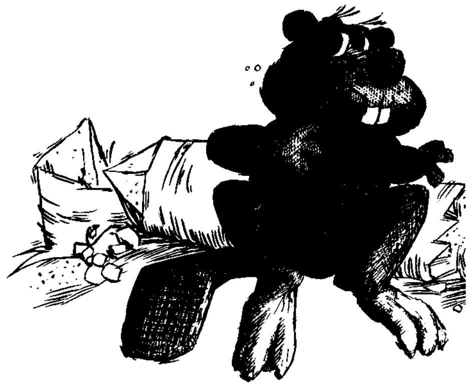

A Characterization of Ternary Simulation of Gate Networks

J.A. Brzozowski
C-J. Seger
VLSI Group

CS-85-37

October, 1985

A Characterization of Ternary Simulation of Gate Networks*

J.A. Brzozowski and C-J. Seger

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada

ABSTRACT

Ternary simulation techniques provide efficient methods for the analysis of the behavior of VLSI circuits. However, the results of ternary simulation have not been completely characterized. In this paper we prove a somewhat modified version of the Brzozowski-Yoeli conjecture (stated in 1976) that the results of the ternary simulation of a gate network N correspond to the results of the binary race analysis of $\tilde{N}$ in the "multiple-winner" model, where $\tilde{N}$ is the network N in which a delay has been inserted in each wire.

1. Introduction

In 1965 Eichelberger [E] proposed the use of ternary simulation for the analysis of races in asynchronous sequential networks. His method was applied to a model of sequential circuits in which the state is determined by feedback lines, and was somewhat informal. In 1979 Eichelberger's ternary algorithm was applied by Brzozowski and Yoeli [B-Y] to a model in which the state is defined by all the gate outputs; also the relationship of the ternary algorithm to the binary General Multiple Winner (GMW) model of races was studied. However, only a partial correspondence was established, leaving open a conjecture (formulated in 1976 and published in 1979 [B-Y]) about complete correspondence. The importance of ternary simulation for practical applications was clearly demonstrated by Bryant [B83a, B84], who incorporated ternary algorithms in his simulators for MOS VLSI circuits. Bryant [B83b] also worked on the conjecture and obtained a characterization of the first part of the ternary algorithm. In this paper we provide a complete characterization of the ternary algorithm.

We introduce the problem by means of an example [B-Y]. Consider the network N_1 in Fig.1. One verifies that the total state $x = 0$, $y = (y_1, y_2, y_3) = (0, 1, 1)$ is stable. (We will write $y = 011$ for short.)

* This research was supported by the Natural Sciences and Engineering Research Council of Canada under grants No. A1617 and A0871.

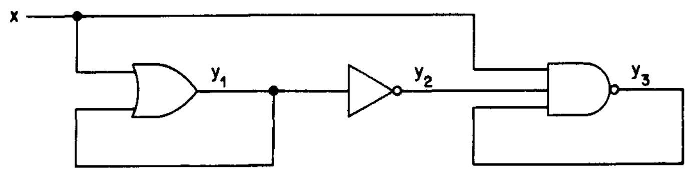

Figure 1. Network N_1.

Suppose now that the input changes to $x = 1$; what will be the final state of the network? This question constitutes the basic problem in the analysis of asynchronous sequential networks.

In our example, the behavior of the network is governed by the following gate excitation equations:

$$Y_1 = x + y_1, \quad Y_2 = y_1', \text{ and } \quad Y_3 = (x\, y_2 y_3)',$$

where Y_i gives the value of the boolean function computed by the gate whose output is y_i, i.e. it is the excitation of the gate. In the total state $x = 1$, $y = 011$, we find $Y = 110$; hence gates 1 and 3 are unstable and we have a race. We use the General Multiple Winner (GMW) model [B-Y] for the analysis of races. Specifically, we assume that gate 1 may win the race (have a smaller delay than gate 3), or gate 3 may win the race, or both can change at the same time. Thus the next state of N_1 could be 111, 010, or 110, depending on the relative delays of gates 1 and 3. For each of these possible successors of the initial state 011 we now repeat the process. The result of this analysis is the diagram of Fig.2, where each state reachable from the initial state 011 is represented by a node, and an arrow from state a to state b indicates that b can be an immediate successor of a. Unstable gate outputs are shown underlined.

It can be seen that the graph has two cycles of length two and one of length one. In the first cycle (states 010 and 011), the variable y_1 is 0 in both states and is unstable in both states. Hence the network can exist in these two states only for a finite amount of time, for eventually y_1 will change. We call such a cycle a transient cycle [B-Y]. Similar remarks apply to the second cycle consisting of 110 and 111. Thus the only nontransient outcome of this transition is the stable state 101.

In general the binary analysis in the GMW model is exponential in the number s of gates because it is possible to have as many as 2^s states in a diagram such as Fig.2. Therefore this approach is impractical for large networks.

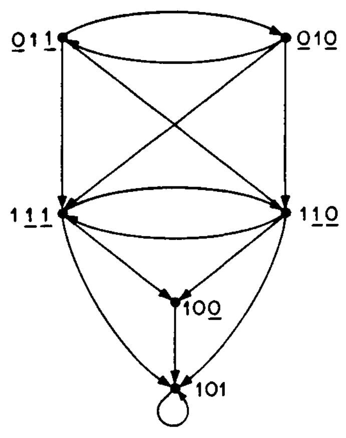

Figure 2. Binary analysis of N_1.

In contrast to this consider the following ternary approach. Starting with state $x = 0$, $y = 011$, change the input to $x = ½$, representing an unknown or changing signal. As a result of this unknown input x, gates 1 and 3 will have unknown values. In the second step, gate 2 will also become ½. This approach is summarized in Fig.3(a) and corresponds to Eichelberger's Procedure A [E, B-Y]. In this case the figure shows that all the gates may become unknown when the input is changing.

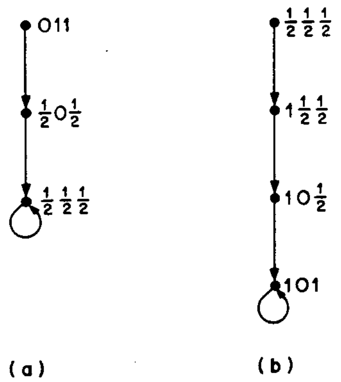

Figure 3. Ternary analysis of N_1: (a) Algorithm A; (b) Algorithm B.

To complete the ternary analysis we now apply the new input $x = 1$ to the ternary state ½ ½ ½ resulting from Procedure A, to see how much of the uncertainty introduced by the transient input can be removed when the final input value becomes known. First, y_1 will become 1 since the input $x = 1$ will force the output to 1, independently of the second input to the OR gate. Second, the output of the inverter will become 0 after y_1 becomes 1. Finally y_3 becomes 1 after y_2 changes to 0. This is summarized in Fig.3(b). Notice that the final outcome of the ternary algorithm is 101 which is precisely the nontransient outcome of the GMW analysis. The ternary Algorithms A and B are both linear in the number of gates.

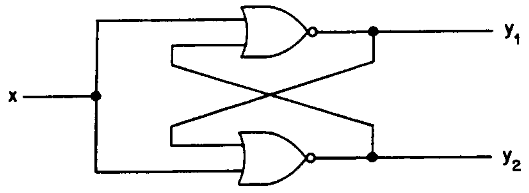

Figure 4. Network N_2.

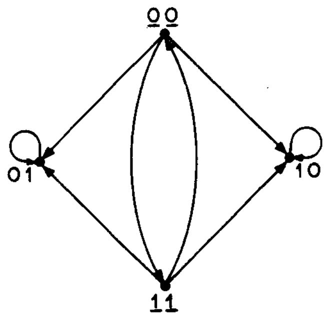

Figure 5. Binary analysis of N_2.

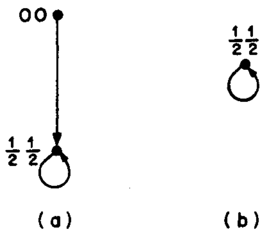

Figure 6. Ternary analysis of N_2: (a) Algorithm A; (b) Algorithm B.

Our second example shows what happens when the result of an input change does not lead to a unique stable state. The binary and ternary analyses of the NOR latch of Fig.4 are shown in Figs. 5 and 6, respectively. The initial state is $x = 1$, $y = 00$ and the input is changed to $x = 0$. The GMW analysis of Fig.5 shows three cycles: the stable states 01, and 10 and an oscillation (00, 11). The latter cycle is not transient like the ones of Fig.2. However, it is "match-dependent" [B-Y] in the sense that a network can only maintain such an oscillation if at each step the two gates have perfectly matched delays.

In the case of N_2 the ternary model of Fig.6 predicts an unknown final state. If one interprets ½ as meaning that the gate could either have the value 0 or the value 1 in any nontransient situation, then the ternary algorithm results are correct.

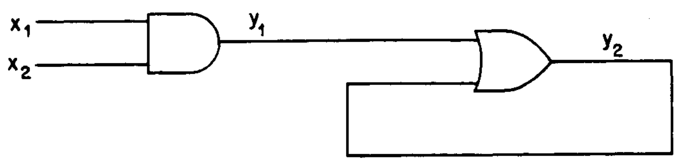

Figure 7. Network N_3.

Our final example [B-Y] for this section shows that the ternary results do not always correspond to the binary results, if one assumes that delays are associated only with gates. If the network N_3 of Fig.7 is started with $x = 01$, $y = 00$ and the input changed to $x = 10$, the binary analysis predicts no change in the state, i.e. $y = 00$. However, the ternary algorithm yields $y = 0½$. This discrepancy can be explained by assuming that wires, as well as gates, have delays. This leads to the conjecture that the ternary and binary results correspond properly if one allows appropriate wire delays. It is this problem that is settled in the present paper.

2. Gate Network Model

We will require a precise mathematical model of gate networks. We assume that the network N has n inputs, described by the vector $x = x_1, \cdots, x_n$ of *input variables*. We also assume that the network has s gates. We will represent each input and each gate by a node in a directed graph $G = (V, E)$, where V is the set of vertices or nodes and E is the set of edges. The edges will represent the connections among the input terminals and gates in the natural way. The input and gate nodes can be distinguished as follows. Nodes of indegree 0 are *input nodes*, and nodes of indegree ≥ 1 are called *gate nodes*. With each gate node we associate a boolean function g_i —the incoming edges to the gate node represent the arguments of g_i.

Sometimes it is convenient to treat all the nodes of G in the same fashion; for this reason we will use the vector $y = y_1, \cdots, y_{n+s}$ of *node labels* or *node variables*. Thus the vector y of node variables represent the *total state* (inputs and gate outputs) of the network. When it is necessary to distinguish between input and gate variables we replace y_i by x_i for $i = 1, \cdots, n$. However, for $i = n+1, \cdots, n+s$, y_i always denotes the output of gate i.

To illustrate these ideas, consider the network N_1 of Fig.8. The graph for N_1 is shown in Fig.9. It has 4 nodes: one input node labeled y_1 (and also x_1) and 3 gate nodes labeled y_2, y_3, y_4. The boolean functions corresponding to these gates are g_2, g_3 and g_4.

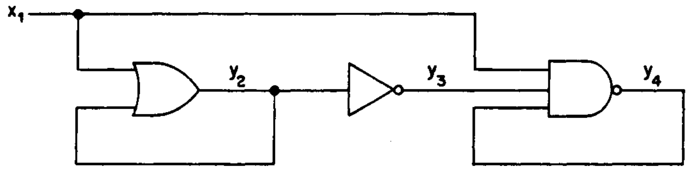

Figure 8. Network N_1.

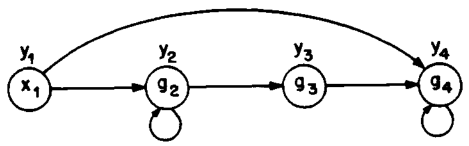

Figure 9. Graph of network N_1.

In summary, we are using the following model. A *gate network* is a directed labeled graph
$N = \langle G, x, y, g \rangle$,

where: $G = (V, E)$ is a finite directed *graph*,
$V = \{1, \cdots, n+s\}$ is the set of *vertices* of G,
$E \subseteq V \times V$ is the set of *edges* of G,
$x = x_1, \cdots, x_n$ is the vector of *input variables*,
$y = y_1, \cdots, y_{n+s}$ is the vector of *node labels* or *node variables*, and
$g = g_{n+1}, \cdots, g_{n+s}$ is a vector of boolean functions.

For an input node the variable y_i always takes the value of the external input x_i, i.e.

$$y_i = x_i \quad \text{for } i = 1, \cdots, n.$$

With each gate node j we associate the boolean function g_j. The arguments of g_j are all those node variables y_i such that $(i, j) \in E$; this represents the fact that node y_i is connected to an input of gate y_j. Note that, by definition, at most one such wire exists between y_i and y_j. Thus the indegree of node y_j is the number of arguments of g_j. For notational convenience we rename the arguments of g_j as follows. For each $(i, j) \in E$ the variable y_i is renamed w_{ij}. Suppose all the inputs of gate j are $w_{i_1 j}, \cdots, w_{i_{m_j} j}$; we will denote this vector simply by w_j. Thus, if the input vector for gate y_j is w_j, $g_j(w_j)$ is called the *excitation* of the gate and we have

$$g_j : B^{m_j} \rightarrow B,$$

where $B = \{0, 1\}$.

We illustrate these details for the network N_1 of Fig.8. The boolean functions associated with the three gates are:

$g_2(w_2) = g_2(w_{12}, w_{22}) = g_2(y_1, y_2) = g_2(x_1, y_2) = x_1 + y_2$,
$g_3(w_3) = g_3(w_{23}) = g_3(y_2) = y_2'$,
$g_4(w_4) = g_4(w_{14}, w_{34}, w_{44}) = g_4(y_1, y_3, y_4) = g_4(x_1, y_3, y_4) = (x_1 y_3 y_4)'$.

We assume that each gate has an arbitrary but finite delay. Thus the gate output y_j may differ from its excitation $g_j(w_j)$. If $y_j = g_j(w_j)$ we say that gate j is *stable;* otherwise it is *unstable.* A network N is said to be in a *stable total state* iff its inputs are fixed and all its gates are stable.

3. Binary Race Model

We now formally define the General Multiple Winner model. Let $b \in B^{n+s}$ be any total state and let $a = b_1, \cdots, b_n$ be a fixed input vector. Define $U(b)$ to be the set of unstable gates in b, i.e.

$$U(b) = \{i : n+1 \le i \le n+s, \text{ and } b_i \ne g_i(w_i)\}.$$

The GMW relation R_a defines the set of successors for any total state b. If b is stable, i.e. if $U(b) = \emptyset$, then the only possible successor is b. If b is unstable, then every state of the form $b^{(S)}$ is a successor of b, where $b^{(S)}$ is b with some components changed as follows. Let S be any subset of the set $U(b)$ of unstable gate variables; to obtain $b^{(S)}$ complement b_i iff $i \in S$. Formally, define the GMW relation R_a, on the set B^{n+s} of total states. For $b \in B^{n+s}$,

if $U(b) = \emptyset$ then $bR_a b$;
if $U(b) \ne \emptyset$ then $bR_a b^{(S)}$ for any $S \subseteq U(b)$, $S \ne \emptyset$.

Following [B-Y] we define the set 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. For example the set of cyclic states reachable from 001 in Fig.2 is $\{011, 010, 111, 110, 101\}$. Thus let

$$\text{cycl}(R_a, b) = \{c \in B^{n+s} : b\ R_a^* c \text{ and } c\ R_a^+ c\},$$

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

A cycle is called *transient* if there exists a gate i, $n+1 \leq i \leq n+s$ which is unstable in all of the states in the cycle and has the same value in all these states. For example, in Fig.2, the cycle consisting of 011 and 010 is transient. Let

$$\text{trans}(R_a, b) = \{c \in \text{cycl}(R_a, b) : c \text{ does not appear in any nontransient cycle}\}$$

and

$$\text{out}(R_a, b) = \text{cycl}(R_a, b) - \text{trans}(R_a, b).$$

The set $\text{out}(R_a, b)$ is the "outcome" of the binary 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. Note that match-dependent cycles are considered non-transient in this model.

4. Ternary Simulation Algorithm

We now describe the ternary simulation. For more details the reader should refer to [B-Y]. Let $T = \{0,1,½\}$. The values 0 and 1 represent the usual logic levels and ½ represents an unknown value. We will use the convention that variables like x_i, y_i, etc. which take values from $B = \{0,1\}$ will have corresponding variables $\mathbf{x}_i, \mathbf{y}_i$, etc. taking values from T. The *partial order* $\leq$ on T is defined by

$$\mathbf{t} \leq \mathbf{t} \text{ for all } \mathbf{t} \in T,$$
$$0 \leq ½ \text{ and } 1 \leq ½.$$

The statement $\mathbf{t} \leq \mathbf{r}$ means that whenever r_i is binary then t_i has the same binary value as r_i, but $\mathbf{r}$ may contain more unknown components (i.e. components with value ½). Thus $\mathbf{r}$ has more "uncertainty" than $\mathbf{t}$.

We write $\mathbf{t} < \mathbf{r}$ if $\mathbf{t} \leq \mathbf{r}$ and $\mathbf{t} \neq \mathbf{r}$. Also we extend the partial order $\leq$ to T^m in the usual way:

$$\mathbf{t} \leq \mathbf{r} \text{ iff } \mathbf{t}_i \leq \mathbf{r}_i \text{ for all } i = 1, \cdots, m.$$

We define the μ*-average* operation on nonempty subsets of the set B as follows:

$$\mu\{0\} = 0, \ \ \mu\{1\} = 1, \ \ \mu\{0,1\} = ½.$$

We extend the μ-average to nonempty sets of vectors from B^m by taking the "component-by-component" μ-average. Thus, if $A \subseteq B^m$, let $A_i = \{a_i : (a_1, \cdots, a_m) \in A\}$ for $1 \leq i \leq m$ be the set of the ith components of all the vectors in A. Then define

$$\mu A = (\mu A_1, \cdots, \mu A_m)$$

For example, $\mu\{(0,0),(0,1)\} = (0,½)$. Clearly $a \leq \mu A$ for every $a \in A$.

For any boolean function $f: B^m \rightarrow B^p$ we define its *ternary extension* $\mathbf{f}: T^m \rightarrow T^p$ by

$$\mathbf{f}(\mathbf{t}) = \mu\{f(a) : a \in B^m \text{ and } a \leq \mathbf{t}\}.$$

It follows that, for $t \in B^m$, $\mathbf{f}(t) = f(t)$, i.e. on binary vectors the ternary extension agrees with the original function. The ternary extension obeys the following monotonicity property [B-Y]:

$$\mathbf{t} \leq \mathbf{r} \text{ implies } \mathbf{f}(\mathbf{t}) \leq \mathbf{f}(\mathbf{r}).$$

First we describe how to compute the ternary excitation **next** for any total state $\mathbf{y} \in T^{n+s}$.

```
function next(y ∈ $T^{n+s}$) ∈ $T^{n+s}$;
begin
    for $j = 1$ to $n$ do
        $\mathbf{next}_j := \mathbf{y}_j$;
    for $j = n+1$ to $n+s$ do
        $\mathbf{next}_j := \mathbf{g}_j(\mathbf{w}_j)$;
end;
```

Here, $\mathbf{g}_j$ is the ternary extension of the boolean function g_j associated with gate j. Note that $\mathbf{w}_j$, as before, is a vector consisting of some components of $\mathbf{y}$. Also, one verifies that

$$\mathbf{y} \leq \tilde{\mathbf{y}} \text{ implies } \mathbf{next}(\mathbf{y}) \leq \mathbf{next}(\tilde{\mathbf{y}}).$$

This follows from the definition of **next** and the monotonicity property of the ternary extension.

The ternary simulation consists of Algorithms A and B described below.

Let $\hat{a}, b = \hat{a}_1, \cdots, \hat{a}_n, c_1, \cdots c_s \in B^{n+s}$ be any stable total state and $a = a_1, \cdots, a_n$ be the new input vector. Let $\mathbf{u} = \mu\{\hat{a}, a\}$.

Algorithm A

```
$h := 0$;
$\mathbf{y}^0 := \mathbf{u}, c$;
repeat
    $h := h+1$;
    $\mathbf{y}^h := \mathbf{next}(\mathbf{y}^{h-1})$;
until $\mathbf{y}^h = \mathbf{y}^{h-1}$;
```

It was shown in [B-Y] that Algorithm A always terminates, i.e. we have a sequence of p distinct ternary states, where $p < s$

$$\mathbf{y}^0, \mathbf{y}^1, \cdots, \mathbf{y}^{p-1}.$$

It was also shown in [B-Y] that Algorithm A can only "increase the uncertainty" in the network state, i.e.

$$\mathbf{y}^h < \mathbf{y}^{h+1} \text{ for } 0 \le h < p-1.$$

Note that $\mathbf{y}^{p-1} = \mathbf{u},\mathbf{r}$, for some $\mathbf{r} \in T^s$.

Next, starting with state $\mathbf{y}^{p-1} = \mathbf{u},\mathbf{r}$, we apply Algorithm B given below.

Algorithm B

```
$h := p$;
$\mathbf{y}^h := a, \mathbf{r}$;
repeat
    $h := h+1$;
    $\mathbf{y}^h := \mathbf{next}(\mathbf{y}^{h-1})$;
until $\mathbf{y}^h = \mathbf{y}^{h-1}$;
```

Algorithm B also terminates [B-Y], i.e. we have a sequence of q distinct ternary states, where $q < s$

$$\mathbf{y}^p, \mathbf{y}^{p+1}, \cdots, \mathbf{y}^{p+q-1},$$

and now the uncertainty decreases, i.e.

$$\mathbf{y}^h > \mathbf{y}^{h+1} \text{ for } p \le h < p+q-1.$$

Note that $\mathbf{y}^{p+q-1} = a, \mathbf{t}$ for some $\mathbf{t} \in T^s$.

5. Main Result

The following result was proved in [B-Y]. Let N be started in some stable state $\hat{a}, c$ and let the input vector change to a, i.e. let the new total state be $b = a, c$. Then the result $\mathbf{y}^B$ of the ternary simulation of N "covers" the non-transient states reachable from b in the GMW graph of N in the sense that:

$$\mu(\text{out}(R_a, b)) \le \mathbf{y}^B.$$

The example of Fig.7 shows that, in general, the two results are not equal. In this paper we show that the results become equal if appropriate delays are added. The main result is formally stated in this section; the following sections then contain the proof. The basic idea is to study a "delay complete" network obtained from the original network by adding an arbitrary but finite inertial delay to each wire. The new network will be called $\tilde{N}$. It contains the gates of N whose outputs are now labeled by the vector $\tilde{y}$, and the special "gates" corresponding to the added delays. The outputs of those delays will be described by the vector $\tilde{z}$.

Each delay may be viewed as a gate performing the identity function. The total state of $\tilde{N}$ is now $(\tilde{y}, \tilde{z})$, but we will compare only $\tilde{y}$ with the corresponding vector y in N. In the theorem below, if $\tilde{y}, \tilde{z}$ appears in a cycle of the GMW graph of $\tilde{N}$, then we will say that $\tilde{y}$ *belongs* to that cycle. Assuming that N and $\tilde{N}$ are started in corresponding initial conditions, the main result is:

Theorem 1 The ternary result $\mathbf{y}^B$ from Algorithm B for any network N is equal to the μ-average of all the binary vectors $\tilde{y}$ which belong to nontransient cycles reachable from the initial state of $\tilde{N}$ in the GMW model. Furthermore, there exists a nontransient cycle Z in the graph of the GMW-relation such that the μ-average of all the vectors $\tilde{y}$ belonging to that one cycle is equal to $\mathbf{y}^B$.

The proof proceeds as follows. In Section 6 we prove that the ternary simulation is insensitive to the addition of delays anywhere in the network N. In particular, the results of the ternary algorithm applied to N and $\tilde{N}$ agree in the gate variables. In Section 7 we characterize the result of Algorithm A and establish the first part of a sequence of states of $\tilde{N}$ that will eventually lead to the nontransient cycle Z of Theorem 1. In Section 8 we find the second part of that sequence, which is related to Algorithm B. In Section 9 we exhibit the nontransient cycle Z. Finally, in Section 10 we complete the proof of Theorem 1.

6. Ternary Simulation and Delays

In this section we show that the ternary simulation is insensitive to the addition of delays anywhere in the network. In the lemma below we insert one delay to N in the wire from node k to node m. The output of this delay will be designated $\tilde{z}_{km}$ and the resulting network will be $\overline{N} = <\tilde{G}, x, \tilde{y}, \tilde{z}_{km}, g>$, where the graph $\tilde{G}$ is G with an extra node added for the "gate" corresponding to the delay $\tilde{z}_{km}$, the input vector x is the same as in N, the total state of $\overline{N}$ is given by $(\tilde{y}, \tilde{z}_{km})$ and the vector g of boolean functions is the same as in N. The "gate function" of delay $\tilde{z}_{km}$ is simply y_k.

We say that Algorithm A or B is *consistent* for N and $\overline{N}$ if the final result is the same for $\tilde{y}$ and y, i.e. the output of the additional delay $\tilde{z}_{km}$ is simply ignored.

Lemma 1 The result of Algorithm A applied to any network N is consistent with the result of Algorithm A applied to the same network with one wire delay $\tilde{z}_{km}$ added.

Proof: Let $N = <G, x, y, g>$ be an arbitrary network and suppose $(k, m) \in E$. Let $\hat{y}^0 = (\hat{a}, b)$ be a stable total state of N and let $\mathbf{y}^0 = \mathbf{u}, b$, where $\mathbf{u} = \mu\{\hat{a}, a\}$, be the initial ternary state of N for Algorithm A. Let $\overline{N} = <\tilde{G}, x, \tilde{y}, \tilde{z}_{km}, g>$ be N with a delay $\tilde{z}_{km}$ inserted in the wire from node k to node m. The total state $\tilde{y}, \tilde{z}_{km} = \hat{a}, b, \hat{y}_k{}^0$ is stable in $\overline{N}$. Let $\mathbf{y}^0, \hat{y}_k{}^0$ be the initial ternary state of $\overline{N}$ for Algorithm A.

We will prove by induction on h that

$$\tilde{y}^h \leq y^h \leq \tilde{y}^{h+1} \text{ for all } h \geq 0.$$

In view of the fact that Algorithm A converges to its final value after a finite number of steps, the lemma will then follow.

If y_k does not change during Algorithm A the lemma is obviously true. Otherwise, assume that y_k becomes ½ at step r, $r \geq 0$. Note that $r = 0$ means that y_k is an input node.

For the basis, $h = 0$, we have $\tilde{y}^0 = y^0$. By the monotonicity of Algorithm A, $\tilde{y}^1 \geq \tilde{y}^0$. Thus we have

$$\tilde{y}^0 \leq y^0 \leq \tilde{y}^1.$$

Suppose now that $\tilde{y}^h \leq y^h \leq \tilde{y}^{h+1}$. Consider the input variables w_{ij} and $\tilde{w}_{ij}$ to all the gates of N. If $(i, j) \neq (k, m)$ then $\tilde{w}_{ij} = \tilde{y}_i$ and it is always true that $w_{ij} = y_i$. Hence

$$\tilde{w}_{ij}{}^h \leq w_{ij}^h \leq \tilde{w}_{ij}{}^{h+1}$$

by the induction hypothesis for the y_i. Also $w_{km} = y_k$ and $\tilde{w}_{km} = z_{km}$. We know that

$$w_{km}^h = \begin{cases} \hat{y}_k^0 & \text{if } h < r \\ ½ & \text{if } h \geq r \end{cases}$$

and

$$\tilde{w}_{km}^h = \begin{cases} \hat{y}_k^0 & \text{if } h < r+1 \\ ½ & \text{if } h \geq r+1 \end{cases}$$

Thus we also have

$$\tilde{w}_{km}^h \leq w_{km}^h \leq \tilde{w}_{km}^{h+1}.$$

Altogether we have

$$\tilde{w}_{ij}^h \leq w_{ij}^h \leq \tilde{w}_{ij}^{h+1} \text{ for all } (i, j) \in E.$$

It now follows by the monotonicity of g_j that

$$\tilde{y}_j^{h+1} = g_j(\tilde{w}_j^h) \leq g_j(w_j^h) = y_j^{h+1} \leq g_j(\tilde{w}_j^{h+1}) = \tilde{y}_j^{h+2}.$$

Hence the induction step goes through and the lemma holds. □

We are now ready to prove the main theorem of this section.

Theorem 2: Informally, the ternary algorithm is not affected by adding any delays to the network being analyzed. Formally, let $N = <G, x, y, g>$ be any network and let $\bar{N} = <\tilde{G}, x, \tilde{y}, \tilde{z}, g>$ be any network obtained from N by the addition of delays in any wires of N. Then the outcomes of Algorithms A and B are consistent for N and $\bar{N}$.

Proof:

Let Lemma 1^D be Lemma 1 with Algorithm A replaced by Algorithm B, with initial state y^{p-1} = u,r and new input $x = a$ for N, and appropriate initial state for $\bar{N}$ depending on the position of the added delay. One can verify that Lemma 1^D is proved by dual arguments, interchanging $\leq$ and $\geq$. We leave the details to the reader.

Altogether, Lemma 1 and Lemma 1^D show that ternary simulation yields consistent results for N and $\bar{N}$, when only one delay has been added.

It now follows by induction on the number of delays added to N that the theorem holds for any $\bar{N}$ as claimed. □

According to Theorem 2 any number of delays can be added to a network without changing the result of the ternary simulation. We now define the *delay-completion* of a network that gives a network which, in some sense, has all possible delays included. More specifically, let $N = <G, x, y, g>$ be a given network. We obtain $\tilde{N} = <\tilde{G}, x, \tilde{y}, \tilde{z}, g>$ by inserting a delay in every input line of every gate of N; $\tilde{N}$ is the delay-completion of N. Let d be the number of delays inserted. As in the case of w_j we define $\tilde{z}_j$ to be the vector $\tilde{z}_{i_1 j}, \cdots, \tilde{z}_{i_{m_j} j}$. This is the vector of input variables of gate j in network $\tilde{N}$.

To illustrate this definition we show the delay-completion of N_1 in Fig.10.

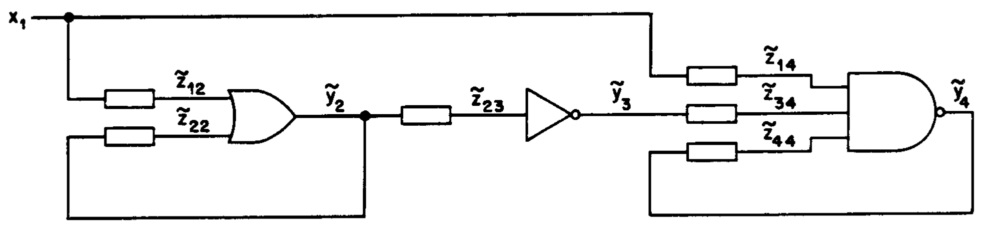

Figure 10. Delay-completion of N_1.

7. Characterization of Algorithm A

The results of this section are an adaptation of the work of Bryant [B83b]. In his model delays are associated only with wires and not with gates. However, the main idea for Theorem 3 is essentially the same. We assume that N and $\tilde{N}$ are started in corresponding initial conditions.

Theorem 3 The ternary result $\mathbf{y}^A$ from Algorithm A for any network N is equal to the μ-average of all the binary vectors $\tilde{y}$ reachable from the initial state of $\tilde{N}$ in the GMW model. Furthermore, there exists a state $\hat{y}, \hat{z}$ reachable from the initial state of $\tilde{N}$, in which each gate output that corresponds to ½ after Algorithm A in the ternary simulation of N, is the complement of its initial value, i.e. $\mathbf{y}^A$ is the μ-average of the initial state and $\hat{y}$.

Proof: We will prove the second part, since the first part follows immediately from the second. Clearly the second part implies that $\mathbf{y}^A$ is $\leq \mu Y$, where Y is the set of all the binary vectors reachable from the initial state of $\tilde{N}$. On the other hand, in [B-Y (Lemma 4)] it was shown that any binary state of $\tilde{N}$ reachable from the initial state is $\leq$ the result of Algorithm A for $\tilde{N}$. By Theorem 2, this result is consistent with that of Algorithm A for N. Hence $\mathbf{y}^A \geq \mu Y$, and the first part follows.

For convenience we now define the standard initial conditions for N and $\tilde{N}$ which will always be used to study a transition from a stable total state when the input vector changes.

Standard Initial Conditions

Network N:	$<G, x, y, g>$		
	$\hat{y}^0 = \hat{a}, b$	-	given stable total state
	$\hat{a}$	-	input vector before change
	a	-	input vector after change
	$\mathbf{u} = \mu(\hat{a}, a)$	-	ternary input vector for Algorithm A
	b	-	initial state of gate nodes of N
Network $\tilde{N}$:	$<\tilde{G}, x, \tilde{y}, \tilde{z}, g>$		
	$\hat{y}^0, \hat{z}^0$	-	stable total state corresponding to $\hat{y}^0$ of network N, where
	$\hat{z}_{ij}^0 = \hat{y}_i^0$		for all $(i, j) \in E$
	$\tilde{y}^0, \tilde{z}^0$	-	total state of $\tilde{N}$ after input change which is also the initial state for the GMW analysis of $\tilde{N}$, where
	$\tilde{y}^0 = a, b$		
	R_a	-	GMW relation for $\tilde{N}$

The second part of the theorem can be stated more formally as: let N be any network and let $\tilde{N}$ be its delay-completion, both started in the standard initial conditions. Let $\mathbf{y}^h$, $0 \leq h \leq p-1$ be the result of Algorithm A after h steps.

Then for each h there exist $\tilde{y}^{2h} \in B^{n+s}$ and $\tilde{z}^{2h} \in B^d$ such that

(i) $(\tilde{y}^0, \tilde{z}^0) R_a^{2h} (\tilde{y}^{2h}, \tilde{z}^{2h})$,

(ii) $\mathbf{y}_j^h = ½$ implies $\tilde{y}_j^{2h} = (\hat{y}_j^0)'$ for $1 \leq j \leq n+s$,

(iii) $\mathbf{y}_j^h = \hat{y}_j^0$ implies $\tilde{y}_j^{2h} = \hat{y}_j^0$ and $\tilde{z}_j^{2h} = \tilde{z}_j^0$ for $1 \leq j \leq n+s$.

We proceed by induction on h. The reader may find it useful to follow the construction in the proof of the theorem in parallel with the construction after Fig.11 for network N_4 of Fig.11.

Basis, $h=0$: One verifies that $\tilde{y}^0, \tilde{z}^0$ satisfies conditions (i)-(iii).

Induction step: Assume that $\tilde{y}^{2h}, \tilde{z}^{2h}$ has been constructed and consider $\mathbf{y}_j^{h+1}$. There are two cases.

(a) $\mathbf{y}_j^{h+1} = ½$ and $\mathbf{y}_j^h = \hat{y}_j^0$. In other words $\mathbf{y}_j$ had the original binary value for the first h steps and changed to ½ in step $h+1$. There must exist at least one such j in every step h, $0 \leq h \leq p-1$. Note that $h+1 \geq 1$ implies that y_j must be a gate output, since all the input node variables became ½ in step 0. Since $\mathbf{y}_j^{h+1} = \mathbf{g}_j(\mathbf{w}_j^h) = ½$, and $\hat{y}_j^0 = g_j(\hat{w}_j^0) \in B$ and by the definition of the ternary extension of g_j, there exists $v_j^h \in B^{m_j}$ such that $g_j(v_j^h) = (\hat{y}_j^0)'$, and $v_j^h \leq \mathbf{w}_j^h$. Note also that $\hat{w}_j^0 \leq \mathbf{w}_j^h$. If $v_{ij}^h \neq \hat{w}_{ij}^0$ (at least one such case must exist), then $\mathbf{w}_{ij}^h = ½$, i.e. $\mathbf{y}_i^h = ½$. By the induction hypothesis $\tilde{y}_i^{2h} = (\hat{y}_i^0)'$ by (ii). Also $\tilde{z}_{ij}^{2h} = \tilde{z}_{ij}^0$ because $\mathbf{y}_j^h = \hat{y}_j^0$ and (iii) applies. But $\tilde{z}_{ij}^0 = \hat{y}_i^0$, and hence the delay $\tilde{z}_{ij}$ is unstable in step $2h$, i.e.

$$\tilde{y}_i^{2h} \neq \tilde{z}_{ij}^{2h} \text{ for all } i \text{ such that } v_{ij}^h \neq \hat{w}_{ij}^0.$$

Let $\tilde{z}_{ij}^{2h+1} = \tilde{y}_i^{2h}$ if $v_{ij}^h \neq \hat{w}_{ij}^0$ and $\tilde{z}_{ij}^{2h+1} = \tilde{z}_{ij}^{2h}$ otherwise. Note that $\tilde{z}_j^{2h+1} = v_j^h$, and hence gate y_j will become unstable in step $2h+1$.

(b) If case (a) does not hold, variable $\mathbf{y}_j$ has not changed from step h to step $h+1$. Let $\tilde{z}_j^{2h+1} = \tilde{z}_j^{2h}$. Note that now gate y_j will be stable in step $2h+1$.

Now define $\tilde{y}^{2h+1} = \tilde{y}^{2h}$. It follows that

$$(\tilde{y}^{2h}, \tilde{z}^{2h})\ R_a\ (\tilde{y}^{2h+1}, \tilde{z}^{2h+1}).$$

Next define $\tilde{y}^{2h+2}, \tilde{z}^{2h+2}$ as follows:

$$\tilde{y}_j^{2h+2} = g_j(\tilde{z}_j^{2h+1}) \text{ for all } j,$$

and

$$\tilde{z}^{2h+2} = \tilde{z}^{2h+1}.$$

The reader can now verify that $\tilde{y}^{2h+2}, \tilde{z}^{2h+2}$ satisfies (i)-(iii). Therefore the induction step goes through and the theorem holds. □

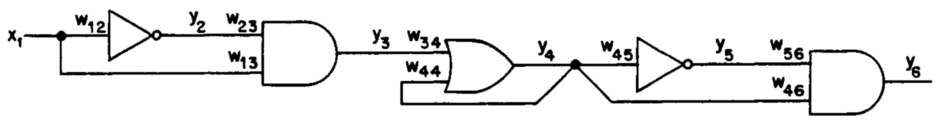

Figure 11. Network N_4.

To illustrate the construction in the proof of Theorem 3 consider network N_4 of Fig.11. Let

$$\hat{a} = 0, \quad a = 1, \quad b = 10010.$$

Then $\hat{y}^0 = 010010$ and $\mathbf{u} = ½$. Also let

$$\begin{array}{l}
\tilde{z}_2 = \tilde{z}_{12} = 0 \\
\tilde{z}_3 = \tilde{z}_{13}, \tilde{z}_{23} = 01 \\
\tilde{z}_4 = \tilde{z}_{34}, \tilde{z}_{44} = 00 \\
\tilde{z}_5 = \tilde{z}_{45} = 0 \\
\tilde{z}_6 = \tilde{z}_{46}, \tilde{z}_{56} = 01
\end{array}$$

Hence

$$\tilde{y}^0, \tilde{z}^0 = 110010,\ 0\ 01\ 00\ 0\ 01$$

The construction described in the proof of Theorem 3 is shown below. Note that all the gates of $\tilde{N}$ are stable in $\tilde{y}^6$, but there are some unstable delays. This fact will be used in a later section.

$$\begin{array}{lll}
\mathbf{y}^0 = \tfrac{1}{2}10010 & \tilde{y}^0, \tilde{z}^0 = & 110010,\ \underline{0}\ \underline{0}1\ 00\ 0\ 01 \\
 & \tilde{y}^1, \tilde{z}^1 = & 1\underline{10}010,\ 1\ 11\ 00\ 0\ 01 \\
\mathbf{y}^1 = \tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}010 & \tilde{y}^2, \tilde{z}^2 = & 101010,\ 1\ 1\underline{1}\ \underline{0}0\ 0\ 01 \\
 & \tilde{y}^3, \tilde{z}^3 = & 101\underline{0}10,\ 1\ 1\underline{1}\ 10\ 0\ 01 \\
\mathbf{y}^2 = \tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}10 & \tilde{y}^4, \tilde{z}^4 = & 101110,\ 1\ 1\underline{1}\ 1\underline{0}\ \underline{0}\ \underline{0}1 \\
 & \tilde{y}^5, \tilde{z}^5 = & 1011\underline{10},\ 1\ 1\underline{1}\ 1\underline{0}\ 1\ 11 \\
\mathbf{y}^3 = \tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2} & \tilde{y}^6, \tilde{z}^6 = & 101101,\ 1\ 1\underline{1}\ 1\underline{0}\ 1\ 1\underline{1}
\end{array}$$

8. Algorithm B - Definite Nodes

Let N be any network started in the standard initial conditions. Let $\mathbf{y}^B$ be the result of Algorithm B. The *indefinite nodes* or *indefinite gates* of N are those gates whose outputs are ½ in $\mathbf{y}^B$; the other nodes will be called *definite*.

Assuming that there is at least one indefinite gate j (i.e. Algorithm B does *not* yield a binary result) that gate must have at least one input w_{ij} where y_i is also an indefinite gate (possibly with $i=j$). Otherwise all inputs to gate j would be binary and its output could not be ½. Since the network N is finite we must have at least one cycle of indefinite nodes; such a cycle will be called indefinite.

Consider now $\tilde{N}$; the indefinite gates of $\tilde{N}$ are the same as those of N. Any delay between two indefinite gates will be called *indefinite*. Eventually we want to show that, if the result of Algorithm B contains at least one ½, there exists a nontransient cycle of length ≥ 2 (i.e. an oscillation) in the graph of the relation R_a for $\tilde{N}$ such that all indefinite gates "take part" in that oscillation, i.e. each gate variable will take on both values 0 and 1 in the cycle. Furthermore that cycle is reachable from the initial state of $\tilde{N}$.

Theorem 4 Let N and $\tilde{N}$ be started in the standard initial conditions and let $\mathbf{y}^B$ be the result of Algorithm B. There exists a state $\tilde{y}, \tilde{z}$ of $\tilde{N}$ reachable from the initial state such that:

(1) All definite gates have the same (binary) value in $\tilde{y}$ as they do in $\mathbf{y}^B$. Furthermore, they are all stable, as are all the delays in the wires leaving definite nodes.

(2) There is at least one unstable wire delay in each indefinite cycle of $\tilde{N}$.

Proof: Let $\tilde{y}^{2p-2}, \tilde{z}^{2p-2}$ be the total state of $\tilde{N}$ constructed in the proof of Theorem 3. Let $\mathbf{y}_j^h$, $p \leq h \leq p+q-1$ be the result of Algorithm B after $h-p$ steps. We first claim that there exist $\tilde{y}^{2h} \in B^{n+s}$ and $\tilde{z}^{2h} \in B^d$ such that

(i) $(\tilde{y}^{2p-2}, \tilde{z}^{2p-2})\, R_a^* \,(\tilde{y}^{2h}, \tilde{z}^{2h})$,

(ii) $\mathbf{y}_j^h \in B$ implies $\tilde{y}_j^{2h} = \mathbf{y}_j^h$ for $1 \leq j \leq n+s$,

(iii) $\mathbf{y}_j^h = ½$ implies $\tilde{y}_j^{2h} = \tilde{y}_j^{2p-2}$ and $\tilde{z}_j^{2h} = \tilde{z}_j^{2p-2}$ for $1 \leq j \leq n+s$.

We proceed by induction on h. The reader may find it useful to follow the construction in the proof in parallel with the construction shown right after this proof for network N_4 of Fig.11.

Basis, $h=p$: Let $\tilde{y}^{2p} = \tilde{y}^{2p-2}$ and $\tilde{z}^{2p} = \tilde{z}^{2p-2}$. Claims (i) and (iii) follow immediately. For (ii), i.e. $\mathbf{y}_j^p \in B$, there are two cases.

(a) $\mathbf{y}_j^{p-1} \in B$. But by Theorem 3, (iii) in proof, $\tilde{y}_j^{2p-2} = \mathbf{y}_j^{p-1}$ and from the definition of Algorithm B it follows that $\mathbf{y}_j^{p-1} \in B$ implies that $\mathbf{y}_j^p = \mathbf{y}_j^{p-1}$ and hence $\tilde{y}_j^{2p} = \tilde{y}_j^{2p-2} = \mathbf{y}_j^{p-1} = \mathbf{y}_j^p$.

(b) $\mathbf{y}_j^{p-1} = ½$. This implies that j is an input node of N, and from Theorem 3, (ii) in proof, it follows that $\tilde{y}_j^{2p} = \tilde{y}_j^{2p-2} = (\hat{y}_j^0)'$. But this is the new input value assigned to $\mathbf{y}_j^p$ by Algorithm B and hence $\tilde{y}_j^{2p} = \mathbf{y}_j^p$.

Induction step: Assume that $\tilde{y}^{2h}$, $\tilde{z}^{2h}$ $(h \geq p)$ has been constructed and consider $\mathbf{y}_j^{h+1}$. There are two cases.

(a) $\mathbf{y}_j^h = ½$ and $\mathbf{y}_j^{h+1} \in B$. In other words $\mathbf{y}_j$ was ½ for the first $h-p$ steps in Algorithm B and changed to a binary value in $\mathbf{y}^{h+1}$. Note that $h+1>p$ implies that y_j must be a gate output, since all the input node variables became binary in $\mathbf{y}^p$. Let $\tilde{z}_{ij}^{2h+1} = \tilde{y}_i^{2h}$ for $i = i_1, \cdots, i_{m_j}$. This corresponds to changing all unstable wire delays to gate j (if any). Now $\tilde{z}_{ij}^{2h+1} \leq \mathbf{w}_{ij}^h$ because $\tilde{z}_{ij}^{2h+1} = \tilde{y}_i^{2h}$ and by the induction hypothesis (iii) we have that if $\mathbf{y}_i^h \in B$ then $\tilde{y}_i^{2h} = \mathbf{y}_i^h$. Note that $\mathbf{y}_j^{h+1} = \mathbf{g}_j(\mathbf{w}_j^h) \in B$ implies that $g_j(\tilde{z}_j^{2h+1}) = \mathbf{y}_j^{h+1}$.

(b) If case (a) does not hold, variable $\mathbf{y}_j$ has not changed from step h to step $h+1$. Let $\tilde{z}_j^{2h+1} = \tilde{z}_j^{2h}$. Note that now gate y_j is stable in step $2h+1$.

Now define $\tilde{y}^{2h+1} = \tilde{y}^{2h}$. It follows that either

$$\tilde{y}^{2h}, \tilde{z}^{2h} = \tilde{y}^{2h+1}, \tilde{z}^{2h+1}$$

or

$$(\tilde{y}^{2h}, \tilde{z}^{2h}) \; R_a \; (\tilde{y}^{2h+1}, \tilde{z}^{2h+1}).$$

In either case

$$(\tilde{y}^{2h}, \tilde{z}^{2h}) \; R_a^* \; (\tilde{y}^{2h+1}, \tilde{z}^{2h+1}).$$

Next define $\tilde{y}^{2h+2}, \tilde{z}^{2h+2}$ as follows:

$$\tilde{y}_j^{2h+2} = g_j(\tilde{z}_j^{2h+1}) \text{ for all } j,$$

and

$$\tilde{z}^{2h+2} = \tilde{z}^{2h+1}.$$

The reader can now verify that $\tilde{y}^{2h+2}$, $\tilde{z}^{2h+2}$ satisfies (i)-(iii). Therefore the induction step goes through and the claim holds.

Now let

$$\tilde{y}^B = \tilde{y}^{2(p+q-1)}$$

and

$$\tilde{z}_{ij}^{B} = \begin{cases} \tilde{y}_i^{2(p+q-1)} & \text{for all } i \text{ such that } i \text{ is a definite node} \\ \tilde{z}_{ij}^{2(p+q-1)} & \text{otherwise.} \end{cases}$$

It follows trivially that all wire delays *from* definite nodes are stable in $\tilde{y}^B, \tilde{z}^B$, but furthermore all definite gates are also stable in $\tilde{y}^B, \tilde{z}^B$. This is because (ii) implies that $\tilde{z}_j^B \leq \mathbf{w}_j^{p+q-1}$ and (by the definition of Algorithm B) it follows that $\tilde{y}_j^B = \tilde{y}_j^{2(p+q-1)} = \mathbf{y}_j^{p+q-1} = \mathbf{g}_j(\mathbf{w}_j^{p+q-1})$ for all definite gates. In other words, $\tilde{y}^B$, $\tilde{z}^B$ is a total state reachable from the initial total state $\tilde{y}^0$, $\tilde{z}^0$ in the GMW relation such that all definite gates are stable and have the binary value predicted by the result of Algorithm B. Furthermore all wire delays from definite nodes are also stable. This completes the proof of Part (1).

For Part (2), clearly it is sufficient to prove the claim for each simple indefinite cycle, where a cycle is simple if it has no repeated nodes except for the first and the last node in the cycle. Let C be an arbitrary simple indefinite cycle in $\tilde{N}$. A gate j in C is said to be *initiating* iff no other gate in C becomes ½ in Algorithm A before gate j. Clearly there must be at least one initiating gate in each simple indefinite cycle. Let j be an initiating gate in C. Assume gate j became ½ at step r of Algorithm A. Note that $r \geq 1$ since an input node cannot be indefinite. Now since node j is in C there must exist a predecessor to j in the cycle, say node i. Note that $i = j$ is permitted. Consider $\tilde{z}_{ij}$. Since j is an initiating gate we have that $\mathbf{y}_i^{r-1} = \hat{y}_i^0$ and hence according to Theorem 3, (iii) in proof, $\tilde{y}_i^{2(r-1)} = \hat{y}_i^0$, and also $\tilde{z}_{ij}^{2(r-1)} = \hat{y}_i^0$. Note that this implies that v_{ij}^{r-1}, as defined in the proof of Theorem 3, satisfies $v_{ij}^{r-1} = \hat{y}_i^0$, i.e. the wire delay from the predecessor to an initiating gate has the original "old" binary value when the initiating gate changes.

After this, $\tilde{z}_{ij}$ is never changed again to construct $\tilde{z}^B$ (since j is an indefinite gate). However we know that $\tilde{y}_i^{2p-2} = (\hat{y}_i^0)'$ (Theorem 3, (ii) in proof) and by the construction of $\tilde{y}^B$, $\tilde{z}^B$ that $\tilde{y}_i^B = \tilde{y}_i^{2p-2}$. It follows that $\tilde{y}_i^B = (\hat{y}_i^0)'$ and $\tilde{z}_{ij}^B = \hat{y}_i^0$ and hence $\tilde{z}_{ij}^B$ is unstable. This complete the proof of part 2. □

The first part of the construction of Theorem 4 for network N_4 is shown below.

$$\begin{array}{lll}
\mathbf{y}^4 = 1\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2} & \tilde{y}^8, \tilde{z}^8 = & 101101,\ 1\ 1\underline{1}\ 1\underline{0}\ 1\ 1\underline{1} \\
 & \tilde{y}^9, \tilde{z}^9 = & 101101,\ 1\ 1\underline{1}\ 1\underline{0}\ 1\ 1\underline{1} \\
\mathbf{y}^5 = 10\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2} & \tilde{y}^{10}, \tilde{z}^{10} = & 101101,\ 1\ 1\underline{1}\ 1\underline{0}\ 1\ 1\underline{1} \\
 & \tilde{y}^{11}, \tilde{z}^{11} = & 10\underline{1}101,\ 1\ 10\ 1\underline{0}\ 1\ 1\underline{1} \\
\mathbf{y}^6 = 100\tfrac{1}{2}\tfrac{1}{2}\tfrac{1}{2} & \tilde{y}^{12}, \tilde{z}^{12} = & 100101,\ 1\ 10\ \underline{10}\ 1\ 1\underline{1}
\end{array}$$

To illustrate the construction of $\tilde{y}^B, \tilde{z}^B$ for N_4 we have

$$\tilde{y}^B, \tilde{z}^B = 100\underline{1}01,\ 1\ 10\ 0\underline{0}\ 1\ 1\underline{1}.$$

Note that wire delay $\tilde{z}_{44}$ is unstable. This will later be used to start an oscillation in the indefinite subnetwork. Note also that all definite nodes have the same values as in $\mathbf{y}^B$.

9. Algorithm B - Indefinite Gates

The main result of this section is captured in the following theorem.

Theorem 5 Let N be a network started in the standard initial conditions. Let $\tilde{y}^B, \tilde{z}^B$ be a total state of $\tilde{N}$ as defined at the end of Section 8. Then there is a nontransient cycle in the GMW graph, reachable from $\tilde{y}^B, \tilde{z}^B$, such that all indefinite gates of $\tilde{N}$ are oscillating.

To simplify the proof of Theorem 5, the following two definitions are useful. Define a state $\tilde{y}, \tilde{z}$ of $\tilde{N}$ to be *consistent* with $\mathbf{y}^B$ iff $\tilde{y} \leq \mathbf{y}^B$, and all the definite nodes and all the delays leaving definite nodes are stable in $\tilde{N}$. Also, a state $\tilde{y}, \tilde{z}$ is *loop-unstable* iff there is at least one unstable wire delay in each simple indefinite cycle of $\tilde{N}$.

We now proceed as follows. Starting with a total state $(\tilde{y}, \tilde{z})$ we first exhibit a sequence of total states of $\tilde{N}$

$$(\tilde{y}, \tilde{z}) = (\tilde{y}^0, \tilde{z}^0),\ \cdots,\ (\tilde{y}^m, \tilde{z}^m),$$

where m is the number of indefinite gates, and in $\tilde{y}^k$, exact k indefinite gate outputs have complementary values to those in $\tilde{y}$ and the other indefinite gate nodes are the same as in $\tilde{y}$. For convenience, we will say that k indefinite nodes have been "marked" in this way. By repeating this process of marking (i.e. complementing) all the indefinite gates we show the existence of an oscillation involving all the indefinite nodes.

Lemma 2 Let N be a network started in the standard initial conditions and $\mathbf{y}^B$ be the result of Algorithm B. Let $\tilde{y}, \tilde{z}$ be any total state of $\tilde{N}$ consistent with $\mathbf{y}^B$ and loop-unstable. Assume that some, but not all, indefinite nodes of $\tilde{N}$ are marked. Assume also that every wire delay between a marked and an unmarked indefinite node is unstable. Then there exists at least one unmarked indefinite node j, such that all indefinite wire delays to j are unstable.

Proof: Consider the directed graph $G' = (V', E')$ where

$V' \subseteq V$, $i \in V'$ iff i is an indefinite gate node and

$E' = \{(i,j) \in V' \times V' : (i,j) \in E \text{ and } \tilde{z}_{ij} \text{ is stable}\}$.

G' can be obtained from the graph G by retaining only the indefinite gate nodes and those indefinite edges that corresponds to stable indefinite delays. G' has two important properties:

(i) there is no edge from a marked node to an unmarked node, and

(ii) there is no cycle in G'.

Both properties follow trivially from the construction of G' and the assumptions in the Lemma.

Now consider a reverse path in G'. Start at some unmarked node $k \in V'$ and traverse G' backwards. From (ii) and the fact that G' is finite it follows that a reverse path in G' started at node k must stop at some node, say j. By property (i) it follows that j must be an unmarked node. Furthermore, since each indefinite gate has at least one input wire from an indefinite gate, it follows that all indefinite wire delays to j must be unstable; otherwise the reverse path could not have stopped at j. Hence the Lemma holds. □

In the following Lemma we will show how instabilities can be "moved". The idea is that if all indefinite wire delays *to* a gate are unstable then it is possible in the GMW relation to find a state reachable from the present state such that all indefinite wire delays *leaving* the gate are unstable.

Lemma 3 Let N be a network started in the standard initial conditions. Let $\tilde{y}$, $\tilde{z}$ be a total state of $\tilde{N}$ consistent with $\mathbf{y}^B$ and loop-unstable. If all indefinite wire delays to indefinite gate j are unstable, then there exists a total state $\tilde{y}^C$, $\tilde{z}^C$, reachable from $\tilde{y}$, $\tilde{z}$, consistent with $\mathbf{y}^B$ and loop-unstable, such that

(i) $\tilde{y}_j^C = (\tilde{y}_j)'$, and

(ii) all indefinite wire delays leaving gate j are unstable.

Proof: Consider gate j. Two cases are possible.

a) Gate j is stable in $\tilde{y}$, $\tilde{z}$, i.e. $\tilde{y}_j = g_j(\tilde{z}_j)$. Since j is an indefinite gate, after Algorithm B for N, we have $\mathbf{y}_j^B = \mathbf{g}_j(\mathbf{w}_j^B) = ½$. Note that $\tilde{z}_j \le \mathbf{w}_j^B$ by construction of $\tilde{y}^B$, $\tilde{z}^B$. By the definition of the ternary extension of g_j, there must exist a $v_j \in B^{m_j}$ such that $g_j(v_j) \neq g_j(\tilde{z}_j)$, and $v_j \le \mathbf{w}_j^B$. If $v_{ij} \neq \tilde{z}_{ij}$ then we must have $\mathbf{w}_{ij}^B = ½$, i.e. $\mathbf{y}_i^B = ½$, and hence $\tilde{z}_{ij}$ is an indefinite wire delay. We want to reach a state in which gate j is unstable. We will do this, if we set the inputs of gate j to the vector v_j. This can be done because all the indefinite wire delays to j are assumed to be unstable. Therefore, define

$$\tilde{z}_{kl}^D = \begin{cases} \tilde{y}_i & \text{if } k = i,\ l = j \text{ and } v_{ij} \neq \tilde{z}_{ij} \\ \tilde{z}_{kl} & \text{otherwise} \end{cases}$$

and

$\tilde{y}^D = \tilde{y}$.

b) Gate j is unstable. Define $\tilde{y}^D = \tilde{y}$ and $\tilde{z}^D = \tilde{z}$.

In either case we have that $(\tilde{y}, \tilde{z})\ R_a^*\ (\tilde{y}^D, \tilde{z}^D)$ and gate j is unstable in $\tilde{y}^D, \tilde{z}^D$.

Now we will simultaneously change gate j and all indefinite delays leaving gate j which are unstable. In this way all the indefinite delays leaving gate j will become unstable after the change. Therefore define $\tilde{y}^C, \tilde{z}^C$ as follows:

$$\tilde{z}_{kl}^C = \begin{cases} \tilde{y}_j^D & \text{if } k = j \text{ and gate } l \text{ is an indefinite gate} \\ \tilde{z}_{kl}^D & \text{otherwise} \end{cases}$$

and

$$\tilde{y}_k^C = \begin{cases} g_j(\tilde{z}_j^D) & \text{if } k = j \\ \tilde{y}_k^D & \text{otherwise} \end{cases}$$

Condition (i) follows from the fact that $\tilde{y}_j^C = g_j(\tilde{z}_j^D) = g_j(v_j) = (\tilde{y}_j)'$. Condition (ii) follows from (i) and the fact that if k is an indefinite gate then $\tilde{z}_{jk}^C = \tilde{y}_j^D = \tilde{y}_j$. Hence the Lemma holds. □

Lemma 4 Let N be a network started in the standard initial conditions. Let $\tilde{y}, \tilde{z}$ be a total state of $\tilde{N}$ which is consistent with $\mathbf{y}^B$ and loop-unstable. Then there exists a total state $\hat{y}, \hat{z}$, reachable from $\tilde{y}, \tilde{z}$, which is consistent with $\mathbf{y}^B$, loop-unstable, and such that all indefinite gates in $\tilde{y}$ and $\hat{y}$ have complementary values.

Proof: We proceeds by induction on the number of indefinite nodes which have been marked, i.e. complemented.

Claim: There exists a state $\tilde{y}^k, \tilde{z}^k$, reachable from $\tilde{y}, \tilde{z}$, with k nodes marked. Furthermore, this state is consistent with $\mathbf{y}^B$, loop-unstable and all indefinite wire delays between marked nodes and unmarked nodes are unstable.

The basis, $k=0$, follows trivially. Suppose the claim holds for k, $k \geq 0$. By Lemma 2 it follows that there exists an unmarked indefinite node j, such that all indefinite wire delays to j are unstable. But Lemma 3 guarantees the existence of a state $\tilde{y}^{k+1}, \tilde{z}^{k+1}$, reachable from $\tilde{y}^k, \tilde{z}^k$, consistent with $\mathbf{y}^B$, loop-unstable and with node j complemented. We now mark node j, and note that all indefinite wire delays between marked nodes and unmarked nodes are still unstable. Hence the induction step goes through and the lemma holds. □

Proof of Theorem 5: Since Lemma 4 can be applied any number of times and there is only a finite number of possible total states there must exist a cycle in the graph. Also by the construction of Lemma 4 it follows that each indefinite gate in $\tilde{N}$ will oscillate. Furthermore, since all definite gates are stable, all wire delays from definite gates are stable, no wire delays to definite gates are changed and all indefinite gates are oscillating it follows trivially that the cycle cannot be transient. □

10. The Conjecture

We are now in a position to prove Theorem 1.

Proof of Theorem 1: Let $Y = \{\tilde{y} : \tilde{y}, \tilde{z} \in \text{out}(R_a, \tilde{y}^0, \tilde{z}^0)\}$. In Theorem 5 we showed that $\mathbf{y}^B \leq \mu Y$. In [B-Y] it was shown that any binary state of $\tilde{N}$ in $\text{out}(R_a, \tilde{y}^0)$ is $\leq$ the result of Algorithm B for $\tilde{N}$. But, by Theorem 2, this result is consistent with that of Algorithm B for N. Hence $\mathbf{y}^B \geq \mu Y$, and $\mathbf{y}^B = \mu Y$. Also, the constructions in Theorems 3, 4, and 5 show the existence of a single cycle Z such that $\mathbf{y}^B = \mu Z$. □

The following is a consequence of Theorem 1. Whenever a network N has a critical race, the network $\tilde{N}$ has an oscillation involving the gates that take part in the race.

The characterization obtained in Theorem 1 does not quite apply to the original conjecture which used a network N^c, described below, instead of $\tilde{N}$. The conjecture network N^c consists of the given network N to which delays have been added in all the input lines and in all the fan-out connections. There are two main differences between N^c and $\tilde{N}$:

1) In case an input x_i fans out to two or more gates, N^c has one delay associated with the input line x_i and additional delays in each fan-out connection from x_i. The network $\tilde{N}$ only has the fan-out delays.
2) In case the output of a gate i is connected to only one gate j (with $i = j$ possible), N^c has no delay in that connection, whereas in $\tilde{N}$ delays are inserted uniformly in all gate-input lines.

First we will show that the extra line delays of N^c as described in 1) above are not necessary. Let $N = \langle G, x, y, g \rangle$ be any network. A node $i \in V$ is said to be *singular* iff the outdegree of i is 1, i.e. the output of node i is only connected to one gate node.

Lemma 5 Let N be any network, let N^c be defined as above, and let N^d be N^c after the removal of all input delays corresponding to input nodes which are not singular. If N^c and N^d are started in the appropriate standard initial conditions, the outcomes of the GMW analyses of N^c and N^d will be consistent with respect to the nodes of N^d.

Proof: Consider N^d with only one extra delay z_{ii} added in input line x_i. If x_i does not change, the behavior of the two networks is identical. If x_i does change, the variable z_{ii} is unstable and can change only once. It must change before the network reaches a nontransient cycle, but after that the network behaves like N^d.

The lemma now follows by induction on the number of delays added to N^d to get N^c. □

From now on we consider the network N^d rather than N^c. Until now we have not explicitly defined any output of network N. One possible interpretation is that the output of every gate is an external output. In that case gates originally considered singular in N^d have an extra connection and are no longer singular. Thus delays will be present in all the lines leaving such gates, and hence all gate-input lines will have delays as in $\tilde{N}$. Therefore, under this assumption $\tilde{N}$ and N^d coincide, and Theorem 1 constitutes a proof of the conjecture.

One could make a different interpretation, namely that only some of the gate nodes are external output nodes. It is then reasonable to also assume that the network has no "useless" gates, i.e. that in the graph G there is a path from every gate node to some output node. This implies that there must be at least one gate node with outdegree ≥ 2 in every simple cycle of G. Now the only way that N^d differs from $\tilde{N}$ is the fact that in N^d a gate-input line from a singular node does not have a delay. For example, refer to $\tilde{N}_1$ of Fig.10. The corresponding network N_1^d, assuming 4 is the only output node, would not have delay $\tilde{z}_{34}$. The proof of the conjecture in this interpretation will be described in another paper.

Acknowledgment

The authors wish to thank Professor Michael Yoeli of the Department of Computer Science, Technion, Haifa, Israel for his many useful comments and suggestions regarding this paper.

References

[B84] R.E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems", *IEEE Transactions on Computers*, Vol. C-33, No. 2, Feb. 1984.

[B83a] R.E. Bryant, "Race Detection in MOS Circuits by Ternary Simulation", In F. Anceau and E.J. Aas (eds.), *VLSI '83*, pp. 85-95, Elsevier Science Publishers B. V. (North-Holland).

[B83b] R.E. Bryant, "Toward a Proof of the Brzozowski-Yoeli Conjecture on Ternary Simulation", Unpublished manuscript, Dec. 1983.

[B-Y] 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.

[E] E.B. Eichelberger, "Hazard Detection in Combinational and Sequential Switching Circuits", *IBM J. Res. Dev.*, Vol. 9, pp. 90-99, Mar. 1965.