Printing Requisition / Graphic Services

1178

1. **Please complete unshaded areas** on form as applicable.
2. **Distribute copies** as follows: White and Yellow to Graphic Services. **Retain Pink Copies for your records.**
3. On completion of order the **Yellow copy will be returned** with the printed material.
4. **Please direct enquiries, quoting requisition number and account number,** to **extension 3451.**

TITLE OR DESCRIPTION: CS-87-63

DATE REQUISITIONED: November 13/87

DATE REQUIRED: ASAP

ACCOUNT NO.: 1 2 6 6 3 1 8 4 1

REQUISITIONER – PRINT: J. Brzozowski

PHONE: 2192

SIGNING AUTHORITY: S. DeAngelis / J. Brzozowski

MAILING INFO – NAME: S. DeAngelis

DEPT.: CS

BLDG. & ROOM NO.: MC 6081E

☒ DELIVER ☐ PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 27

NUMBER OF COPIES: 100

TYPE OF PAPER STOCK: ☐ BOND ☐ NCR ____ PT. ☐ COVER ☐ BRISTOL ☐ SUPPLIED ☐ Alpac Ivory 140M

PAPER SIZE: ☐ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17 ☐ 10x14 Glosscoat 10 pt Rolland Tint

PAPER COLOUR: ☐ WHITE ☒ ____ INK: ☒ BLACK ☐ ____

PRINTING: ☐ 1 SIDE ____ PGS. ☒ 2 SIDES ____ PGS. NUMBERING FROM ____ TO ____

BINDING/FINISHING: ☒ COLLATING ☐ STAPLING ☐ ____ HOLE PUNCHED ☐ PLASTIC RING

FOLDING/PADDING: 7x10 saddle stitched CUTTING SIZE

Special Instructions

Beaver Cover

Both cover and inside in black ink

please

COPY CENTRE — OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP — OPER. NO. | TIME | LABOUR CODE

D 0 1

D 0 1

D 0 1

TYPESETTING — QUANTITY

P A P 0 0 0 0 0 0 — T 0 1

P A P 0 0 0 0 0 0 — T 0 1

P A P 0 0 0 0 0 0 — T 0 1

PROOF

P R F

P R F

P R F

NEGATIVES — QUANTITY | OPER. NO. | TIME | LABOUR CODE

F L M — C 0 1

F L M — C 0 1

F L M — C 0 1

F L M — C 0 1

F L M — C 0 1

PMT

P M T — C 0 1

P M T — C 0 1

P M T — C 0 1

PLATES

P L T — P 0 1

P L T — P 0 1

P L T — P 0 1

STOCK

0 0 1

0 0 1

0 0 1

0 0 1

BINDERY

R N G — B 0 1

R N G — B 0 1

R N G — B 0 1

M I S 0 0 0 0 0 0 — B 0 1

OUTSIDE SERVICES

$ ____ COST

TAXES – PROVINCIAL ☐ FEDERAL ☐

GRAPHIC SERV. OCT. 85 482-2

UNIVERSITY OF WATERLOO
COMPUTER SCIENCE DEPARTMENT

On the Existence of Speed-Independent Circuits

C-J. Seger

Research Report
CS-87-63

November, 1987

On the Existence of Speed-Independent Circuits†

C-J. Seger

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

A circuit is called *speed-independent* if its nontransient behavior does not depend on the size of the delays in the different components of the circuit. We show that the class of speed-independent circuits is relatively small — in fact, very many useful circuits cannot be realized in a speed-independent design. For example, we show that there does not exist any speed-independent mod-k counter for any $k>1$. The results are derived using a very general model of a network which is applicable to both gate circuits and more modern MOS switch-level circuits. Furthermore, the results are also robust with respect to different delay assumptions and definitions of speed-independence.

1. Introduction

The ideas to be discussed will be introduced by means of some examples. Consider the gate circuit G_1 of Fig. 1. Assume that the gates can have arbitrarily large, but finite, inertial delays, and that the wires are delay-free. Assume also that any transition between 0 and 1 or 1 and 0 is instantaneous. The total state $x=0$, $y=(y_1,y_2,y_3)=(0,1,1)$ (we will write 011 for short) is stable, i.e. the excitation of each gate is equal to the current value of the gate output, and thus there is no tendency to change state. It is easy to verify that, if the input changes to $x=1$, the network will eventually end up in the new stable state $y=101$. Since this state is the only nontransient state reachable no matter what the gate delays are, we say that the transition is speed-independent. On

† This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant A0871, by a scholarship from the Institute for Computer Research, University of Waterloo, and by an Ontario Graduate Scholarship.

the other hand, consider the gate circuit G_2 of Fig. 2. If this circuit is started in the stable state $x=0$, $y=(y_1,y_2,y_3)=100$ and the input changes to 1, the circuit can end up in either the state 000 or 001 depending on the relative sizes of the delays in gates 1 and 2. In fact, gates 1 and 2 both become unstable after the input changes, i.e. these two gates are involved in a "race". If gate 1 changes first, the outcome is 000; if gate 2 is faster, state 001 can also be reached. Since the nontransient outcome of this transition depends on the internal delays in the circuit, the transition is not speed-independent.

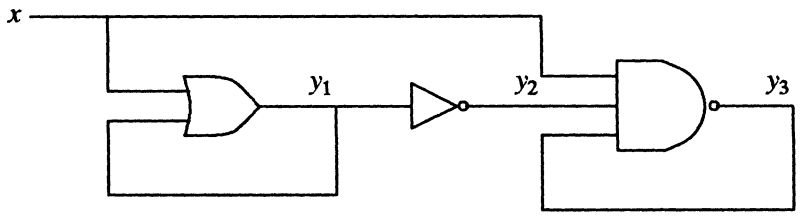

Figure 1. Gate circuit G_1.

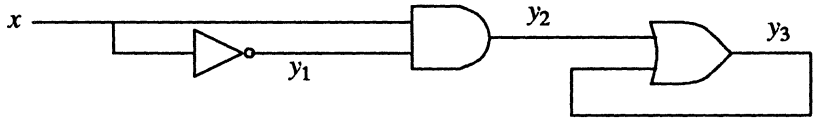

Figure 2. Gate circuit G_2.

It is important to note that the classification of transitions according to speed-independence is very sensitive to the race model chosen. Consider, for example, the gate circuit G_3 of Fig. 3. Assume the circuit is started in the stable state $x=0$, $y=(y_1,y_2,y_3,y_4)=1000$ and that the input changes to $x=1$. It is easy to verify that this transition is speed-independent if only the gates have delays associated with them. However, if the wire between the first inverter (gate 1) and the AND gate (gate 3) also has a delay, the transition is no longer speed-independent, since the network can then end up in either the state $y=0100$ or the state $y=0101$. The latter state can be reached when the wire delay between gates 1 and 3 exceeds the delay of gate 2.

The first study of speed-independent circuits was done by Muller and Bartky in the late 1950's [9, 11]. They assumed that only the gates have delays associated with them, that all transitions are instantaneous,

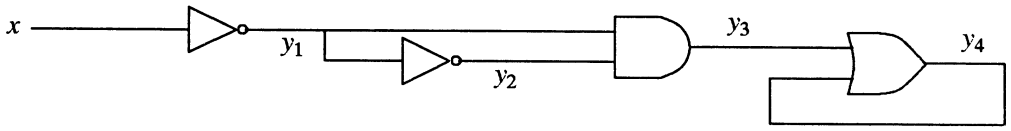

Figure 3. Gate circuit G_3.

and that the circuits do not have any inputs. Using these assumptions, they developed a theory for such speed-independent circuits. For more details, the reader is referred to [9, Chapter 10] which contains a thorough treatment of this theory. One of the main differences between Muller and Bartky's work and ours is the fact that they developed a theory for circuits that do not have any input signals, whereas we are interested in the response of a circuit to a sequence of input changes.

With the arrival of MOS VLSI circuits, the idea of speed-independence has again become of interest [7, 10, 14], because delays can be substantial on a VLSI chip, and it is very difficult (and expensive) to correct a circuit that contains some timing problem. Hence, the idea of designing the circuit in such a way that it is guaranteed to work correctly, independently of the sizes of the delays, sounds very attractive.

One of the problems with speed-independent design is the difficulty of verifying that a given circuit really is speed-independent. For many delay models the problem of determining the behavior of a circuit is NP-hard [6, 12] or even PSPACE-complete [13] and thus intractable for anything but trivial circuits. Fortunately, there do exist reasonably realistic race and delay models in which the outcome of a transition can be determined in polynomial time. Several such models were discussed in [3]. In particular, the extended multiple winner (XMW) model was defined and characterized. The XMW model assumes that delays can be arbitrary, but finite. Furthermore, in contrast to more classical binary race models, the XMW model allows changes to be relatively slow and to go through an intermediate value $\times$.

The main contributions of this paper are the following results. First we define an abstract sequential machine that summarizes the speed-independent behavior of a circuit. Second, we show how a sequential circuit can be converted into an equivalent combinational circuit, in the sense that ternary simulation gives the same results for both circuits. Third, the following basic properties of speed-independent circuits are derived:

1) Any odd number of changes of the same set of inputs must leave the network in the same state.

2) If a multiple-input change is speed-independent, then the circuit must end up in the same state if this multiple-input change is made "step-by-step".

3) Suppose some set of inputs changes. If an output node does not change value for this input change, and there is no hazard on this node for this transition, then that output node will keep the same value for any number of changes of that same set of inputs.

Unfortunately, these properties are quite restrictive. Using these necessary conditions, we show that the class of speed-independent circuits is quite small. For example, we show that it is not possible to design a speed-independent mod-2 counter, i.e. a circuit whose output changes with half the frequency of its input signal. Finally, we show that the above results also hold for quite different delay assumptions and definitions of speed-independence.

Recently, Ebergen [4] showed that a very large class of circuits (including mod-2 counters) could be realized in a speed-independent design if a small set of basic "building blocks" could be designed speed-independently. Unfortunately, our results show that such building blocks do not exist. The reason that there do not exist speed-independent designs for many functions is the "pessimism" of the assumptions of speed-independence. In particular, it is not very realistic to assume that delays can be arbitrary large. In most cases it is quite reasonable to give some bounds for the delays in a circuit. This would lead to circuits and circuit designs that can be said to be *speed-insensitive*, rather than speed-independent. We think that such a concept is much more practical. Unfortunately, it is quite difficult to analyze the behavior of a circuit under such bounded delay assumptions. This problem must be solved before speed-insensitive circuits can be designed and verified reliably.

The paper is structured as follows. In Section 2 we establish a general framework for describing both classical gate circuits and more modern MOS switch-level circuits. The basic race model used is discussed in Section 3. An efficient algorithm for computing the outcome of a transition in this race model is described in Section 4. The material in Sections 2-4 has been described in detail elsewhere [3], but a summary of the main results has been included to make this paper self-contained. An abstract machine, called a fundamental mode speed-

independent asynchronous machine, is defined in Section 5 in order to summarize the speed-independent behavior of a network. A useful trick to transform an arbitrary sequential network into a combinational network is described in Section 6. In Section 7 we derive some general and, as it turns out, very restrictive properties of speed-independent circuits. In Section 8 we show that the results obtained in Section 7 are fairly robust with regard to different delay and race models. Finally, in Section 9 we discuss some ramifications the results may have and also point out some areas of further research.

2. Network Model

A rather general concept of a network is introduced in this section. As was shown in [3], this model provides a common framework for representing both gate networks and switch-level MOS networks. For a more comprehensive discussion of this framework and a description of how gate circuits and switch-level MOS circuits can be modeled using the framework, the reader is referred to [3]. We will use the convention that x, y, and z denote vectors of state variables, whereas x_i, y_i, and z_i denote single components of the vectors. Similarly, a, b, c, and d denote particular constant vectors of state variables, and a_i, b_i, c_i, and d_i denote their components.

Let $\{0,1\}$ be the set of the two usual binary values, and let $T=\{0,1,\times\}$. The symbol $\times$ will be used to denote an unknown or intermediate value.

A *network* N is a finite directed labeled graph $N=<V,E,x,y,Y>$, where

$V=\{1,\ldots,m\}$ is a set of vertices,

$E\subseteq V\times V$ is a set of edges,

$x=(x_1,\ldots,x_n)$, $n\leq m$, is a vector of *input variables* taking values from T,

$y=(y_1,\ldots,y_m)$ is a vector of *vertex variables* taking values from T,

$Y=(Y_1,\ldots,Y_m)$ is a vector of ternary *excitation functions*.

Vertices $1,\ldots,n$ are all of indegree 0, and are called *input vertices*. Vertices $n+1,\ldots,m$ are *function vertices* and are all of indegree ≥ 1. The excitation function of an input vertex j is the function $Y_j:T\rightarrow T$ defined simply as $Y_j=x_j$. For a function vertex j the excitation function is a ternary function $Y_j:T^{d_j}\rightarrow T$, where d_j denotes the indegree of vertex j. An edge $(i,j)\in E$ shows that Y_j is a function of y_i. Thus, for a function vertex, Y_j depends only on some subset of $\{y_1,\ldots,y_m\}$. The ordered pair $\langle x,y\rangle$, $x\in T^n$, $y\in T^m$, is called the *total state* of N. For notational

convenience, we will treat an excitation function Y_j as a function of the total state of N, i.e. we will write $Y_j(x,y)$†. The vertex variable y_j is interpreted as the present state of a vertex, whereas the excitation function $Y_j(x,y)$ computes the value to which the vertex is trying to change, when the present input is x and the present state is y.

If $y_i = Y_i(x,y)$ then vertex i is *stable*; otherwise it is *unstable*. A given total state $\langle x,y\rangle$ is *stable* if each vertex is stable. A network will remain in a stable total state indefinitely, unless the input changes, in which case the state becomes unstable. If there are two or more vertices that are unstable in a total state, we say that there is a *race*. In general, there may be several possible successor states for a given unstable state. This set of possible successor states depends on the race model used, as elaborated in the next section.

3. Race Model

The underlying idea of speed-independence is that a circuit should exhibit the same behavior, independently of the sizes of the delays in the circuit. For this reason we will analyze a circuit using the extended multiple winner (XMW) race model. For a more complete description of the XMW model, the reader is referred to [3].

There are three basic ideas behind the XMW model. First, it is assumed that the inputs remain fixed after each change until the network has a chance to "stabilize". This corresponds to the fundamental-mode operation assumption of [8]. Second, the past history is completely ignored, in the sense that all unstable vertices have the same chance of "winning" a race no matter when they entered the race. Third, any unstable vertex with a binary present value may take on an intermediate value $\times$. We define the XMW model more formally as follows:

Define the partial order $\sqsubseteq$ on T as follows: $t_i \sqsubseteq t_i$ for all $t_i \in T$, $0 \sqsubseteq \times$, and $1 \sqsubseteq \times$. The partial order is extended to T^r, $r \geq 1$, in the obvious way: $s \sqsubseteq t$ iff $s_i \sqsubseteq t_i$ for $1 \leq i \leq r$. Also, the partial order is extended to ordered pairs in the natural way: $\langle \tilde{a}, \tilde{b}\rangle \sqsubseteq \langle a,b\rangle$ iff $\tilde{a} \sqsubseteq a$ and $\tilde{b} \sqsubseteq b$. For $s,t \in T^r$, $r \geq 1$, we write $s \sqsubset t$ when $s \sqsubseteq t$ and $s \neq t$. The least upper bound, denoted *l.u.b.*, on the partial order $\sqsubseteq$ is defined as usual.

---

† Strictly speaking we should write $Y_j(\langle x,y\rangle)$, but the angle brackets are omitted to improve readability.

Intuitively, $\times$ is used to denote an "unknown" or "intermediate" value. Thus $s \sqsubset t$ indicates that s has less "uncertainty" (more binary values) than t.

The following fundamental assumption is made about the excitation function of any network N:

$$\langle a,b\rangle \sqsubseteq \langle c,d\rangle \quad \text{implies} \quad Y(a,b) \sqsubseteq Y(c,d).$$

This is a monotonicity property of the excitation function that is consistent with our use of the value $\times$. Basically, if the total state is more uncertain, the excitation cannot become less uncertain.

For any $a \in T^n$ and $b \in T^m$, define $U(a,b)$ to be the set of unstable vertices in b, i.e.

$$U(a,b) = \{i : 1 \leq i \leq m, \text{ and } b_i \neq Y_i(a,b)\}.$$

The XMW relation R_a on the set T^m defines the set of successors for any total state $\langle a,b\rangle$, $a \in T^n$, $b \in T^m$, as follows. If b is stable, i.e. if $U(a,b) = \varnothing$, then the only possible successor is b, i.e. $b R_a b$. Otherwise, let $b R_a \bar{b}$, for any $\bar{b}$ such that:

1) $\bar{b} \neq b$, and
2) $\bar{b}_i \in \{b_i, Y_i(a,b), l.u.b.\{b_i, Y_i(a,b)\}\}$ $1 \leq i \leq m$.

No other pairs are related by R_a.

For any input vector $a \in T^n$ and any state $b \in T^m$, define the set $\text{cycl}(R_a,b)$ to be the set of total states of N that appear in cycles in the relation R_a and are reachable from b. Note that each stable state reachable from b is in $\text{cycl}(R_a,b)$. Formally,

$$\text{cycl}(R_a,b) = \{c \in T^m : b R_a^* c \text{ and } c R_a^+ c\},$$

where R_a^+ is the transitive closure of R_a, and R_a^* is the reflexive transitive closure of R_a.

The concept of a transient cycle is introduced in order to capture the fact that delays cannot be infinite. A cycle is called *transient* if there exists a vertex v which is unstable in all of the states in the cycle, has the same value in all these states, and either that value is binary or the excitation of v is the same in all these states. If a cycle is not transient, it is called *nontransient*. Let

$$\text{out}(R_a,b) = \{c \in \text{cycl}(R_a,b) : c \text{ appears in a nontransient cycle}\}.$$

The set $\text{out}(R_a,b)$ is the *outcome* of the XMW analysis of the behavior of

N when started in total state b, in the sense that it consists of all the states N can be in, under nontransient conditions.

The reader should note that $\text{out}(R_a, b) \neq \emptyset$. This follows from the fact that the sequence of states obtained by changing all unstable vertices to their excitations in every state of the sequence must eventually lead to a cycle. This cycle of states must be nontransient by definition, and thus all such states appear in $\text{out}(R_a, b)$.

The following property of the XMW relation will be used later. Assume the binary input vectors $\hat{a}$ and a differ in at least two components, and that $\tilde{a}$ is a binary input vector "between" $\hat{a}$ and a (i.e. that $\tilde{a}_j$ is either equal to $\hat{a}_j$ or a_j for $j = 1,\ldots,n$, but $\tilde{a} \neq \hat{a}$ and $\tilde{a} \neq a$). If a network N is started in the stable total state $\langle \hat{a}, b \rangle$, and the input is first changed to $\tilde{a}$ and then later to a, the proposition states that the outcome after this second change is contained in the outcome obtained when the input is changed immediately from $\hat{a}$ to a. More formally:

Proposition 1 Let $\hat{a}$, $\tilde{a}$, $a \in \{0,1\}^n$ be three input vectors such that $\tilde{a} \neq \hat{a}$, $\tilde{a} \neq a$, and $\tilde{a} \sqsubseteq l.u.b.\{\hat{a}, a\}$. Furthermore, assume that $\langle \hat{a}, b \rangle$ is a stable total state of N. Then

$$\{d : d \in \text{out}(R_a, c), \text{ where } c \in \text{out}(R_{\tilde{a}}, b)\} \subseteq \text{out}(R_a, b).$$

Proof: We prove the following stronger version of the claim:

$$\{d : d \in \text{out}(R_a, c), \text{ where } b R_{\tilde{a}}^* c\} \subseteq \text{out}(R_a, b).$$

Proposition 1 then follows directly from the fact that $\text{out}(R_{\tilde{a}}, b) \subseteq \{c : b R_{\tilde{a}}^* c\}$.

The proof consists of two steps. First we show that if $b R_{\tilde{a}}^* c$, then $b R_a^* c$. Second we show that if $b R_a^* c$, then $\text{out}(R_a, c) \subseteq \text{out}(R_a, b)$. Together, these give that

$$\{d : d \in \text{out}(R_a, c), \text{ where } b R_{\tilde{a}}^* c\} \subseteq \{d : d \in \text{out}(R_a, c), \text{ where } b R_a^* c\} \subseteq \text{out}(R_a, b)$$

establishing the claim.

The first property follows from the following observations. First, since $\langle \hat{a}, b \rangle$ is a stable total state of N, we can conclude that $(b_1,\ldots,b_n) = \hat{a}$. Furthermore, by the definition of $\tilde{a}$, $\tilde{a}_j$ is either equal to $\hat{a}_j$ or a_j. Hence, it follows that $U(\tilde{a}, b) \subset U(a, b)$ and that $Y_j(\tilde{a}, b) = Y_j(a, b)$ for all $j \in U(\tilde{a}, b)$. Since only unstable vertices can change according to the XMW relation, it is easy to prove by induction on k that: if $b R_{\tilde{a}}^k d$ then $b R_a^k d$, $U(\tilde{a}, d) \subset U(a, d)$ and that

$Y_j(\tilde{a},d)=Y_j(a,d)$ for all $j\in U(\tilde{a},d)$. We leave the details of the proof to the reader.

The second property, i.e. that if bR_a^*c then $\text{out}(R_a,c)\subseteq\text{out}(R_a,b)$, follows immediately from the definition of out. □

The XMW model permits us to predict the outcome of any input change under very general assumptions about delays in a network. In fact each vertex may have an arbitrary finite inertial delay. It is natural to define the concept of speed-independence in terms of a network's behavior according to an XMW analysis.

Definition 1 A network N, started in a binary stable total state $\langle\hat{a},b\rangle$ and with a new binary input vector a, is said to be *speed-independent with respect to* $\langle a,b\rangle$ if $\text{out}(R_a,b)$ contains a single binary state.

The reader should note that, by the definition of out, the single state in $\text{out}(R_a,b)$ must be a stable state. Note also that the term speed-independence is defined with respect to a certain starting configuration. We will later extend this idea to cover the general behavior of a network.

The above definition differs from Muller and Bartky's original definition of speed-independence [9] in three ways. First, the underlying race model is somewhat different, in that the XMW model includes an intermediate value $\times$, whereas their model is a classical binary race model. Second, our definition excludes transitions that can cause the network to enter a nontransient oscillation. We will return to these differences in Section 8. Finally, our network model is different in that we include inputs to the circuit, whereas Muller and Bartky's model does not. In fact, the characterization of the behavior of a speed-independent circuit in response to a sequence of input changes is the main problem studied in this paper.

4. Ternary Simulation

The XMW race model, though conceptually simple and relatively natural, is computationally intractable; in the worst case the graph of the relation R_a may have $O(3^m)$ vertices. Fortunately, there exists an efficient algorithm computing essentially the same information. This simulation procedure, due to Eichelberger [5], consists of Algorithms A and B described below. For more details of the procedure and proofs of the claims below, the reader is referred to [3].

Let N be a network started in the stable total state $\langle \hat{a}, b \rangle$ and with a new input vector a. Furthermore, let $\mathbf{a} = l.u.b.\{\hat{a}, a\}$. Algorithm A is defined by:

Algorithm A
h:=0;
$\mathbf{y}^0$:=b;
repeat
  h:=h+1;
  for $i = 1$ *to* m
    $\mathbf{y}_i^h = Y_i(\mathbf{a}, \mathbf{y}^{h-1})$;
until $\mathbf{y}^h = \mathbf{y}^{h-1}$;

In the following we will use A (B) to denote the name of the algorithm, and A (B) to denote the length of the sequence of states produced by Algorithm A (B).

Proposition 2 Algorithm A produces a finite sequence $\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^A$ of states, where $A \leq m$, and $\mathbf{y}^h \sqsubset \mathbf{y}^{h+1}$ for $0 \leq h < A$.

Algorithm B is defined next:

Algorithm B
h:=0;
$\mathbf{z}^0$:=$\mathbf{y}^A$;
repeat
  h:=h+1;
  for $i = 1$ *to* m
    $\mathbf{z}_i^h = Y_i(a, \mathbf{z}^{h-1})$;
until $\mathbf{z}^h = \mathbf{z}^{h-1}$;

Proposition 3 Algorithm B produces a finite sequence $\mathbf{z}^0, \mathbf{z}^1, \ldots, \mathbf{z}^B$ of states, where $B \leq m$, and $\mathbf{z}^h \sqsupset \mathbf{z}^{h+1}$ for $0 \leq h < B$.

The following two theorems are of vital importance for the remaining part of this paper.

Theorem 1 The result $\mathbf{y}^A$ of Algorithm A is the least upper bound of all the states reachable from the initial state b according to the XMW race model, i.e. $\mathbf{y}^A = l.u.b.\{c \in T^m : b R_a^* c\}$.

Theorem 2 The result $\mathbf{z}^B$ of Algorithm B is the least upper bound of all the nontransient cyclic states reachable from b according to the XMW race model, i.e. $\mathbf{z}^B = l.u.b.\ \mathrm{out}(R_a, b)$.

The above theorems dealt with the nontransient behavior of a given network. However, the network being analyzed is often a part of a larger system. In such cases, some subset of the vertices may be

"visible" to the rest of the system. We will call such vertices *output vertices*. When output vertices are present, there is a new problem. Consider, for example, an output vertex that has the value 0 initially, and also in all the states of out(R_a, b). It is quite possible that the vertex has the value 1 or $\times$ in some of the transient states during the transition. Such short pulses must be detected, since they may cause unwanted state changes in the rest of the system controlled by this output vertex.

As usual, assume that N is started in the stable total state $\langle \hat{a}, b \rangle$ and that the input changes to a. We say that there is a *static 1-hazard* on an output vertex i for the transition $\hat{a} \rightarrow a$ iff $b_i = 1$, $c_i = 1$ for every state $c \in \text{out}(R_a, b)$, and there exists a state d such that $b R_a^* d$ and $d_i \neq 1$. A static 0-hazard is defined similarly.

The following theorem shows how the results of ternary simulation can be used to detect static output hazards:

Theorem 3 Assume network N is started in the stable total state $\langle \hat{a}, b \rangle$ and the input changes to a. Let $\mathbf{y}^A$ be the result of Algorithm A and $\mathbf{z}^B$ be the result of Algorithm B. Then output vertex i has a static 1(0)-hazard iff $b_i = \mathbf{z}_i^B = 1(0)$ and $\mathbf{y}_i^A = \times$.

The concept of speed-independence of Section 3 is not very convenient, since it is defined in terms of an XMW analysis. However, by Theorem 2, it is easy to verify that the following definition of speed-independence is equivalent to that of Section 3.

Definition 1' A network N started in the binary stable total state $\langle \hat{a}, b \rangle$ and with a new binary input vector a, is said to be speed-independent with respect to $\langle a, b \rangle$ if the result of Algorithm B for this change is binary.

5. Behavioral Model

In this section we combine the different concepts introduced in the earlier sections to define an abstract finite-state machine to describe the speed-independent behavior of a given network.

Definition 2 The *fundamental-mode, speed-independent, asynchronous machine* (FSAM), M, corresponding to a network N, consists of the following:

1) A finite set Q of internal states. Each state $q \in Q$ corresponds to a binary stable state of N.

2) A finite set Σ of input symbols. Each input symbol $\sigma \in \Sigma$ is a nonempty subset of $\{1,\ldots,n\}$ representing the set of input variables being complemented.

3) A finite nonempty set $Z \subseteq \{1,\ldots,m\}$ of output vertices.

4) A mapping τ (called the *transition map* of M) of a subset D of $Q \times \Sigma$ to Q.

5) An initial stable state $q^0 \in Q$.

Furthermore, the transition map must satisfy the following condition: The network N is speed-independent with respect to the state p for the input change σ and this transition takes N to the state q iff $\tau(p,\sigma)=q$. The reader can easily verify that this condition implies that M is deterministic.

We denote M by the ordered quintuple $M=(Q,\Sigma,Z,\tau,q^0)$. Note that M may be incompletely specified, i.e. $D \subset Q \times \Sigma$. This means that if, for example, $(p,\sigma) \notin D$, then N is not speed-independent with respect to the state p for the input change σ. Hence, a network may be speed-independent only with respect to certain transitions. The reader should also note that there cannot exist any state $p \in Q$ and input symbol $\sigma \in \Sigma$ such that $\tau(p,\sigma)=p$, i.e. there cannot be any self loops in a diagram of M. The reason for this is that each state $p \in Q$ represents a stable state of N, and the state of the network includes the values of the input vertices. Since an input symbol σ represents some input variables being complemented, it follows that if $\tau(p,\sigma)=q$, then $p \neq q$.

The definition of M is somewhat redundant, since the input symbol σ that takes the machine from state p to state q can be deduced from the values of the first n components of p and q. However, the redundancy simplifies the notation and will be retained.

In many cases it is appropriate to impose one further restriction on M. As described in Section 4, it is often important to ensure that the transitions of a network are free of output hazards. A FSAM M is said to be *hazard-free* if for all $p,q \in Q$ and $\sigma \in \Sigma$ such that $\tau(p,\sigma)=q$, there is no hazard on any output vertex during the transition from p to q.

To illustrate the above definitions, and also to give an example of a nontrivial speed-independent network, we present the gate circuit G_4 of Fig. 4. Following [3], and associating one vertex with each gate, it is easy to derive the abstract network N_1 of Fig. 5, with the excitation functions:

$$Y_1 = x_1 \quad Y_3 = y_1y_9 \quad Y_5 = (y_3 + y_6)' \quad Y_7 = y_2y_6 \quad Y_9 = (y_7 + y_{10})'$$
$$Y_2 = x_2 \quad Y_4 = y_1y_{10} \quad Y_6 = (y_4 + y_5)' \quad Y_8 = y_2y_5 \quad Y_{10} = (y_8 + y_9)'$$

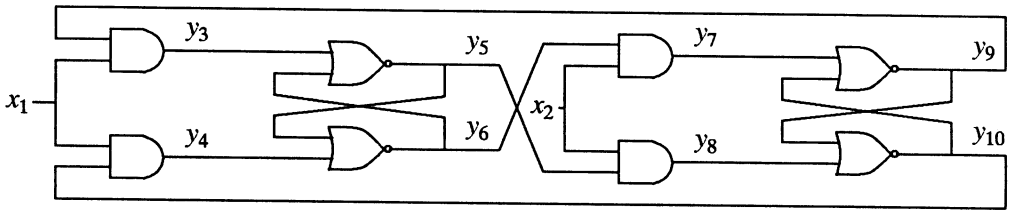

Figure 4. Gate circuit G_4.

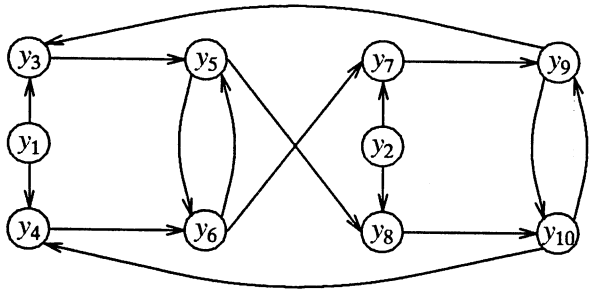

Figure 5. Network N_1.

In order to simplify the analysis, the reduction procedure described in [3] is carried out using the set $\{5, 9\}$ as the feedback vertex set. This yields the reduced network N_1^r of Fig. 6 with excitation functions:

$$Y_1 = x_1 \quad Y_5 = (y_1y_9 + (y_1(y_2y_5 + y_9)' + y_5)')'$$
$$Y_2 = x_2 \quad Y_9 = (y_2(y_1(y_2y_5 + y_9)' + y_5)' + (y_2y_5 + y_9)')'$$

(Note that it was shown in [3] that XMW analyses of N_1 and N_1^r are equivalent. We will return to this in Section 8.)

Assume that N_1^r is started in the stable total state $\langle 00, 0011\rangle$, i.e. assume that $q^0 = 0011$. The FSAM M_1 corresponding to the network N_1^r is shown in Fig. 7. (To simplify the picture, it is assumed that all vertices of N_1^r are output vertices.) Note that no transition caused by a multiple-input change is speed-independent for this network. It is interesting to note that for any state p in M for which an input symbol σ is allowed, any odd number of σ's will take the machine to the same state. For example, $\tau(0011, \{1\}) = 1001$, $\tau(1001, \{1\}) = 0001$ and

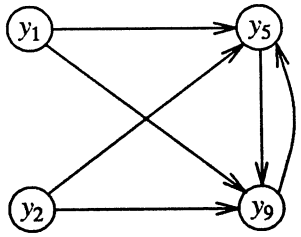

Figure 6. Network N_1^r.

$\tau(0001, \{1\}) = 1001$. In Section 7 we will show that this is no coincidence but a fundamental property of speed-independent circuits.

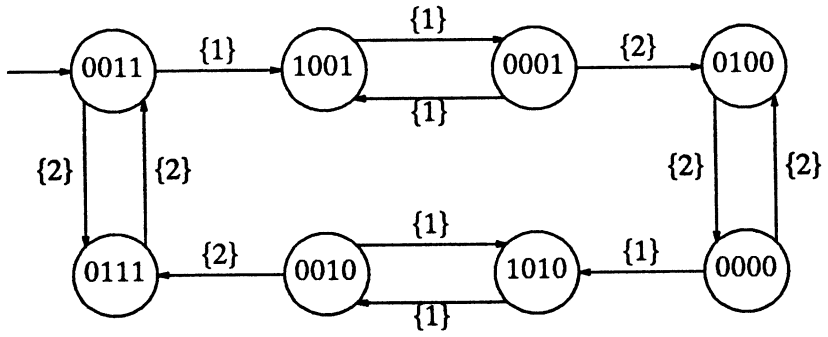

Figure 7. FSAM M_1 corresponding to network N_1^r.

6. The Composite Network Function

In this section we show how to derive a combinational network from a sequential network, in such a way that the two networks yield identical results for ternary simulation. This transformation will serve as one of our basic tools to derive the behavior of an arbitrary speed-independent network in Section 7. The intuitive idea can be illustrated as follows. Suppose we are given a network N with m vertices as in Fig. 8(a). Assume the state $\langle \hat{a}, b \rangle$ is a stable total state of N and that the input changes to a. Normally, using the ternary simulation algorithm described in Section 4, we start with the state b, and then iteratively compute "next-states", $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^A$. The next-states are computed as follows:

$$\mathbf{y}^i = Y(\mathbf{a}, \mathbf{y}^{i-1}),$$

where $\mathbf{a}$ is the $l.u.b.\{\hat{a}, a\}$. This process is repeated until a stable state $\mathbf{y}^A$ is reached. We know that $A \leq m$ by the monotonicity of Algorithm A. Consider now the following alternative approach. Suppose that all

edges leaving the vertices are broken. We then get a combinational circuit of the form shown in Fig. 8(b). By construction, this combinational circuit is such that $y_j^{out} = Y_j(x, y^{in})$ for $1 \leq j \leq m$. Now, by connecting m copies of this combinational network in series we get a combinational circuit of the form shown in Fig. 8(c). It is easy to see that, if $y^{in} = b$ and $x = \mathbf{a}$, we get $y^{out} = \mathbf{y}^A$. The reader can easily verify that the same idea can be applied for Algorithm B as well. These concepts will be made more precise below.

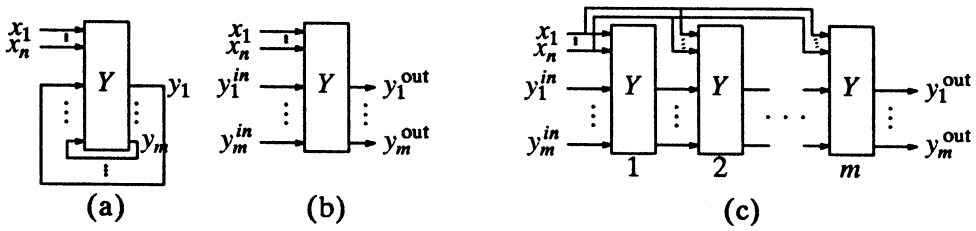

Figure 8. (a) Original network;
(b) combinational network;
(c) composite network.

Given a network N, define its *composition function* $F : T^{n+m} \to T^m$ as $F(x,y) = Y^{(m)}(x,y)$, where $Y^{(h)}$ is defined recursively as follows:

$$Y^{(h)}(x,y) = \begin{cases} Y(x, Y^{(h-1)}(x,y)) & \text{if } h \geq 1 \\ y & \text{if } h = 0 \end{cases}$$

The following properties of F will be used later.

Proposition 4 $F(x,y)$ is monotonic, i.e. $\langle \tilde{a}, \tilde{b} \rangle \sqsubseteq \langle a, b \rangle$ implies that $F(\tilde{a}, \tilde{b}) \sqsubseteq F(a,b)$.

Proof: Assume that $\langle \tilde{a}, \tilde{b} \rangle \sqsubseteq \langle a, b \rangle$. We prove by induction on h that $Y^{(h)}(\tilde{a}, \tilde{b}) \sqsubseteq Y^{(h)}(a,b)$ for $h \geq 0$. From this it follows immediately that $F(\tilde{a}, \tilde{b}) \sqsubseteq F(a,b)$, and hence that F is monotonic.

Basis:

$h = 0$. Trivially true.

Induction hypothesis:

Assume that $Y^{(h)}(\tilde{a}, \tilde{b}) \sqsubseteq Y^{(h)}(a,b)$, for some $h \geq 0$.

Induction step:

By the definition of $Y^{(h+1)}$ it follows that $Y^{(h+1)}(\tilde{a}, \tilde{b}) = Y(\tilde{a}, Y^{(h)}(\tilde{a}, \tilde{b}))$, and that $Y^{(h+1)}(a,b) = Y(a, Y^{(h)}(a,b))$. Since $Y^{(h)}(\tilde{a}, \tilde{b}) \sqsubseteq Y^{(h)}(a,b)$, by the induction hypothesis, and Y is

assumed to be monotonic, we can conclude that $Y(\tilde{a}, Y^{(h)}(\tilde{a}, \tilde{b})) \sqsubseteq Y(a, Y^{(h)}(a,b))$. Hence the induction step goes through and the proposition follows. □

Proposition 5 If $\langle a,b \rangle$ is a stable total state of N, then $F(a,b)=b$.

Proof: We prove by induction on h that $Y^{(h)}(a,b)=b$ for $h \geq 0$. It then follows immediately that $F(a,b)=b$.

Basis:

$h=0$. Trivially true.

Induction hypothesis:

Assume that $Y^{(h)}(a,b) = b$, for some $h \geq 0$.

Induction step:

By the definition of $Y^{(h+1)}$ it follows that $Y^{(h+1)}(a,b)=Y(a,Y^{(h)}(a,b))$. Since $Y^{(h)}(a,b) = b$, by the induction hypothesis, and $\langle a,b \rangle$ is assumed to be a stable total state, i.e. $Y(a,b)=b$, we can conclude that $Y^{(h+1)}(a,b)=b$ and the induction step goes through. □

Lemma 1 Assume a network N is started in the stable total state $\langle \hat{a}, b \rangle$ and that the input is changed to a. Let $\mathbf{y}^A$ and $\mathbf{z}^B$ be the results of Algorithms A and B for this input change respectively. Furthermore, let $\mathbf{a} = l.u.b.\{\hat{a}, a\}$ and $F(x,y)$ be the composition function of N. We then have the following properties of F:

(i)	$F(\hat{a},b)=b$	(stability)
(ii)	$F(\mathbf{a},b)=\mathbf{y}^A$	(result of Alg. A)
(iii)	$F(\mathbf{a},\mathbf{y}^A)=\mathbf{y}^A$	(stability)
(iv)	$F(a,\mathbf{y}^A)=\mathbf{z}^B$	(result of Alg. B)
(v)	$F(a,\mathbf{z}^B)=\mathbf{z}^B$	(stability)

Proof: Properties (i), (iii), and (v) follows immediately from Proposition 5 and the definition of Algorithms A and B. For property (ii), let $\mathbf{y}^0,\ldots,\mathbf{y}^A$ be the sequence of states produced by Algorithm A. Note that $\mathbf{y}^0=b$ and that $A \leq m$. We now show, by induction on h, that $Y^{(h)}(\mathbf{a},b)=\mathbf{y}^h$ for $0 \leq h \leq A$.

Basis:

$h=0$. Since $Y^{(0)}(x,b)=b$, and $\mathbf{y}^0=b$, the result follows immediately.

Induction hypothesis:

Assume $Y^{(h)}(\mathbf{a},b)=\mathbf{y}^h$ for some $h \geq 0$.

Induction step:

By the definition of $Y^{(h+1)}$ it follows that $Y^{(h+1)}(\mathbf{a},b)=Y(\mathbf{a},Y^{(h)}(\mathbf{a},b))$. Similarly, by the definition of Algorithm A, it follows that $\mathbf{y}^{h+1}=Y(\mathbf{a},\mathbf{y}^h)$. Since $Y^{(h)}(\mathbf{a},b) = \mathbf{y}^h$, by the induction hypothesis, we can conclude that

$$Y^{(h+1)}(\mathbf{a},b) = Y(\mathbf{a},Y^{(h)}(\mathbf{a},b)) = Y(\mathbf{a},\mathbf{y}^h) = \mathbf{y}^{h+1}.$$

Hence the induction step goes through.
From the above we can conclude in particular that $Y^{(A)}(\mathbf{a},b) = \mathbf{y}^A$. If $A=m$, then $F(\mathbf{a},b)=Y^{(m)}(\mathbf{a},b)=Y^{(A)}(\mathbf{a},b)=\mathbf{y}^A$ and property (ii) is proved. Otherwise, i.e. if $A<m$, it is easy to verify that $Y^{(h+1)}(\mathbf{a},b)=Y^{(h)}(\mathbf{a},b)=Y^{(A)}(\mathbf{a},b)=\mathbf{y}^A$ for $h\geq A$. This follows simply from the fact that $\langle \mathbf{a},\mathbf{y}^A\rangle$ is a stable total state, i.e. that $\mathbf{y}^A=Y(\mathbf{a},\mathbf{y}^A)$. Altogether this establishes property (ii). Using similar arguments, it is easy to verify property (iv). □

The above Lemma is crucial for the proofs in Section 7, since it allows us to use the composite network function F instead of the network N directly. Hence, we can establish the results using a combinational, rather than sequential, network — a substantial simplification.

7. Fundamental Properties of Speed-Independent Circuits

In this section we derive some general properties that are common to all speed-independent circuits. The following three theorems summarize the main results of this paper.

Theorem 4 Let N be any network. Let $M=(Q,\Sigma,Z,\tau,q^0)$ be the fundamental-mode, speed-independent, asynchronous machine corresponding to N. If there exist states p,q and $r\in Q$, and an input symbol $\sigma\in\Sigma$, such that $\tau(p,\sigma)=q$ and $\tau(q,\sigma)=r$, then $\tau(r,\sigma)=q$.

In other words, any odd number of changes of the same set of inputs must leave the network in the same state. An interesting special case occurs when the network has only one input. From the above theorem it follows that any FSAM, for a circuit with only one input, can have at most 3 states. Since the value of the input vertex is part of the state of the network, any such FSAM must have at least 2 states. Hence, a one-input network can only have an FSAM with 2 or 3 states. In fact, it is easy to see that the only possible machines are the ones shown in Fig. 9.

From this we can conclude, for example, that there does not exist a speed-independent mod-2 counter (a mod-2 counter is a circuit with one input, one output, and whose output changes with half the frequency of its input signal). In fact, there does not exist a speed-independent mod-k counter for any $k>1$.

The next theorem deals with multiple-input changes, i.e. when more than one input vertex changes at the same time.

Figure 9. The only possible FSAM's for a network with only one input.

Theorem 5 Let N be any network. Let $M=(Q,\Sigma,Z,\tau,q^0)$ be the fundamental-mode, speed-independent, asynchronous machine corresponding to N. If there exist states p,q and $r\in Q$ and input symbols σ and σ_1, such that σ_1 is a proper subset of σ, $\tau(p,\sigma)=r$, and $\tau(p,\sigma_1)=q$, then $\tau(q,\sigma-\sigma_1)=r$.

The theorem states, roughly, that if a multiple-input change is speed-independent, then the network must end up in the same state if this multiple-input change is made "step-by-step".

Theorem 4 above dealt with the total state of a speed-independent network. Our last theorem of this section gives conditions on the values of specific output vertices. Since we deal here with outputs, we restrict our attention to hazard-free transitions.

Theorem 6 Let N be any network. Let $M=(Q,\Sigma,Z,\tau,q^0)$ be the hazard-free, fundamental-mode, speed-independent, asynchronous machine corresponding to N. Assume that vertex j is an output vertex, i.e. that $j\in Z$. If there exist states p,q and $r\in Q$, and an input symbol $\sigma\in\Sigma$, such that $\tau(p,\sigma)=q$, $\tau(q,\sigma)=r$ and $p_j=q_j=\alpha\in\{0,1\}$, then $r_j=\alpha$.

From the above and from Theorem 4 we can draw the following conclusion. If an output does not change value for some input change σ, then it will not change for any sequence of σ's.

We prove the above theorems with the aid of a series of lemmas. In fact, we prove a somewhat stronger result in that we do not restrict our attention to only binary stable states of N. The following assumptions will be used for Lemmas 2, 3 and 4. Let N be any network operated according to the fundamental-mode assumption, i.e. the network is given sufficient time to "settle down" after every input change before the input changes again. Let F denote the composite network function of N as defined in Section 6. Furthermore, assume that the input sequence is given by $a^0, a^1, a^2, a^3,\ldots = \hat{a}, a, \hat{a}, a,\ldots$, i.e. that the input is cycled between the binary input vectors $\hat{a}$ and a. Assume that $\langle\hat{a},b^0\rangle$ is a stable total state of N. Let $b^{i,i+1}$ denote the result of

Algorithm A for the transition from the stable total state $\langle a^i, b^i \rangle$ when the input changes to a^{i+1}. Similarly, let b^{i+1} denote the result of Algorithm B for the same transition. The following lemma is the key lemma to all subsequent results. The lemma states that if, at some point, a vertex with a binary value does not react to an input change, it will never react thereafter.

Lemma 2 If there exists a $k \geq 1$ such that $b_j^{k-1} = b_j^{k-1,k} = b_j^k = \alpha \in \{0,1\}$, then $b_j^{i-1,i} = b_j^i = \alpha$ for all $i \geq k$.

Proof: We prove this by induction on i.

Basis:

$i = k$. Trivially true by the assumptions in the lemma.

Induction hypothesis:

Assume $b_j^{i-1,i} = b_j^i = \alpha$ for some $i \geq k$.

Induction step

First note that $l.u.b.\{a^{i-1}, a^i\} = l.u.b.\{a^i, a^{i+1}\} = l.u.b.\{\hat{a}, a\} = \mathbf{a}$. By the monotonicity of Algorithm B (Proposition 3), it follows that $b^{i-1,i} \sqsupseteq b^i$ and hence, by the monotonicity of the composite network function (Proposition 4), that $F_j(\mathbf{a}, b^{i-1,i}) \sqsupseteq F_j(\mathbf{a}, b^i)$. However, by Lemma 1 (iii), $F(\mathbf{a}, b^{i-1,i}) = b^{i-1,i}$ and, in particular, $F_j(\mathbf{a}, b^{i-1,i}) = b_j^{i-1,i}$ which is equal to α by the induction hypothesis. Hence, $\alpha = F_j(\mathbf{a}, b^{i-1,i}) \sqsupseteq F_j(\mathbf{a}, b^i)$ and thus $F_j(\mathbf{a}, b^i) = \alpha$. Furthermore, by Lemma 1 property (ii), it follows that $b_j^{i,i+1} = F_j(\mathbf{a}, b^i)$ and hence $b_j^{i,i+1} = \alpha$. In other words, the value of vertex j after Algorithm A for the input change a^i to a^{i+1} will be α. Finally, by the monotonicity of Algorithm B (Proposition 3) it follows immediately that $b^{i,i+1} \sqsupseteq b^{i+1}$ and therefore that $b_j^{i+1} = \alpha$. Hence the induction step goes through and the lemma follows. □

From Lemma 2 we get the following corollary.

Corollary 1 (Monotonicity for change sequences) For all $k \geq 1$, $b^{k-1,k} \sqsupseteq b^{k,k+1}$.

Proof: It suffices to show that whenever $b_j^{k-1,k}$ is binary, then $b_j^{k,k+1}$ has the same value. Suppose $b_j^{k-1,k} = \alpha \in \{0,1\}$. From the monotonicity of Algorithm A (Proposition 2) and the monotonicity of Algorithm B (Proposition 3), it follows that $b_j^{k-1} = b_j^k = \alpha$. Hence, $b_j^{k-1} = b_j^{k-1,k} = b_j^k = \alpha \in \{0,1\}$ and Lemma 2 applies. Thus $b_j^{i-1,i} = b_j^i = \alpha$ for all $i \geq k$ and, in particular, $b_j^{k,k+1} = \alpha$. □

The following two lemmas give conditions on the values of a vertex after an odd and an even number of input changes respectively. The first lemma states that, if a vertex has a binary value after one input change, then it will have the same value after any odd number of input changes. The second lemma is similar, but for even number of changes.

Lemma 3 If $b_j^1 = \alpha \in \{0,1\}$, then $b_j^{2i-1} = \alpha$ for all $i \geq 1$.

Proof: By induction on i.

Basis:

$i = 1$. Trivially true by the assumption in the lemma.

Induction Hypothesis:

Assume $b_j^{2i-1} = \alpha$ for some $i \geq 1$.

Induction Step:

Since $i \geq 1$ and thus $2i-2 \geq 0$, the state $b^{2i-2,2i-1}$ is well defined. By Lemma 2, property (iv), it follows that $b^{2i-1} = F(a^{2i-1}, b^{2i-2,2i-1})$ and, in particular, that $b_j^{2i-1} = F_j(a^{2i-1}, b^{2i-2,2i-1})$. By the same arguments, $b_j^{2i+1} = F_j(a^{2i+1}, b^{2i,2i+1})$. However, by Corollary 1 it follows that $b^{2i-2,2i-1} \sqsupseteq b^{2i-1,2i} \sqsupseteq b^{2i,2i+1}$. Also, by assumption, $a^{2i-1} = a^{2i+1} = a$ and thus $\langle a^{2i-1}, b^{2i-2,2i-1} \rangle \sqsupseteq \langle a^{2i+1}, b^{2i,2i+1} \rangle$. This, together with the monotonicity of F (Proposition 4), shows that $F(a^{2i-1}, b^{2i-2,2i-1}) \sqsupseteq F(a^{2i+1}, b^{2i,2i+1})$. Thus, $b_j^{2i-1} = F_j(a^{2i-1}, b^{2i-2,2i-1}) \sqsupseteq F_j(a^{2i+1}, b^{2i,2i+1}) = b_j^{2i+1}$, and since $b_j^{2i-1} = \alpha$, by the induction hypothesis, it follows that $b_j^{2i+1} = \alpha$ and the induction step goes through. □

Lemma 4 If $b_j^2 = \alpha \in \{0,1\}$, then $b_j^{2i} = \alpha$ for all $i \geq 1$.

Proof: By arguments similar to those in the proof of Lemma 3. □

We are now in a position to prove Theorem 4.

Proof of Theorem 4: There are two cases to consider. First, if $r = p$, the result follows trivially from the fact that M is a deterministic machine. Otherwise, when $r \neq p$ we prove the result by contradiction. Assume $\tau(r, \sigma) = s$ for some $s \neq q$. Then the state s must differ from the state q in at least one component, say $q_j \neq s_j$. However, by the definition of M it follows that all the states p, q, r, and s are binary stable states of N. Since $\tau(p, \sigma) = q$ and $q_j = \alpha \in \{0,1\}$, Lemma 3 applies, showing that s_j must be equal to α, contradicting the assumption that $q_j \neq s_j$. Hence the result follows. □

Theorem 5 follows from a more fundamental property of the XMW race model.

Proof of Theorem 5: Let $\hat{a}=(p_1,\ldots,p_n)$, $\tilde{a}=(q_1,\ldots,q_n)$, and $a=(r_1,\ldots,r_n)$. It is easy to verify that $\tilde{a}\neq\hat{a}$, $\tilde{a}\neq a$, and that $\tilde{a}\sqsubseteq l.u.b.\{\hat{a},a\}$. Since p is assumed to be a stable state of N, it follows that $\langle\hat{a},p\rangle$ must be a stable total state of N, and hence, by Proposition 1, it follows that:

$$\{d: d\in\text{out}(R_a,c),\ \text{and}\ c\in\text{out}(R_{\hat{a}},p)\}\subseteq\text{out}(R_a,p).$$

Since $\tau(p,\sigma)=r$ it follows that $\text{out}(R_a,p)=\{r\}$, and similarly, since $\tau(p,\sigma_1)=q$, that $\text{out}(R_{\hat{a}},p)=\{q\}$. Altogether this gives that $\text{out}(R_a,q)\subseteq\text{out}(R_a,p)$. However, since $\text{out}(R_a,q)\neq\varnothing$ and $\text{out}(R_a,p)=\{r\}$ it follows that $\text{out}(R_a,q)=\{r\}$ and thus that $\tau(q,\sigma-\sigma_1)=r$ establishing the theorem. □

Finally we prove Theorem 6.

Proof of Theorem 6: There are two cases to consider. First, if $r=p$, the result follows trivially. Otherwise, if $p\neq r$, consider the output vertex j. Since all transitions are assumed to be hazard-free, the transition from p to q in particular must be hazard-free. However, since $p_j=q_j=\alpha\in\{0,1\}$, Theorem 3 applies, showing that the result of Algorithm A for this transition must be equal to α. Using the same notation as in Lemmas 2, 3 and 4 above, we can conclude that $p_j=b_j^0=b_j^{0,1}=b_j^1=q_j=\alpha$. Hence Lemma 2 applies, establishing that $b_j^i=\alpha$ for all $i\geq 1$, and in particular that $r_j=\alpha$. □

Using the above results, it is easy to verify that the following six types of vertex behavior are the only possible for a vertex in a speed-independent network when the input alternates between the two binary input vectors $\hat{a}$ and a:

1) The vertex never reacts.
2) The vertex changes value on the first input change and keeps this value from there on.
3) The vertex changes value for every input change.
4) The vertex keeps the same value, although there may be a short pulse during every input change.
5) The vertex keeps the same value, although there may be a short pulse during the first input change.

6) The vertex keeps the same value for the first input change, except that there may be a short pulse during this transition. For the remaining changes, the vertex changes value for every input change.

Note that only behaviors 1-3 are normally acceptable for an output vertex.

The gate circuit G_5 of Fig. 10 contains gates of all the above types if it is started in the stable total state $x=0$, $y=(y_1,\ldots,y_7)=1010100$, and the input oscillates between 1 and 0. In particular, gate 1 is of type 1, gates 2 and 3 are of type 2, gate 4 is of type 5, gate 5 is of type 3, gate 6 is of type 4, and finally gate 7 is of type 6.

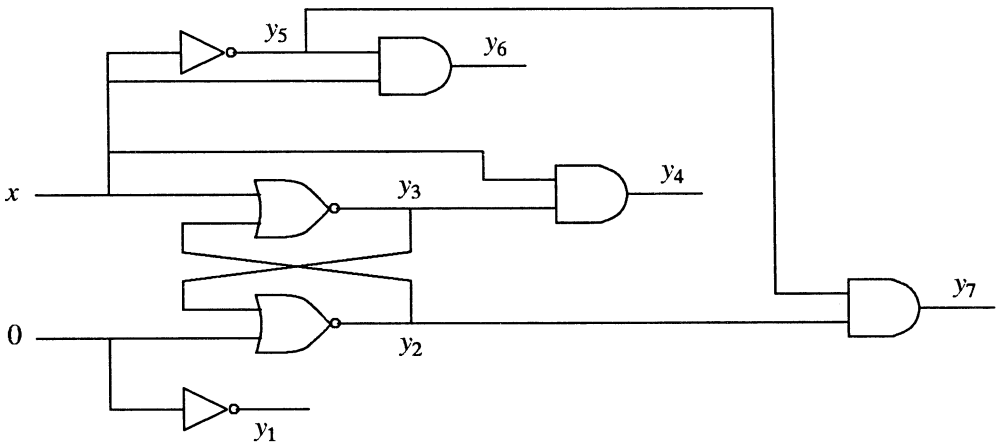

Figure 10. Gate network G_5.

8. Alternative Notions of Speed-Independence

In the previous section we derived some general properties of speed-independent circuits. However, we used a rather restricted definition of speed-independence. In this section we extend these result to different definitions of speed-independence.

Ternary simulation was the main tool used to prove the results in the previous section. It is easy to verify that all the results of Section 7 (except possibly Theorem 5) carry over to any race and network model that can be shown to correspond to ternary simulation. In [3] two summarizing theorems were proved for gate circuits and switch-level circuits respectively. However, the following concepts are required before we can state these theorems. The *general multiple winner* (GMW) model is

a race model very similar to the XMW model, but one in which all state variables are assumed to be binary. Hence, the model is primarily tailored towards gate circuits. As with the XMW model, delays are assumed to be inertial and arbitrary, but finite. For more details about the GMW model, the reader is referred to [2]. In a *gate-delay model*, only the gates are assumed to have delays associated with them. In a *gate- and wire-delay model*, both the wires and the gates are assumed to have inertial delays associated with them. Finally, in a *feedback-delay model*, it is assumed that there are delays only in the feedback lines of the circuit. The following theorem for gate circuits was proved in [3]:

Theorem 7 Let N be the gate-delay model of a circuit, let $\dot{N}$ be the feedback-delay model, and $\tilde{N}$ the gate- and wire-delay model. The following analysis techniques are all equivalent for gate circuits from the point of view of nontransient state behavior and static output hazards:

1. GMW analysis of $\tilde{N}$.
2. XMW analysis of (a) $\tilde{N}$, (b) N, and (c) $\dot{N}$.
3. Ternary simulation of (a) $\tilde{N}$, (b) N, and (c) $\dot{N}$.

There are several different alternatives for calculating node excitations in MOS switch-level models; these representing different design philosophies for CMOS and NMOS circuits. In [3] some of these alternatives for CMOS circuits were discussed. For more details, the reader is referred to [3] and also to Bryant's work [1]. Let N be a *node-delay* model, i.e. a model in which a delay is associated with every internal node; let $\dot{N}$ be the same model reduced to feedback variables; and let $\tilde{N}$ be the *node- and transistor-delay* model where delays are assumed in both nodes and transistors. In [3] the following result was established:

Theorem 8 The following analysis techniques are all equivalent for switch-level circuits, using any one of the node excitation models described in [1, 3], from the point of view of nontransient state behavior and static output hazards:

1. XMW analysis of: (a) $\tilde{N}$, (b) N, and (c) $\dot{N}$.
2. Ternary simulation of: (a) $\tilde{N}$, (b) N, and (c) $\dot{N}$.

The above two theorems allow us to give a number of different definitions of speed-independence, that are all equivalent in the sense that the results of Section 7 hold for each of them. For example, if we assume that only the feedback lines of a gate circuit can have delays and that transitions can go through an intermediate value $\times$ (i.e. using the XMW model) we know, by Theorem 7, that Theorems 4 and 6 are still valid. Hence, even in such a restricted model, where only the feedback lines have delays and the rest of the circuit is built of ideal

(delay-free) components, one cannot construct a speed-independent mod-2 counter.

One of our basic assumptions in the definition of speed-independence was that the circuits are operated in fundamental mode [8], i.e. input changes occur so seldom that the circuits have time to settle down after each change before the input changes again. This assumption seems to contradict the basic idea of speed-independence, since here we implicitly introduce an assumption about the sizes of the delays in the circuit. However, since this assumption makes it easier to design speed-independent circuits, and our main results are negative, it follows that if the fundamental-mode assumption is removed, the class of speed-independent circuits can only become smaller. In fact, suppose a circuit behaves correctly for a change σ_1 followed by σ_2, where $\sigma_1 \cap \sigma_2 = \emptyset$ and σ_2 may occur before the network has reached a stable state. It is easy to verify that a necessary and sufficient condition for this is that the multiple-input change $\sigma_1 \cup \sigma_2$ must be speed-independent.

Another possible change in the definition of speed-independence is to relax the condition that a circuit must reach a unique stable state after each input change. This can be achieved by allowing the circuit to enter any nontransient oscillation as a result of an input change. In such a case one can define a nondeterministic version of the sequential fundamental-mode machine of Section 5. However, it is not difficult to show that, for the XMW model, the results of Section 7 still hold for such a model. In particular, one can show that the *l.u.b.* of all the states reachable after some sequence of input changes is equal to the result obtained by using ternary simulation for every input change. Since Lemmas 2-4 did not require that the total states of N reached after each input change be binary, the result follows immediately. Whether this result also holds for the GMW model using gate and wire delays is still an open question.

Finally, it is worth mentioning that the race model used by Muller and Bartky in their original work on speed-independence was the GMW model, but one in which only the gates were assumed to have delays. It is not known whether using such a definition of speed-independence would substantially increase the size of the class of speed-independent gate circuits.

9. Conclusions

In this paper we have derived some general properties of speed-independent circuits. We have shown that the class of speed-independent circuits is quite small, and that many useful functions cannot be realized with a speed-independent design. One might argue that this implies that the concept of speed-independence must be abandoned. However, one may also interpret the results as showing that the definitions of speed-independence used in this paper are too pessimistic — in particular, that the race models chosen are too pessimistic. It is not very surprising that one cannot design a circuit that is guaranteed to work when the "devil's advocate" is allowed to insert arbitrarily many, arbitrarily large delays anywhere in the circuit. This points to the need of more realistic race models — models in which delays are bounded by some lower and upper bounds. One could then define a circuit to be speed-insensitive if the behavior of the circuit is independent of the size of the delays under the assumption that the delays are within these bounds. Unfortunately, at the present time, there does not exist any method to analyze a circuit under such a delay assumption. Methods for doing so must be developed before speed-insensitive circuits can be designed and verified reliably.

Acknowledgement

The author wishes to thank Professor John Brzozowski of the Department of Computer Science, University of Waterloo, for many useful comments and suggestions regarding this paper.

References

[1] R. E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Transactions on Computers*, Vol. C-33, No. 2, pp. 160-177, Feb. 1984.

[2] J. A. Brzozowski and M. Yoeli, "On a Ternary Model of Gate Networks," *IEEE Transactions on Computers*, Vol. C-28, No. 3, pp. 178-183, Mar. 1979.

[3] J. A. Brzozowski and C-J. Seger, "A Unified Theory of Asynchronous Circuits," Technical Report CS-87-24, Department of Computer Science, University of Waterloo, Waterloo, Ontario,

Canada, Mar. 1987.

[4] J. C. Ebergen, *Translating Programs into Delay-Insensitive Circuits*, Ph.D. Thesis, Eindhoven University of Technology, 1987.

[5] E. B. Eichelberger, "Hazard Detection in Combinational and Sequential Switching Circuits," *IBM Journal of Research and Development*, Vol. 9, pp. 90-99, Mar. 1965.

[6] T. Lengauer and S. Näher, "An analysis of Ternary Simulation as a Tool for Race-detection in Digital MOS Circuits," *Integration, the VLSI Journal*, No. 4, pp. 309-330, 1986.

[7] A. J. Martin, "The Design of a Self-Timed Circuit for Distributed Mutual Exclusion," in *Proc. 1985 Chapel Hill Conference on VLSI*, Computer Science Press, pp. 245-260, 1985.

[8] E. J. McCluskey, Jr., "Transients in Combinational Logic Circuits," in *Redundancy Techniques for Computing Systems*, Wilcox, R. H. and W. C. Mann, Eds., Spartan Books, Washington, 1962, pp. 9-46.

[9] R. E. Miller, *Switching Theory, Vol. 2*, Wiley, New York, 1965.

[10] C. E. Molnar, T.-P. Fang, and F. U. Rosenberger, "Synthesis of Delay-Insensitive Modules," in *Proc. 1985 Chapel Hill Conference on VLSI*, Computer Science Press, pp. 67-86, 1985.

[11] D. E. Muller and W. S. Bartky, "A Theory of Asynchronous Circuits," *Proceedings of an International Symposium on the Theory of Switching*, Vol. 29 of the *Annals of the Computation Laboratory of Harvard University*, pp. 204-243, Harvard University Press, 1959.

[12] V. Ramachandran, "Algorithmic Aspects of MOS VLSI Switch-Level Simulation with Race Detection," *IEEE Transactions on Computers*, Vol. C-35, No. 5, pp. 462-475, May 1986.

[13] C-J. Seger, "The Complexity of Race Analysis," In preparation.

[14] C. L. Seitz, "System Timing," in *Introduction to VLSI systems*, C. A. Mead and L. A. Conway, Eds., Addison-Wesley Publishing Company, Reading, Massachusetts, 1980, pp. 218-262.