changed, but from which we can reach state  $0h1$  with two input changes. Thus, we must have the subbehavior:

$$0e0 \xrightarrow{u} 1e0 \xrightarrow{u} 1fg \xrightarrow{u} 0fg \xrightarrow{u} 0h1.$$

We can now consider two subcases.

Case 1: If  $g = 1$  then the above sequence projects to the behavior  $B_3$ , if we ignore all but the first and the last components. By Lemma 7,  $B$  cannot be realized.

Case 2:  $g = 0$ . We now have the following projection:

$$00 \xrightarrow{u} 10 \xrightarrow{u} 00 \xrightarrow{u} 01,$$

where the state 10 is stable in the behavior  $B$ , because the output is not changing (by assumption). By Theorem 4.3 of [18], this behavior is not realizable by any network even if it is operating in fundamental mode. The basic result from [18] states that if an output does not change during the first input change, then it will not change during any number of subsequent input changes in any network that operates properly in fundamental mode and is free of static hazards in the outputs. The other cases, where  $ad = 10$  or  $ad = 01$  or  $ad = 11$  are all handled similarly using Lemma 8.  $\square$

## VIII. CONCLUDING REMARKS

We have presented formal characterizations of the two modes of operation of a network of basic elements: the fundamental mode and the input-output mode. In doing so, we have used the General Multiple Winner Model for representing

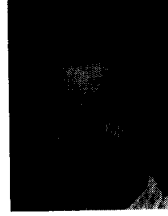




**Janusz A. (John) Brzozowski (M'87)** received the B.A.Sc. and M.A.Sc. degrees in electrical engineering from the University of Toronto in 1957 and 1959, respectively, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University in 1962.

He was Assistant Professor from 1962 to 1965 and Associate Professor from 1965 to 1967 in the Department of Electrical Engineering, University of Ottawa. Since 1967 he has been a Professor in the Department of Computer Science, University of Waterloo, Waterloo, Ont., Canada. In the periods 1978–1983 and 1987–1989 he was chair of that department. He has had visiting appointments at the University of California, Berkeley (1965–1966), University of Paris (1974–1975), University of São Paulo (1983), Kyoto University (1984), and Eindhoven University (1989–1990). He has published many papers in the areas of algebraic theory of regular languages, finite automata, asynchronous circuits, and testing. He is also co-author of *Digital Networks* (Englewood Cliffs, NJ: Prentice-Hall, 1976). His present research interests include asynchronous circuits, delay-insensitive design, VLSI models, testing, automata, and formal languages.

Dr. Brzozowski is a member of the ACM, EATCS, and the Association of Professional Engineers of Ontario.



**Jo C. Ebergen** received the Master's degree in Mathematics from Eindhoven University of Technology in 1983. In 1987, he received the Ph.D. degree from Eindhoven University of Technology.

From 1983 until 1987 he worked as a researcher at the Centre for Mathematics and Computer Science in Amsterdam in the area of VLSI design. Currently, he is Assistant Professor at the University of Waterloo, Waterloo, Ont., Canada. His main research interests are programming methodology, parallel computations, and delay-insensitive circuit design.

Dr. Ebergen is a member of ACM and EATCS.