RAIRO-Inf. Theor. Appl. **39** (2005) 67-91
DOI: 10.1051/ita:2005004

GATE CIRCUITS IN THE ALGEBRA OF TRANSIENTS

Janusz Brzozowski[1] and Mihaela Gheorghiu[2]

Abstract. We study simulation of gate circuits in the infinite algebra of transients recently introduced by Brzozowski and Ésik. A transient is a word consisting of alternating 0s and 1s; it represents a changing signal. In the algebra of transients, gates process transients instead of 0s and 1s. Simulation in this algebra is capable of counting signal changes and detecting hazards. We study two simulation algorithms: a general one that works with any initial state, and a special one that applies only if the initial state is stable. We show that the two algorithms agree in the stable case. We also show that the general algorithm is insensitive to the removal of state variables that are not feedback variables. We prove the sufficiency of simulation: all signal changes occurring in binary analysis are predicted by the general algorithm. Finally, we show that simulation can be more pessimistic than binary analysis, if wire delays are not taken into account. We propose a circuit model that we conjecture to be sufficient for proving the equivalence of simulation and binary analysis for feedback-free circuits.

Mathematics Subject Classification. 06A06, 94C10, 94C12.

Introduction

Asynchronous circuits, in contrast to synchronous ones, operate without a clock. Interest in asynchronous circuits has grown in recent years [1, 9, 12, 16], because they offer the potential for higher speed and lower energy consumption, avoid clock distribution problems, handle metastability safely, and are amenable to modular design.

Keywords and phrases. Algebra, digital circuit, hazard detection, signal changes, simulation, transient.

[1] School of Computer Science, University of Waterloo, Waterloo, ON, N2L 3G1, Canada; brzozo@uwaterloo.ca
[2] Department of Computer Science, University of Toronto, Toronto, ON, M5S 3G4, Canada; mg@cs.toronto.edu

© EDP Sciences 2005

Despite its advantages, asynchronous design has some problems, among them, hazards. A hazard is an unwanted signal pulse, caused by stray delays. If a signal is not supposed to change, but has such an unwanted pulse, the hazard is called *static*; if the signal is supposed to change, the unwanted pulse constitutes a *dynamic* hazard. Another type of behavior that is usually undesirable in a digital circuit is an *oscillation*, which is a sequence of states repeated infinitely often.

Because hazards may affect the correctness of computations they are important, and much research has been done on their detection. Early work [13, 17] used Boolean algebra and Karnaugh maps in complex procedures that depend very much on the structure of the circuits. Multi-valued algebras have been used as an alternative to the Boolean methods; see [3] for a survey. Among these algebras, one of the most successful is the three-valued algebra used by the *ternary simulation* algorithm introduced in [8]. This simulation algorithm provides a simple and very efficient (linear time) method of detecting static hazards and oscillations, but is not capable of detecting dynamic hazards. A complete characterization of the ternary simulation in terms of binary analysis is given in [6]. The characterization states that the simulation provides the least upper bound of the result of binary analysis, under the assumption that both gates and wires have arbitrary, but finite, delays. As a corollary, it is shown that static hazards and oscillations are correctly detected by simulation. The algorithm, originally defined for stable initial states, is generalized in [15] to handle any initial state. None of the other multi-valued algebras proposed for hazard analysis provides a simulation algorithm as well defined and as well understood as ternary simulation.

In a recent paper [2], Brzozowski and Ésik introduced a general infinite-valued algebra, the *change-counting* algebra C, that generalizes all the successful multi-valued algebras previously used for hazard analysis. They also proposed a simple simulation algorithm that generalizes ternary simulation. The algorithm is capable not only of detecting and identifying hazards, but also of counting the worst-case signal changes, in any (synchronous or asynchronous) gate circuit. This could provide an estimate of energy consumption.

The purpose of our work is to characterize the simulation algorithm of Brzozowski and Ésik. We study two simulation algorithms: a general one, A, that works with any initial state, and $\tilde{\text{A}}$, that applies if the initial state is stable. We show that the two algorithms agree in the stable case. We also show that A is insensitive to the removal of state variables that are not feedback variables. We prove the sufficiency of simulation: all signal changes occurring in binary analysis are predicted by Algorithm A. Finally, we show that simulation can be more pessimistic than binary analysis, if wire delays are not taken into account. We propose a circuit model that we conjecture to be sufficient for proving the equivalence of simulation and binary analysis for feedback-free circuits.

The article is structured as follows. In Section 1 we define the network model of gate circuits. We present Brzozowski and Ésik's algebra C of transients in Section 2. In Section 3, we describe the classical binary analysis and present a new interpretation of its results in terms of histories of the circuit variables. Section 4 introduces the simulation method based on algebra C. We define an

algorithm that is more general than that of [2], in the sense that it does not require the initial state to be stable. Our algorithm is called Algorithm A. We also present the original definition of the simulation [2], with stable initial state, and call it Algorithm $\tilde{\text{A}}$; we show that Algorithms A and $\tilde{\text{A}}$ are equivalent (in the stable-state case) under models containing input delays. In Section 5 we prove that A is insensitive to the removal of state variables that are not feedback variables. In Section 6 we establish the sufficiency of simulation by showing that all signal changes occurring in binary analysis appear also in the result of Algorithm A, for any gate circuit. This result implies that simulation detects all hazardous signal changes. In Section 7 we study conditions under which binary analysis covers simulation. Section 8 concludes the paper.

We use the following notational conventions. For an integer $n > 0$, the set $\{1, \ldots, n\}$ is denoted by $[n]$. Boolean operations AND, OR, NOT, and XOR are denoted $\wedge$, $\vee$, $\bar{}$, and $\veebar$, respectively. Whenever possible, we write the steps of our proofs in the form

$$P \quad \mathcal{R} \quad Q \qquad \{\, \mathcal{F} \,\},$$

where P and Q are statements or expressions, $\mathcal{R}$ is a relation such as $=$, $\leq$, $\geq$, $\Rightarrow$, or $\Leftrightarrow$, and $\mathcal{F}$ is a series of facts. These deduction steps should be read as: P is in relation $\mathcal{R}$ with Q, by the facts in $\mathcal{F}$.

1. Network model

This section is based on [5]. In a gate circuit with n inputs and m gates, we have a set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of n *input variables* and a set $\mathcal{S} = \{s_1, \ldots, s_m\}$ of m *state variables*, usually corresponding to gates. Input and state variables take values in a multi-valued domain $\mathcal{D}$. Each state variable s_i has an *excitation* S_i. In general, S_i is a function of some inputs $X_{j_1}, \ldots, X_{j_l} \in \mathcal{X}$, and some state variables $s_{i_1}, \ldots, s_{i_k} \in \mathcal{S}$, *i.e.*, $S_i = f(X_{j_1}, \ldots, X_{j_l}, s_{i_1}, \ldots, s_{i_k})$, where $f : \mathcal{D}^{l+k} \to \mathcal{D}$. In the case where $\mathcal{D} = \{0, 1\}$, the excitation is the Boolean function of the corresponding gate. For multi-valued domains, the excitation is an extension of the corresponding Boolean gate function. We also treat S_i as a function from $\mathcal{D}^{n+m}$ into $\mathcal{D}$. Thus, let $\tilde{S}_i : \mathcal{D}^{n+m} \to \mathcal{D}$ be $\tilde{S}_i(a \cdot b) = f(a_{j_1}, \ldots, a_{j_l}, b_{i_1}, \ldots, b_{i_k})$, for any $a \cdot b$. From now on we write S_i for $\tilde{S}_i$; the meaning is clear from the context.

Definition 1.1. A *network* is a tuple $N = \langle \mathcal{D}, \mathcal{X}, \mathcal{S}, \mathcal{E} \rangle$, where $\mathcal{D}$ is the domain of values, $\mathcal{X} = \{X_1, \ldots, X_n\}$, the set of input variables, $\mathcal{S} = \{s_1, \ldots, s_m\}$, the set of state variables with associated excitations $S_1, \ldots, S_m$, and $\mathcal{E} \subseteq (\mathcal{X} \times \mathcal{S}) \cup (\mathcal{S} \times \mathcal{S})$, a set of directed edges. There is an edge between x and y if and only if the excitation of y depends[1] on x. The *network graph* is the digraph $(\mathcal{X} \cup \mathcal{S}, \mathcal{E})$.

[1] A function $f : \mathcal{D}^k \to \mathcal{D}$ depends on its ith argument if there exist $d, d' \in \mathcal{D}$, $d = (d_1, \ldots, d_{i-1}, d_i, d_{i+1}, \ldots, d_k)$, $d' = (d_1, \ldots, d_{i-1}, d'_i, d_{i+1}, \ldots, d_k)$ such that $f(d) \neq f(d')$.

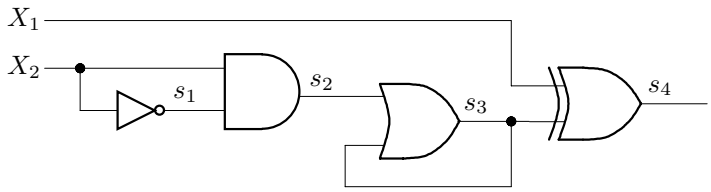

FIGURE 1. Sample gate circuit.

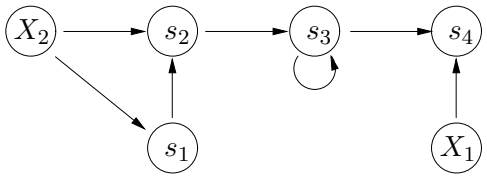

FIGURE 2. Network graph for circuit of Figure 1.

A circuit is feedback-free if, starting at any point in the circuit and proceeding *via* connections through gates in the direction input to output, it is not possible to reach the same point twice. Every feedback-free circuit has an acyclic network graph. The converse does not hold, since there do exist circuits with feedback that have acyclic graphs (see [5]).

Example 1.2. The circuit of Figure 1 has input variables X_1 and X_2, and state variables s_1, s_2, s_3, s_4. In the domain $\mathcal{D} = \{0, 1\}$, the excitations are $S_1 = \overline{X_2}$, $S_2 = X_2 \wedge s_1$, $S_3 = s_2 \vee s_3$, and $S_4 = X_1 \veebar s_3$. The network graph is shown in Figure 2.

For convenience, a circuit with associated state variables will be referred to as a circuit or a network. This is justified, because a circuit with state variables determines a unique network.

A *state* of N is an m-tuple b of values from $\mathcal{D}$ assigned to state variables $s_1, \ldots, s_m$. A *total state* is an $(n + m)$-tuple $c = a \cdot b$ of values from $\mathcal{D}$, the n-tuple a being the values of the input variables, and the m-tuple b, the values of state variables. The dot " $\cdot$ " separates input from state variables.

For any $i \in [m]$, the value of S_i in total state $a \cdot b$ is denoted $S_i(a \cdot b)$. The tuple $(S_1(a \cdot b), \ldots, S_m(a \cdot b))$ is denoted by $S(a \cdot b)$. For any $a \cdot b$, we define the set of unstable state variables as $U(a \cdot b) = \{s_i \mid b_i \neq S_i(a \cdot b)\}$. Thus, $a \cdot b$ is *stable* if and only if $U(a \cdot b) = \emptyset$, *i.e.*, $S(a \cdot b) = b$.

For any state variable $s_i \in \mathcal{S}$, its *fan-in set* is $\phi(s_i) = \{x \mid x \in \mathcal{X} \cup \mathcal{S}, (x, s_i) \in \mathcal{E}\}$.

2. TRANSIENTS

In this section we present the infinite domain of transients, introduced in [2]. A *binary word* is any word in $\{0, 1\}^*$. A *transient* is a nonempty binary word in which no two consecutive symbols are the same. Thus the set of all transients

FIGURE 3. Transients as words for waveforms.

is $\mathbf{T} = 0(10)^* \cup 1(01)^* \cup 0(10)^*1 \cup 1(01)^*0$. Transients represent waveforms in a natural way, as shown in Figure 3.

We use boldface symbols to denote transients, tuples of transients, and functions of transients. For any transient $\mathbf{t}$ we denote by $\alpha(\mathbf{t})$ and $\omega(\mathbf{t})$ its first and last characters, respectively. A transient can be obtained from any nonempty binary word by *contraction*, *i.e.*, elimination of all duplicates immediately following a symbol (*e.g.*, the contraction of 00100011 is 0101). For a binary word s we denote by $\hat{s}$ the result of its contraction. For any $\mathbf{t}, \mathbf{t}' \in \mathbf{T}$, we denote by $\mathbf{t}\mathbf{t}'$ the concatenation of $\mathbf{t}$ and $\mathbf{t}'$.

For $t, t' \in \{0,1\}^*$, t is a *prefix* of t' if there exists a (possibly empty) binary word t'' such that $t' = tt''$. The prefix relation is a partial order on $\{0,1\}^*$; it is denoted $\leq$. In this paper we restrict the partial order relation $\leq$ to $\mathbf{T}$. Also, for $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_m)$ and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_m)$ in $\mathbf{T}^m$, we say that $\mathbf{u}$ is a prefix of $\mathbf{v}$ and write $\mathbf{u} \leq \mathbf{v}$, if $\mathbf{u}_i \leq \mathbf{v}_i$, for all $i \in [m]$.

Extensions of Boolean functions to functions of transients are formally defined in [2]. Any Boolean function $f : B^n \to B$ is extended to a function $\mathbf{f} : \mathbf{T}^n \to \mathbf{T}$ so that, for any tuple $(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ of transients, $\mathbf{f}$ produces the longest transient when $\mathbf{t}_1, \ldots, \mathbf{t}_n$ are applied to the inputs of a gate performing the Boolean function f. We give an example below; for more details see [2].

Example 2.1. Let f be the two-input OR function and $\mathbf{f}$, its extension. Suppose we want to compute $\mathbf{f}(01, 010)$. We construct a digraph $D(01, 010)$ in which the nodes consist of all the pairs $(\mathbf{t}, \mathbf{t}')$ of transients such that $(\mathbf{t}, \mathbf{t}') \leq (01, 010)$, and there is an edge between any two pairs $\mathbf{p}, \mathbf{p}'$ only if $\mathbf{p} \leq \mathbf{p}'$, and $\mathbf{p}$ differs from $\mathbf{p}'$ in exactly one coordinate by exactly one letter. The resulting graph is shown in Figure 4a. Also, for each node $(\mathbf{t}, \mathbf{t}')$ in the graph we consider as its label the value $f(\omega(\mathbf{t}), \omega(\mathbf{t}'))$. This results in a graph of labels, shown in Figure 4b. The value of $\mathbf{f}(01, 010)$ is the contraction of the label sequence of those paths in the graph of labels that have the largest number of alternations between 0 and 1. Therefore, $\mathbf{f}(01, 010) = 0101$.

Let $z(\mathbf{t})$ and $u(\mathbf{t})$ denote the number of 0s and the number of 1s in a transient $\mathbf{t}$, respectively. We denote by $\otimes$ and $\oplus$ the extensions of the Boolean AND and OR operations, respectively. It is shown in [2] that for any $\mathbf{w}, \mathbf{w}' \in \mathbf{T}$ of length >1, $\mathbf{w} \otimes \mathbf{w}' = \mathbf{t}$, where $\mathbf{t} \in \mathbf{T}$ is such that

$$\alpha(\mathbf{t}) = \alpha(\mathbf{w}) \wedge \alpha(\mathbf{w}'), \;\; \omega(\mathbf{t}) = \omega(\mathbf{w}) \wedge \omega(\mathbf{w}'), \quad \text{and} \quad u(\mathbf{t}) = u(\mathbf{w}) + u(\mathbf{w}') - 1.$$

Similarly, $\mathbf{w} \oplus \mathbf{w}' = \mathbf{t}$, where $\mathbf{t} \in \mathbf{T}$ is such that

$$\alpha(\mathbf{t}) = \alpha(\mathbf{w}) \vee \alpha(\mathbf{w}'), \;\; \omega(\mathbf{t}) = \omega(\mathbf{w}) \vee \omega(\mathbf{w}'), \quad \text{and} \quad z(\mathbf{t}) = z(\mathbf{w}) + z(\mathbf{w}') - 1.$$

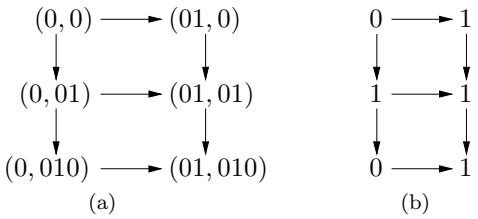

FIGURE 4. Graph $D(01, 010)$ with labels.

If one of the arguments is 0 or 1, the following rules apply:

$$\mathbf{t} \oplus 0 = 0 \oplus \mathbf{t} = \mathbf{t}, \quad \mathbf{t} \oplus 1 = 1 \oplus \mathbf{t} = 1,$$
$$\mathbf{t} \otimes 1 = 1 \otimes \mathbf{t} = \mathbf{t}, \quad \mathbf{t} \otimes 0 = 0 \otimes \mathbf{t} = 0.$$

The complement $\overline{\mathbf{t}}$ of $\mathbf{t} \in \mathbf{T}$ is obtained by complementing each character of $\mathbf{t}$. For example, $\overline{1010} = 0101$.

Algebra $C = (\mathbf{T}, \oplus, \otimes, \overline{\ }, 0, 1)$, is called the *change-counting algebra*, and is a commutative de Morgan bisemigroup [2]. We also refer to C as the *algebra of transients*.

We denote by $\mathbf{t} \circ \mathbf{t}'$ concatenation followed by contraction, *i.e.*, $\mathbf{t} \circ \mathbf{t}' = \widehat{\mathbf{t}\mathbf{t}'}$. The $\circ$ operation is associative, and also satisfies for $\mathbf{t}, \mathbf{t}', \mathbf{t}_1, \ldots, \mathbf{t}_n \in \mathbf{T}$ and $b \in \{0, 1\}$:

1. if $\mathbf{t} \leq \mathbf{t}'$ then $b \circ \mathbf{t} \leq b \circ \mathbf{t}'$; and
2. $\mathbf{t}_1 \circ \ldots \circ \mathbf{t}_n = \widehat{\mathbf{t}_1 \ldots \mathbf{t}_n}$.

We extend $\circ$ to tuples of transients the obvious way: for any m-tuples $\mathbf{u}, \mathbf{v}$ of transients, $\mathbf{u} \circ \mathbf{v} = \mathbf{w}$, where $\mathbf{w}$ is such that $\mathbf{w}_i = \mathbf{u}_i \circ \mathbf{v}_i$, for all $i \in [m]$.

3. Binary analysis of networks

In response to a change of its inputs, a network passes through a sequence of states as its internal signals change. By analyzing a network we mean exploring all such sequences of states. This section describes a formal analysis model introduced by Muller and Bartky [14], and later called the *General Multiple Winner* (GMW) model [7]. Our presentation follows that of [5], but here we refer to the GMW model as *binary analysis*. We derive new properties of binary analysis by examining the histories of the network variables along all possible paths.

3.1. Definition of binary analysis

In binary analysis we use the binary domain, $\mathcal{D} = \{0, 1\}$. We describe the behavior of a network started in a given state with the input kept constant at

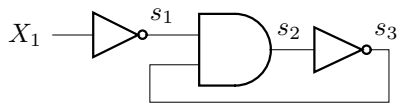

FIGURE 5. Gate circuit for binary analysis.

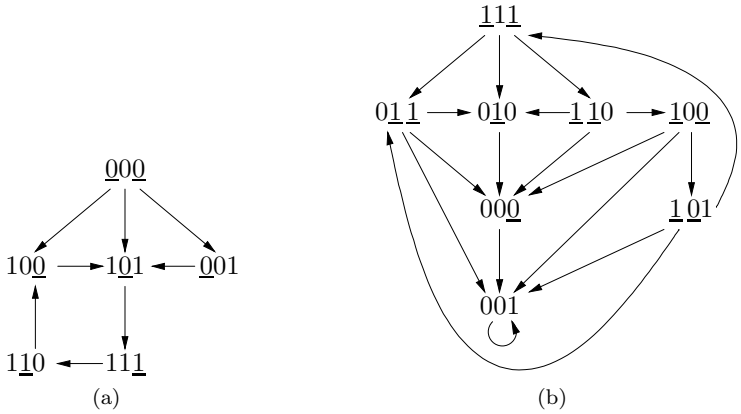

FIGURE 6. Sample $G_a(b)$ graphs for circuit of Figure 5.

value $a \in \{0,1\}^n$, by defining a binary relation R_a on the set $\{0,1\}^m$ of states of N. For any $b \in \{0,1\}^m$,

bR_ab, if $U(a \cdot b) = \emptyset$, *i.e.*, if total state $a \cdot b$ is stable, and

bR_ab^K, if $U(a \cdot b) \neq \emptyset$, and K is any nonempty subset of $U(a \cdot b)$,

where by b^K we mean b with all the variables in K complemented. No other pairs of states are related by R_a. As usual, we associate a digraph with the R_a relation, and denote it G_a.

For given $a \in \{0,1\}^n$, and $b \in \{0,1\}^m$ we define the set of all states reachable from b in relation R_a as

$$reach(R_a(b)) = \{c \mid bR_a^*c\},$$

where R_a^* is the reflexive and transitive closure of R_a. We denote by $G_a(b)$ the subgraph of G_a corresponding to $reach(R_a(b))$.

In examples, we represent tuples without commas or parentheses, for convenience. Thus $(0,0,0)$ is written as 000, etc.

Example 3.1. For the circuit in Figure 5, graph $G_0(000)$ is shown in Figure 6a, where unstable variables are underlined. Note that the graph contains no stable states. Graph $G_1(111)$ is shown in Figure 6b. Here there is one stable state. To illustrate hazardous behavior, consider path $\pi_1 = 111, 011, 001$. Here s_2 changes once from 1 to 0, and s_3 does not change. However, along path $\pi_2 = 111, 110, 100, 101, 011, 001$, s_2 changes from 1 to 0 to 1 to 0, and s_3 changes from 1 to 0 to 1. If the behavior of π_1 is the intended one, then π_2 violates it. Along π_2 there are unwanted signal pulses: a 1-pulse in s_2, and a 0-pulse in s_3. The first

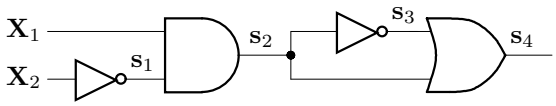

FIGURE 7. Circuit illustrating inertial delays.

pulse is an example of a dynamic hazard, and the second, of a static hazard. Such pulses can introduce errors in the circuit operation.

3.2. HISTORIES

In the following we assume an arbitrary network N and take $G_a(b)$ to be any one of its binary analysis graphs.

Definition 3.2. Let $\pi = s^0, \ldots, s^h$ be a path of length $h \geq 0$ in $G_a(b)$. Recall that each s^j is a tuple $(s_1^j, \ldots, s_m^j)$. For any $i \in [m]$, we denote by $\boldsymbol{\sigma}_i^\pi$ the transient $\widehat{s_i^0 \ldots s_i^h}$, which shows the changes of the i-th state variable along path π. We refer to it as the *history* of variable s_i along the path. We define $\boldsymbol{\Sigma}_i^\pi$ to be $\widehat{E_i}$, where $E_i = S_i(a \cdot s^0)S_i(a \cdot s^1) \ldots S_i(a \cdot s^h)$, and call it the *excitation history* of variable s_i along path π. The histories of all variables along π constitute tuple $\boldsymbol{\sigma}^\pi = (\boldsymbol{\sigma}_1^\pi, \ldots, \boldsymbol{\sigma}_m^\pi)$. The histories of all excitations along π form tuple $\boldsymbol{\Sigma}^\pi = (\boldsymbol{\Sigma}_1^\pi, \ldots, \boldsymbol{\Sigma}_m^\pi)$.

Consider a physical circuit, and assume that one of its gates is stable in some state; then its output has the same value as its excitation. If the excitation changes to a new value, the gate output also changes after some delay. When the excitation changes back to its original value, the gate output follows the new change. However, if the excitation pulse is very short, the gate output may not change at all, because of the inertial nature of the gate delay.

In our mathematical representation of a circuit, we are able to model the inertial delay of a gate by differentiating between the value of a variable and the value of its excitation. The stability of a gate depends only on the present state of the circuit. Thus, if a variable s_i is unstable in some state, and another variable changes, causing a change of the excitation of s_i, then s_i is stable again. Because of this, the histories $\boldsymbol{\sigma}_i^\pi$ and $\boldsymbol{\Sigma}_i^\pi$ are not always the same. If s_i is unstable initially, they are obviously different, since their first characters are different, that is $s_i^0 \neq S_i(a \cdot s_i^0)$. However, $\boldsymbol{\sigma}_i^\pi$ and $\boldsymbol{\Sigma}_i^\pi$ can differ, even if s_i is stable initially.

Example 3.3. In the circuit of Figure 7, the excitations are: $S_1 = \overline{X_2}$, $S_2 = X_1 \otimes s_1$, $S_3 = \overline{s_2}$, $S_4 = s_2 \oplus s_3$. One verifies that graph $G_{11}(1011)$ contains path $\pi = \underline{1}\,\underline{0}11, 0\underline{1}\,\underline{1}1, 0011$, on which variable s_3 changes fewer times than its excitation. We have $\boldsymbol{\sigma}_3^\pi = 1$, whereas $\boldsymbol{\Sigma}_3^\pi = 101$.

In the following we establish a formal relationship between histories of variables and histories of their excitations.

Proposition 3.4. *Let $\pi = s^0, \ldots, s^h, s^{h+1}$ be a path in $G_a(b)$, and let $\pi' = s^0, \ldots, s^h$. Then, $\boldsymbol{\sigma}^\pi \leq \boldsymbol{\sigma}^{\pi'} \circ S(a \cdot s^h)$.*

Proof. For any variable s_i we have one of the following cases.

Case I. s_i changes during the transition from s^h to s^{h+1}. Then s_i must be unstable in state s^h, *i.e.*, $S_i(a \cdot s^h) \neq s_i^h$, and $s_i^{h+1} = S_i(a \cdot s^h)$, by the definition of binary analysis. Hence $\boldsymbol{\sigma}_i^{\pi} = s_i^0 \ldots \widehat{s_i^h s_i^{h+1}} = \widehat{s_i^0 \ldots s_i^h} \circ s_i^{h+1} = \boldsymbol{\sigma}_i^{\pi'} \circ S_i(a \cdot s^h)$.

Case II. s_i does not change during the transition from s^h to s^{h+1}. Then $s_i^{h+1} = s_i^h$, by the definition of binary analysis. Then $\boldsymbol{\sigma}_i^{\pi} = s_i^0 \ldots \widehat{s_i^h s_i^{h+1}} = \widehat{s_i^0 \ldots s_i^h} = \boldsymbol{\sigma}_i^{\pi'} \leq \boldsymbol{\sigma}_i^{\pi'} \circ S_i(a \cdot s^h)$. Thus, our claim holds. □

Corollary 3.5. *For any path $\pi = s^0, \ldots, s^h, s^{h+1}$ in $G_a(b)$, with $\pi' = s^0, \ldots, s^h$ we have $\boldsymbol{\sigma}^{\pi} \leq s^0 \circ \boldsymbol{\Sigma}^{\pi'}$.*

Proof. $\boldsymbol{\sigma}^{\pi} \leq \boldsymbol{\sigma}^{\pi'} \circ S(a \cdot s^h) \leq (\ldots((s^0 \circ S(a \cdot s^0)) \circ S(a \cdot s^1)) \circ \ldots) \circ S(a \cdot s^h) = s^0 \circ (S(a \cdot s^0) \circ S(a \cdot s^1) \circ \ldots \circ S(a \cdot s^h)) = s^0 \circ \boldsymbol{\Sigma}^{\pi'}$. □

Corollary 3.6. *Let $\pi = s^0, \ldots, s^h$ be a path in $G_a(b)$. For all variables s_i that are stable in s^0, we have*

$$\boldsymbol{\sigma}_i^{\pi} \leq \boldsymbol{\Sigma}_i^{\pi}. \tag{1}$$

Proof. If $h = 0$ then $\boldsymbol{\sigma}_i^{\pi} = s_i^0$ and $\boldsymbol{\Sigma}_i^{\pi} = S_i(a \cdot s^0)$ for all variables s_i. For s_i initially stable we have $s_i^0 = S_i(a \cdot s^0)$ by the definition of stability, and the claim holds.

For $h > 0$ the following argument applies for any s_i initially stable:

$$\begin{array}{lll} \boldsymbol{\sigma}_i^{\pi} & \leq & s_i^0 \circ \boldsymbol{\Sigma}_i^{\pi'} \qquad \{ \text{ Cor. 3.5 with } \pi' = s^0, \ldots, s^{h-1} \ \} \\ & = & \boldsymbol{\Sigma}_i^{\pi'} \qquad \{ \text{ stability of } s_i \text{ in } s^0 \text{ and } \alpha(\boldsymbol{\Sigma}_i^{\pi'}) = S_i(a \cdot s^0) \ \} \\ & \leq & \boldsymbol{\Sigma}_i^{\pi} \qquad \{ \text{ definition of } \boldsymbol{\Sigma}_i^{\pi} \text{ as } \boldsymbol{\Sigma}_i^{\pi} = \boldsymbol{\Sigma}_i^{\pi'} \circ S_i(a \cdot s^h) \ \}. \end{array}$$ □

The paths for which equality holds in (1) are called *hazard-preserving paths.*

Definition 3.7. Let π be a path in $G_a(b)$, and s_i a state variable that is initially stable. We call π *hazard-preserving on* s_i if $\boldsymbol{\sigma}_i^{\pi} = \boldsymbol{\Sigma}_i^{\pi}$. For $V \subseteq \mathcal{S}$, path π is *hazard-preserving on* V if it is hazard-preserving on all $s_i \in V$.

We also establish a relationship between excitation histories and extended excitations.

Proposition 3.8. *For any path $\pi = s^0, \ldots, s^h$ in $G_a(b)$, $\boldsymbol{\Sigma}^{\pi} \leq \mathbf{S}(a \cdot \boldsymbol{\sigma}^{\pi})$.*

Proof. Let $\pi_j = s^0, \ldots, s^j$, for all j such that $0 \leq j \leq h$. Then $\boldsymbol{\sigma}^{\pi_0} \leq \boldsymbol{\sigma}^{\pi_1} \leq \ldots \leq \boldsymbol{\sigma}^{\pi_h} = \boldsymbol{\sigma}^{\pi}$. Thus $a \cdot \boldsymbol{\sigma}^{\pi_0} \leq a \cdot \boldsymbol{\sigma}^{\pi_1} \leq \ldots \leq a \cdot \boldsymbol{\sigma}^{\pi}$, which means that $a \cdot \boldsymbol{\sigma}^{\pi_0}, a \cdot \boldsymbol{\sigma}^{\pi_1}, \ldots, a \cdot \boldsymbol{\sigma}^{\pi}$ is a subsequence q of nodes on a path p from $a \cdot (\alpha(\boldsymbol{\sigma}_1^{\pi}) \ldots \alpha(\boldsymbol{\sigma}_m^{\pi})) = a \cdot s^0 = a \cdot \boldsymbol{\sigma}^{\pi_0}$ to $a \cdot \boldsymbol{\sigma}^{\pi}$ in the graph $D(a \cdot \boldsymbol{\sigma}^{\pi})$. For any $i \in [m]$, we consider the labeling of graph $D(a \cdot \boldsymbol{\sigma}^{\pi})$ with Boolean excitation S_i. Let λ be the sequence of labels of p. The sequence of labels on q is $E_i = S_i(a \cdot s^0), S_i(a \cdot s^1), \ldots, S_i(a \cdot s^h)$. Since q is a subsequence of p, we have $\widehat{E_i} \leq \hat{\lambda}$. By the definition of extended Boolean functions, $\mathbf{S}_i(a \cdot \boldsymbol{\sigma}^{\pi})$ is the longest transient obtained by the contraction

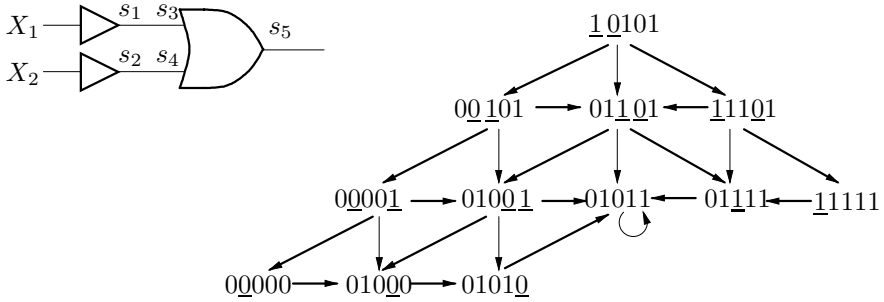

FIGURE 8. Circuit with binary analysis.

of the label sequences of paths from $a \cdot \boldsymbol{\sigma}^{\pi_0}$ to $a \cdot \boldsymbol{\sigma}^{\pi}$ in graph $D(a \cdot \boldsymbol{\sigma}^{\pi})$. Hence $\hat{\lambda} \leq \mathbf{S}_i(a \cdot \boldsymbol{\sigma}^{\pi})$. By the definition of excitation history, $\boldsymbol{\Sigma}_i^{\pi} = \widehat{E_i}$. It follows that $\boldsymbol{\Sigma}_i^{\pi} \leq \mathbf{S}_i(a \cdot \boldsymbol{\sigma}^{\pi})$. □

Corollary 3.9. *Let path π be as in Proposition 3.8, and let s_i be a state variable whose excitation is independent of the input variables. Let $\{s_{i_1}, \ldots, s_{i_k}\}$ be the fan-in set of s_i. Then*

$$\boldsymbol{\Sigma}_i^{\pi} \leq \mathbf{S}_i(\boldsymbol{\sigma}_{i_1}^{\pi}, \ldots, \boldsymbol{\sigma}_{i_k}^{\pi}). \tag{2}$$

Proof. Since S_i depends only on state variables $\{s_{i_1}, \ldots, s_{i_k}\}$, from the definition of excitations in $\mathbf{N}$ we have $\mathbf{S}_i(a \cdot \boldsymbol{\sigma}^{\pi}) = \mathbf{S}_i(\boldsymbol{\sigma}_{i_1}^{\pi}, \ldots, \boldsymbol{\sigma}_{i_k}^{\pi})$. The claim now follows immediately by Proposition 3.8. □

The paths for which equality holds in (2) are called *worst-case paths.*

Definition 3.10. Let π be a path in $G_a(b)$, s_i, a state variable whose excitation is independent of the input variables, and $\phi(s_i) = \{s_1, \ldots, s_k\}$. We call π *worst-case on* s_i if $\boldsymbol{\Sigma}_i^{\pi} = \mathbf{S}_i(\boldsymbol{\sigma}_1^{\pi}, \ldots, \boldsymbol{\sigma}_k^{\pi})$. For a set $V \subseteq \mathcal{S}$ of state variables, path π is *worst-case on* V if it is worst-case on each $s_i \in V$.

Example 3.11. In the circuit in Figure 8, a state variable is associated with each gate and also with each wire between gates. Wire delays can be viewed as gates performing the identity function. We indicate wire delays by adding state variables associated with wires. The excitations are $S_1 = X_1$, $S_2 = X_2$, $S_3 = s_1$, $S_4 = s_2$, $S_5 = s_3 \vee s_4$. We show $G_{01}(10101)$ in the figure. Path $\pi = 10101, 00101, 00001, 01001, 01011$ is worst-case on s_5, since $\Sigma_5^{\pi} = 101 = 10 \oplus 01$. However, $\pi' = 10101, 11101, 11111, 01111, 01011$ is not worst-case, since $\boldsymbol{\Sigma}_5^{\pi'} = 1$, whereas $10 \oplus 01 = 101$.

3.3. Binary analysis of feedback-free circuits

For feedback-free circuits, we now show that simultaneous changes do not affect the histories of variables. For a graph $G_a(b)$ we denote by $G'_a(b)$ the subgraph of $G_a(b)$ in which exactly one unstable variable changes at each step.

For the circuit in Figure 8, subgraph $G'_{01}(10101)$ is shown by boldface edges.

Definition 3.12. In any binary analysis graph $G_a(b)$, two paths π and π' are called *equivalent* if their histories are the same, *i.e.*, if $\boldsymbol{\sigma}^{\pi} = \boldsymbol{\sigma}^{\pi'}$.

Proposition 3.13. *Let $G_a(b)$ be a binary analysis graph of a feedback-free circuit. Then for any path π in $G_a(b)$ there exists an equivalent path π' in $G'_a(b)$.*

Proof. Take any path $\pi = s^0, \ldots, s^{p-1}, s^p, \ldots, s^h$, with $0 < p \leq h$. Let $s_1, \ldots, s_k$ be the variables that change in step p. Since the circuit has no feedback, the graph of N is acyclic, and there exists a topological ordering of its state variables. Let $s_k, \ldots, s_1$ be the topological order of the k variables, *i.e.*, if $s_i \in \phi(s_j)$, then $i > j$, for $i, j \in [k]$. It follows that for any $i \in [k]$, $s_i \notin \phi(s_j)$, for all j such that $i \leq j \leq k$. This means that changing s_i in a state in which $s_k, \ldots, s_i$ are all unstable, leads to a state in which $s_k, \ldots, s_{i+1}$ are still unstable, where $i \in [k-1]$. This allows us to replace step p of π with a sequence $r^0, \ldots, r^k$ of k steps, where $r^0 = s^{p-1}, r^k = s^p$, and for any $i \in [k]$ only s_i changes in step i. The new path $\pi' = s^0, \ldots, s^{p-1}, r^1, \ldots, r^{k-1}, s^p, \ldots, s^h$ has the same history as π, *i.e.*, $\boldsymbol{\sigma}^{\pi} = \boldsymbol{\sigma}^{\pi'}$. In a similar way we eliminate all steps with simultaneous changes, and in the end obtain an equivalent path with no simultaneous changes. □

Example 3.14. In Figure 8, path $10101, 01101, 01011$ involves two double changes of variables. An equivalent path with single changes is $10101, 00101, 01101, 01001, 01011$.

The following example shows that this result does not hold for circuits with feedback.

Example 3.15. Consider a circuit with one input X, and two gates with excitations $S_1 = \overline{X \vee s_2}$ and $S_2 = \overline{X \vee s_1}$. One verifies that in the graph $G'_0(00)$, the only possible paths are $00, 01$ and $00, 10$. However, in $G_0(00)$ there is an infinite path $00, 11, 00, 11, 00, \ldots$

4. Simulation with algebra C

While binary analysis is an exhaustive analysis of a circuit, it is inefficient, since the state space is exponential. Simulation using a multi-valued domain is an efficient alternative, if not all the information from binary analysis is needed. A simulation algorithm using algebra C has been proposed in [2]; it generalizes ternary simulation [5, 8]. We now give a more general version of the simulation algorithm, and show how it relates to the original version. This parallels the extension of ternary simulation from stable initial state to any initial state [5].

Given any circuit, we define two networks: a binary network $N = \langle \{0,1\}, \mathcal{X}, \mathcal{S}, \mathcal{E} \rangle$ and the *transient network* $\mathbf{N} = \langle \mathbf{T}, \mathcal{X}, \mathcal{S}, \mathcal{E} \rangle$, having set $\mathbf{T}$ of transients as the domain. The two networks have the same input and state variables, but these variables take values from different domains. A state of network $\mathbf{N}$ is a tuple of transients; the value of the excitation of a variable is also a transient. Excitations in $\mathbf{N}$ are the extensions to C of the Boolean excitations in N. It is shown in [2] that an extended Boolean function depends on one of its arguments if and only

if the corresponding Boolean function depends on that argument. Therefore N and $\mathbf{N}$ have the same set of edges.

Recall that binary variables, words, tuples and excitations in N are denoted by italic characters (*e.g.*, s, S). Transients, tuples of transients, and excitations in $\mathbf{N}$ are denoted by boldface characters (*e.g.*, $\mathbf{s}$, $\mathbf{S}$), and components of a tuple by subscripts (*e.g.*, $\mathbf{s}_i$, s_i).

4.1. General simulation: Algorithm A

We use N for binary analysis and $\mathbf{N}$ for simulation. We record in the value of a variable of $\mathbf{N}$ all the changes in that variable since the start of the simulation, as dictated by its excitation. For variables that are stable initially, since the initial state agrees with the initial excitation, the state transient and the excitation transient will be the same. Consequently, at each step we just copy the excitation into the variable. For example, with initial state 0 and excitation 0, if the excitation becomes 01, we set the variable to 01, and so on. For variables that are initially unstable, we first record the initial state, and then the excitation. The operator that gives us the desired result in both cases is $\circ$; thus we have *new_value* = *initial_value* $\circ$ *excitation*.

Let $a \cdot b$ be a (binary) total state of $\mathbf{N}$. Algorithm A is defined as follows:

Algorithm A
$\mathbf{s}^0 := b$;
$h := 1$;
$\mathbf{s}^h := b \circ \mathbf{S}(a \cdot \mathbf{s}^0)$;
while $(\mathbf{s}^h <> \mathbf{s}^{h-1})$ **do**
    $h := h + 1$;
    $\mathbf{s}^h := b \circ \mathbf{S}(a \cdot \mathbf{s}^{h-1})$;

Algorithm A produces a sequence $\mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^h, \ldots$, where $\mathbf{s}^h = (\mathbf{s}_1^h, \mathbf{s}_2^h, \ldots, \mathbf{s}_m^h) \in \mathbf{T}^m$, for all $h \geq 0$. This sequence can be finite, if we reach $\mathbf{s}^{h_0} = \mathbf{s}^{h_0-1}$ for some $h_0 > 0$, or infinite otherwise. For convenience, we sometimes consider the finite sequences as being infinite, with $\mathbf{s}^h = \mathbf{s}^{h_0}$, for all $h > h_0$.

It is shown in [2] that any extended Boolean function $\mathbf{f} : \mathbf{T}^m \to \mathbf{T}$ is *monotonic* with respect to the prefix order, *i.e.*, for any $\mathbf{x}, \mathbf{y} \in \mathbf{T}^m$, if $\mathbf{x} \leq \mathbf{y}$, then $\mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{y})$.

Proposition 4.1. *The sequence resulting from Algorithm A is nondecreasing or monotonic with respect to the prefix order, that is, for all* $h \geq 0$, $\mathbf{s}^h \leq \mathbf{s}^{h+1}$.

Proof. Since extended Boolean functions are monotonic with respect to the prefix order, so are excitations. We proceed by induction on h.
Basis, $h = 0$: $\mathbf{s}^0 = b \leq b \circ \mathbf{S}(a \cdot \mathbf{s}^0) = \mathbf{s}^1$.
Induction step: $\mathbf{s}^h = b \circ \mathbf{S}(a \cdot \mathbf{s}^{h-1}) \leq b \circ \mathbf{S}(a \cdot \mathbf{s}^h) = \mathbf{s}^{h+1}$. □

For feedback-free circuits, the sequence resulting from Algorithm A is finite. We can see this if we order the state variables by levels as follows. Level 1 consists of all state variables which depend only on external inputs. Level l consists of all state variables which depend only on variables of level $< l$, and on at least one

TABLE 1. Result of algorithm A.

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$	state
1	1	1	0	1	1	$\mathbf{s}^0$
1	1	10	01	1	1	$\mathbf{s}^1$
1	1	10	010	10	1	$\mathbf{s}^2$
1	1	10	010	101	1010	$\mathbf{s}^3$
1	1	10	010	101	10101	$\mathbf{s}^4$

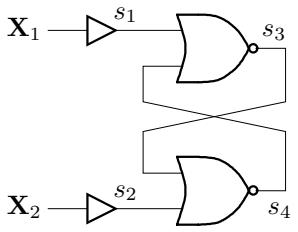

FIGURE 9. Circuit with infinite simulation.

variable of level $l-1$. Since the inputs do not change during simulation, level-1 variables change at most once, in the first step of Algorithm A. In general, level-i variables change at most i times. Since the number of levels is finite, our claim follows.

For display reasons, in examples of simulation we write binary states as words, but during computations they are regarded as tuples.

Example 4.2. Consider the feedback-free circuit in Figure 7. The extended excitations are: $\mathbf{S}_1 = \overline{\mathbf{X}_2}$, $\mathbf{S}_2 = \mathbf{X}_1 \otimes \mathbf{s}_1$, $\mathbf{S}_3 = \overline{\mathbf{s}_2}$, $\mathbf{S}_4 = \mathbf{s}_2 \oplus \mathbf{s}_3$. For the initial state $a \cdot b = 11 \cdot 1011$, Algorithm A results in Table 1.

Example 4.3. For circuits with feedback the simulation sequence may be infinite. Consider the circuit of Figure 9. The excitations are: $\mathbf{S}_1 = \mathbf{X}_1$, $\mathbf{S}_2 = \mathbf{X}_2$, $\mathbf{S}_3 = \overline{s_1 \oplus s_4}$, $\mathbf{S}_4 = \overline{s_2 \oplus s_3}$. We run Algorithm A for this network started in state $a \cdot b = 00 \cdot 1100$; the resulting sequence of states, which is infinite, is illustrated in Table 2.

4.2. SIMULATION WITH STABLE INITIAL STATE: ALGORITHM $\tilde{\text{A}}$

Algorithm A above makes no assumptions about the starting state $a \cdot b$. If the network starts in a stable total state and the inputs change, then we have a slightly simpler formulation which we call Algorithm $\tilde{\text{A}}$; this is the version used in [2]. Assume that $\mathbf{N}$ is started in stable total state $\tilde{a} \cdot b$ and the input tuple changes

TABLE 2. Infinite simulation.

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$	state
0	0	1	1	0	0	$\mathbf{s}^0$
0	0	10	10	0	0	$\mathbf{s}^1$
0	0	10	10	01	01	$\mathbf{s}^2$
0	0	10	10	010	010	$\mathbf{s}^3$
0	0	10	10	0101	0101	$\mathbf{s}^4$
...	...	...	...	...	...	...

TABLE 3. Result of Algorithm $\tilde{\text{A}}$.

$\mathbf{X}_1$	$\mathbf{X}_2$	$\tilde{\mathbf{s}}_1$	$\tilde{\mathbf{s}}_2$	$\tilde{\mathbf{s}}_3$	$\tilde{\mathbf{s}}_4$	state
1	10	0	0	1	1	$\tilde{\mathbf{s}}^0$
1	10	01	0	1	1	$\tilde{\mathbf{s}}^1$
1	10	01	01	1	1	$\tilde{\mathbf{s}}^2$
1	10	01	01	10	1	$\tilde{\mathbf{s}}^3$
1	10	01	01	10	101	$\tilde{\mathbf{s}}^4$

to a. Algorithm $\tilde{\text{A}}$ is defined below. It results in a sequence $\tilde{\mathbf{s}}^0, \tilde{\mathbf{s}}^1, \ldots, \tilde{\mathbf{s}}^h, \ldots$, of states, where $\tilde{\mathbf{s}}^h = (\tilde{\mathbf{s}}^h_1, \tilde{\mathbf{s}}^h_2, \ldots, \tilde{\mathbf{s}}^h_m) \in \mathbf{T}^m$, for all $h \geq 0$.

Algorithm $\tilde{\mathbf{A}}$
$\mathbf{a} = \tilde{a} \circ a$;
$\tilde{\mathbf{s}}^0 := b$;
$h := 1$;
$\tilde{\mathbf{s}}^h := \mathbf{S}(\mathbf{a} \cdot \tilde{\mathbf{s}}^0)$;
while $(\tilde{\mathbf{s}}^h <> \tilde{\mathbf{s}}^{h-1})$ **do**
    $h := h + 1$;
    $\tilde{\mathbf{s}}^h := \mathbf{S}(\mathbf{a} \cdot \tilde{\mathbf{s}}^{h-1})$;

Example 4.4. We illustrate Algorithm $\tilde{\text{A}}$ with the network in Figure 7, started in stable state $\tilde{a} \cdot b = 11 \cdot 0011$, with the input changing to $a = 10$. The result is shown in Table 3.

It is shown in [2] that the sequence of states resulting from Algorithm $\tilde{\text{A}}$ is nondecreasing with respect to the prefix order, *i.e.*, Algorithm $\tilde{\text{A}}$ is monotonic.

For feedback-free circuits, the time complexity of $\tilde{\text{A}}$ has been shown in [2] to be polynomial in the number of state variables and inputs. The additional work that A performs when compared to $\tilde{\text{A}}$ does not affect this complexity. Thus, for feedback-free circuits, the running time of A is polynomial as well.

For our next result, we modify the circuit model slightly. For each input X_i we add a delay, called *input delay*, with output s_i and excitation $S_i = X_i$. This follows the model of [5]. The following shows that Algorithms A and $\tilde{\text{A}}$ are equivalent for any network $\mathbf{N}$ started in a stable state, provided that $\mathbf{N}$ contains input delays.

Theorem 4.5. *Let $\mathbf{N}$ be a network containing input delays. Let $\tilde{\mathbf{s}}^0, \tilde{\mathbf{s}}^1, \ldots, \tilde{\mathbf{s}}^h, \ldots$ be the sequence of states produced by Algorithm $\tilde{A}$ for $\mathbf{N}$ started in the stable (binary) total state $\tilde{a} \cdot b$ with the input tuple changing to a. Then, for all $h \geq 0$, $\tilde{\mathbf{s}}^h = \mathbf{s}^h$, where $\mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^h, \ldots$ is the sequence of states produced by Algorithm A for $\mathbf{N}$ started in total state $a \cdot b$.*

Proof. We prove the theorem by induction on h.
Basis, $h = 0$. Since $\mathbf{s}^0 = b = \tilde{\mathbf{s}}^0$, the basis holds.
First step, $h = 1$. In states $\tilde{\mathbf{s}}^0$ and $\mathbf{s}^0$ only input delays can be unstable; therefore only they can change in the first step of $\tilde{\text{A}}$, and of A. One easily verifies that $\tilde{\mathbf{s}}^1 = \mathbf{s}^1$.
Induction step. For any $i \in [m]$, if $\mathbf{s}_i$ is an input delay then $\mathbf{s}_i^h = \mathbf{s}_i^{h-1}$ and $\tilde{\mathbf{s}}_i^h = \tilde{\mathbf{s}}_i^{h-1}$, because in both algorithms the state of input delays does not change after the first step. By the induction hypothesis, we have $\mathbf{s}_i^h = \tilde{\mathbf{s}}_i^h$. If $\mathbf{s}_i$ is not an input delay, then it is initially stable in both algorithms, and its excitation does not depend on the input tuple, *i.e.*, $\mathbf{S}_i(a \cdot \mathbf{x}) = \mathbf{S}_i(\mathbf{a} \cdot \mathbf{x})$, for any (internal) state tuple $\mathbf{x}$. Then $\mathbf{s}_i^h = \mathbf{S}_i(a \cdot \mathbf{s}^{h-1}) = \mathbf{S}_i(\mathbf{a} \cdot \tilde{\mathbf{s}}^{h-1}) = \tilde{\mathbf{s}}_i^h$. Hence $\tilde{\mathbf{s}}_i^h = \mathbf{s}_i^h$, for all $i \in [m]$. □

To illustrate this theorem, we run Algorithm $\tilde{\text{A}}$ on the circuit of Figure 9 with stable initial state $11 \cdot 1100$ and inputs changing to 00. The resulting states are identical to those in Table 2.

Note that the result does not necessarily hold for networks without input delays. Consider a two-input AND gate with inputs $\mathbf{X}_1$, $\mathbf{X}_2$ and state variable $\mathbf{s}_1$ with excitation $\mathbf{S}_1 = \mathbf{X}_1 \otimes \mathbf{X}_2$. Algorithm $\tilde{\text{A}}$ with initial state $01 \cdot 0$ and inputs changing to 10 produces states $\tilde{\mathbf{s}_1}^0 = 0$, $\tilde{\mathbf{s}_1}^1 = 010$, while Algorithm A with initial state $10 \cdot 0$ produces only state $\mathbf{s}^0 = 0$.

5. Reduced networks

We now study how simulation is affected by the removal of state variables from networks. This is important in determining the minimum number of variables sufficient for simulating a circuit. Our results parallel the ones in [5].

Let $\mathbf{N} = \langle \mathbf{T}, \mathcal{X}, \mathcal{S}, \mathcal{E} \rangle$ be a transient network with m state variables and n inputs. We assume that $\mathbf{N}$ contains at least one state variable that is not an input delay and does not depend on itself. Without loss of generality, we assume that the last state variable, $\mathbf{s}_m$, is such a variable; we can always renumber the variables in such a way that the assumption holds. Hence, for any total state $\mathbf{a} \cdot \mathbf{b} \in \mathbf{T}^{n+m}$ of $\mathbf{N}$, we have $\mathbf{S}_m(\mathbf{a} \cdot \mathbf{b}) = \mathbf{S}_m(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1}, \mathbf{t})$, where $\mathbf{t}$ is any transient.

Let $\dot{\mathbf{N}} = \langle \mathbf{T}, \mathcal{X}, \dot{\mathcal{S}}, \dot{\mathcal{E}} \rangle$ be the network obtained from $\mathbf{N}$ by removing state variable $\mathbf{s}_m$ as described below; we call $\dot{\mathbf{N}}$ a *reduced* network of $\mathbf{N}$. The set $\dot{\mathcal{S}}$ of state variables is $\mathcal{S} \setminus \{\mathbf{s}_m\}$, with labels $\dot{\mathbf{s}}_1, \ldots, \dot{\mathbf{s}}_{m-1}$ and excitations $\dot{\mathbf{S}}_1, \ldots, \dot{\mathbf{S}}_{m-1}$,

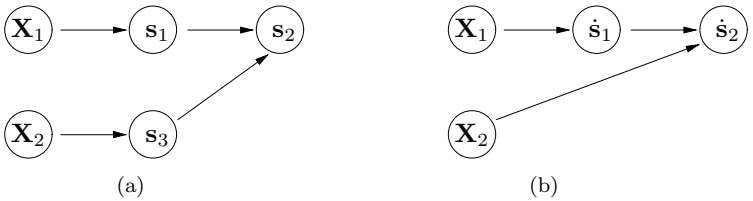

FIGURE 10. Illustration of state variable removal.

respectively. The excitations in $\dot{\mathbf{N}}$ are defined as follows, for all $i \in [m-1]$ and any total state $\mathbf{c} \cdot \dot{\mathbf{d}} \in \mathbf{T}^{n+m-1}$ of $\dot{\mathbf{N}}$:

$$\dot{\mathbf{S}}_i(\mathbf{c} \cdot \dot{\mathbf{d}}) = \mathbf{S}_i(\mathbf{c} \cdot (\dot{\mathbf{d}}, \mathbf{S}_m(\mathbf{d}, \mathbf{t}))),$$

where $\mathbf{t}$ is any transient. In other words, we replace the mth variable with its excitation.

Example 5.1. Consider the network $\mathbf{N}$ with the graph shown in Figure 10a, and excitations $\mathbf{S}_1 = \mathbf{X}_1$, $\mathbf{S}_2 = \mathbf{s}_1 \oplus \mathbf{s}_3$, $\mathbf{S}_3 = \mathbf{X}_2$. In Figure 10b we show the network $\dot{\mathbf{N}}$ obtained from $\mathbf{N}$ by removing $\mathbf{s}_3$. The new excitations are $\dot{\mathbf{S}}_1 = \mathbf{X}_1$, $\dot{\mathbf{S}}_2 = \dot{\mathbf{s}}_1 \oplus \mathbf{X}_2$.

The next proposition states that the results of algorithm A for a network $\mathbf{N}$ and for a reduced version $\dot{\mathbf{N}}$ of $\mathbf{N}$ agree on the variables that are common to $\dot{\mathbf{N}}$ and $\mathbf{N}$, in the sense that the resulting state sequences "sandwich" each other.

Proposition 5.2. *Let $\mathbf{s}^0, \ldots, \mathbf{s}^h, \ldots$ be the sequence resulting from Algorithm A for $\mathbf{N}$ started in binary total state $a \cdot b$. Let $\dot{\mathbf{s}}^0, \ldots, \dot{\mathbf{s}}^g$ be the sequence resulting from Algorithm A for $\dot{\mathbf{N}}$ started in binary total state $a \cdot \dot{b}$, where $\dot{b}_i = b_i$ for all $i \in [m-1]$. Then, for each $h \geq 0$ there exists $j > 0$ such that, for all $i \in [m-1]$,*

$$\mathbf{s}_i^h \leq \dot{\mathbf{s}}_i^h \leq \mathbf{s}_i^{h+j}. \tag{3}$$

Proof. We prove (3) by induction on $h \geq 0$.
Basis, $h = 0$. By the definition of Algorithm A, and the hypothesis of the proposition, we have $\mathbf{s}_i^0 = b_i = \dot{\mathbf{s}}_i^0$, for all $i \in [m-1]$. By the monotonicity of Algorithm A, we know $\mathbf{s}^0 \leq \mathbf{s}^1$. Then $\mathbf{s}_i^0 \leq \dot{\mathbf{s}}_i^0 \leq \mathbf{s}_i^1$, for all $i \in [m-1]$, and the basis holds with $j = 1$.
Induction hypothesis. Suppose (3) holds for some $h \geq 0$.
Induction step. We show that for all $i \in [m-1]$,

$$\mathbf{s}_i^{h+1} \leq \dot{\mathbf{s}}_i^{h+1} \leq \mathbf{s}_i^{h+1+(j+1)}.$$

For any $i \in [m-1]$ the following argument applies:

$$\begin{aligned}
\mathbf{s}_i^{h+1} &= b_i \circ \mathbf{S}_i(\mathbf{a} \cdot \mathbf{s}^h) && \{ \text{ definition of A } \} \\
&\leq b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{s}_m^h)) && \{ \text{ induction hypothesis, monotonicity of excitations, and property of } \circ \} \\
&= b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot \mathbf{s}^{h-1}))) && \{ \text{ definition of A } \} \\
&\leq b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot (\dot{\mathbf{s}}^{h-1}, \mathbf{s}_m^{h-1})))) && \{ \text{ induction hypothesis, monotonicity of excitations, and property of } \circ \} \\
&\leq b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{s}_m^h)))) && \{ \text{ monotonicity of A } \} \text{ monotonicity of excitations, and property of } \circ \}.
\end{aligned}$$

Hence

$$\mathbf{s}_i^{h+1} \leq b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{s}_m^h)))). \tag{4}$$

Next, we note that

$$b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{s}_m^h)))) = \dot{\mathbf{s}}_i^{h+1}, \tag{5}$$

since

$$b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{s}_m^h)))) = b_i \circ \dot{\mathbf{S}}_i(\mathbf{a} \cdot \dot{\mathbf{s}}^h) = \dot{\mathbf{s}}_i^{h+1},$$

by the definitions of $\dot{\mathbf{S}}_i$ and of A, and using a property of $\circ$.

We have

$$\begin{aligned}
& b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{s}_m^h)))) \\
\leq\ & b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\mathbf{s}_1^{h+j}, \ldots, \mathbf{s}_{m-1}^{h+j}, \mathbf{S}_m(\mathbf{a} \cdot \mathbf{s}^{h+j}))) && \{ \text{ induction hypothesis, monotonicity of A and of excitations } \} \\
=\ & b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\mathbf{s}_1^{h+j}, \ldots, \mathbf{s}_{m-1}^{h+j}, \mathbf{s}_m^{h+j+1})) && \{ \text{ definition of A } \} \\
\leq\ & b_i \circ \mathbf{S}_i(\mathbf{a} \cdot \mathbf{s}^{h+j+1}) && \{ \text{ monotonicity of A and of excitations } \} \\
=\ & \mathbf{s}_i^{h+j+2} && \{ \text{ definition of A } \}.
\end{aligned}$$

Thus,

$$b_i \circ \mathbf{S}_i(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{S}_m(\mathbf{a} \cdot (\dot{\mathbf{s}}^h, \mathbf{s}_m^h)))) \leq \mathbf{s}_i^{h+j+2}. \tag{6}$$

By (4)–(6) we obtain the claim of the induction step. □

Corollary 5.3. *Algorithm A terminates on* $\mathbf{N}$ *if and only if A terminates on* $\dot{\mathbf{N}}$. *Let* $\mathbf{s}^H$ *be the final state resulting from A on* $\mathbf{N}$, *and* $\dot{\mathbf{s}}^G$ *that resulting from A on* $\dot{\mathbf{N}}$. *Then,* $\mathbf{s}^H$ *and* $\dot{\mathbf{s}}^G$ *agree on their common variables, that is for all* $i \in [m-1]$,

$$\mathbf{s}_i^H = \dot{\mathbf{s}}_i^G.$$

TABLE 4. Simulation of reduced network.

$\mathbf{X}_1$	$\mathbf{X}_2$	$\dot{\mathbf{s}}_1$	$\dot{\mathbf{s}}_2$	$\dot{\mathbf{s}}_3$	state
0	0	1	1	0	$\dot{\mathbf{s}}^0$
0	0	10	10	0	$\dot{\mathbf{s}}^1$
0	0	10	10	010	$\dot{\mathbf{s}}^2$
0	0	10	10	01010	$\dot{\mathbf{s}}^3$
...	...	...	...	...	...

Proof. This follows immediately from (3). □

It is worth noting that for ternary simulation a relation similar to (3) has been proven [5], but there j is always 1. In our case this does not hold. Consider the network of Figure 9. The simulation of this network started in state $00 \cdot 1100$ is illustrated in Table 2. Suppose we remove variable s_4 and obtain a reduced version of that network with variables $\dot{\mathbf{s}_1}, \dot{\mathbf{s}_2}, \dot{\mathbf{s}_3}$ and excitations $\dot{\mathbf{S}_1} = \mathbf{X}_1, \quad \dot{\mathbf{S}_2} = \mathbf{X}_2, \quad \dot{\mathbf{S}_3} = \overline{s_1 \oplus \overline{s_2 \oplus s_3}}$. The simulation of the reduced network with initial state $00 \cdot 110$ is shown in Table 4. Comparing Tables 2 and 4, we note that $\mathbf{s}_3^3 \leq \dot{\mathbf{s}}_3^3 \not\leq \mathbf{s}_3^4$.

Definition 5.4. A set $\mathcal{F}$ of vertices of a directed graph $\mathcal{G}$ is called a *feedback-vertex* set if every cycle in $\mathcal{G}$ contains at least one vertex from $\mathcal{F}$.

We now show that any feedback-vertex set [5] is sufficient in order to simulate a circuit.

Consider any graph $\mathcal{G}$. If we remove all vertices belonging to a feedback-vertex set $\mathcal{F}$, along with all their incident edges, the resulting graph is acyclic. The following proposition is a slight variation of a similar result from [5].

Proposition 5.5. *For any feedback-vertex set $\mathcal{F}$ of a network $\mathbf{N}$ there exists a reduced network $\dot{\mathbf{N}}$ of $\mathbf{N}$ with vertex set $\mathcal{X} \cup \mathcal{I} \cup \mathcal{F}$, where $\mathcal{I}$ is the set of input delays.*

Proof. See [5].

Theorem 5.6. *Let $\mathcal{F}$ be any feedback-vertex set of a network $\mathbf{N}$, and let $\dot{\mathbf{N}}$ be its reduced version $\dot{\mathbf{N}}$ having vertex set $\mathcal{X} \cup \mathcal{I} \cup \mathcal{F}$. If Algorithm A terminates on $\mathbf{N}$, the final state of $\mathbf{N}$ agrees with that of $\dot{\mathbf{N}}$ with respect to the state variables in $\mathcal{I} \cup \mathcal{F}$.*

Proof. By Corollary 5.3, the results are the same for the remaining variables if only one variable is removed. The claim of the theorem now follows by induction on the number of state variables not in $\mathcal{I} \cup \mathcal{F}$ and by Proposition 5.2. □

6. COVERING OF BINARY ANALYSIS BY SIMULATION

Given the two networks N and $\mathbf{N}$ modeling a gate circuit, we perform binary analysis for N and Algorithm A for $\mathbf{N}$, both with the same starting total state

$a \cdot b$. The binary analysis results in graph $G_a(b)$. Let the state sequence resulting from Algorithm A be $\mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^h, \ldots$, where $\mathbf{s}^h = (\mathbf{s}_1^h, \mathbf{s}_2^h, \ldots, \mathbf{s}_m^h) \in \mathbf{T}^m$, for all $h \geq 0$.

We now show that binary analysis is covered by Algorithm A. Take any path from the initial state b in graph $G_a(b)$. Suppose the length of the path is h. For each state variable s_i, we show that its history along the path is a prefix of the value $\mathbf{s}_i^h$ that variable $\mathbf{s}_i$ takes in the h-th iteration of Algorithm A.

Example 6.1. Consider the binary counterpart of the transient network in Figure 7 with $S_1 = \overline{X_2}$, $S_2 = X_1 \wedge s_1$, $S_3 = \overline{s_2}$, $S_4 = s_2 \vee s_3$. In $G_{11}(1011)$, with the same initial total state as in Example 4.2, we find, for instance, path $\pi = 1011, 1111, 0111, 0001$ of length $h = 3$. The history of variable s_3, for example, is 10. The value of $\mathbf{s}_3$ in the third step of Algorithm A is $\mathbf{s}_3^3 = 101$, which has 10 as a prefix. In fact, this holds for all variables, since $(10, 010, 10, 1) \leq (10, 010, 101, 1010)$.

Theorem 6.2. *For all paths $\pi = s^0, \ldots, s^h$ in $G_a(b)$, with $s^0 = b$, we have $\boldsymbol{\sigma}^\pi \leq \mathbf{s}^h$, where $\mathbf{s}^h$ is the $(h+1)$st state in the sequence resulting from Algorithm A.*

Proof. We prove the theorem by induction on $h \geq 0$.
Basis, $h = 0$. We have $\pi = s^0 = b = \mathbf{s}^0$; hence $\boldsymbol{\sigma}^\pi = s^0 = \mathbf{s}^0$, so the claim holds.
Induction hypothesis. The claim holds for some $h \geq 0$, *i.e.*, for all paths π of length h from b in $G_a(b)$, we have $\boldsymbol{\sigma}^\pi \leq \mathbf{s}^h$.
Induction step. Let $\gamma = s^0, \ldots, s^h, s^{h+1}$ be a path of length $h+1$ from b in $G_a(b)$. Then $\pi = s^0, \ldots, s^h$ is a path of length h, and we have

$$\begin{array}{lcll} \boldsymbol{\sigma}^\gamma & \leq & s^0 \circ \boldsymbol{\Sigma}^\pi & \{ \text{ Cor. 3.5 } \} \\ & \leq & b \circ \mathbf{S}(a \cdot \boldsymbol{\sigma}^\pi) & \{ \ s^0 = b \text{ and Prop. 3.8 } \} \\ & \leq & b \circ \mathbf{S}(a \cdot \mathbf{s}^h) & \{ \text{ induction hypothesis, monotonicity of excitations,} \\ & & & \quad \text{and property of } \circ \ \} \\ & = & \mathbf{s}^{h+1} & \{ \text{ definition of Algorithm A } \}. \end{array}$$ □

Corollary 6.3. *If Algorithm A terminates with state $\mathbf{s}^H$, then for any path π from b in $G_a(b)$, $\boldsymbol{\sigma}^\pi \leq \mathbf{s}^H$.*

Proof. Suppose there exists a path π from b in $G_a(b)$ that satisfies $\boldsymbol{\sigma}_i^\pi > \mathbf{s}_i^H$, for some $i \in [m]$. Let h be the length of π. If $h \leq H$, Theorem 6.2 shows that $\boldsymbol{\sigma}^\pi \leq \mathbf{s}^h$. We also have $\mathbf{s}^h \leq \mathbf{s}^H$, by Proposition 4.1. So $\boldsymbol{\sigma}^\pi \leq \mathbf{s}^H$, and in particular $\boldsymbol{\sigma}_i^\pi \leq \mathbf{s}_i^H$, which contradicts our supposition. If $h > H$, then Theorem 6.2 states that $\boldsymbol{\sigma}^\pi \leq \mathbf{s}^h$. By our convention, $\mathbf{s}^h = \mathbf{s}^H$. So, again we have $\boldsymbol{\sigma}_i^\pi \leq \mathbf{s}_i^H$, which is a contradiction. □

In summary, every history of a variable along a path in the binary analysis is a prefix of the transient predicted by the simulation with Algorithm A. In general, however, simulation may predict more changes than binary analysis, as will be shown next.

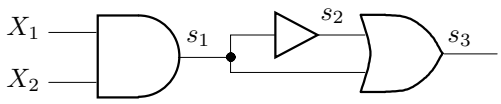

FIGURE 11. Circuit showing the need for wire delays.

TABLE 5. Result of Algorithm Ã.

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	state
01	10	0	0	0	$\mathbf{s}^0$
01	10	010	0	0	$\mathbf{s}^1$
01	10	010	010	010	$\mathbf{s}^2$
01	10	010	010	01010	$\mathbf{s}^3$

7. From simulation to binary analysis

We now study conditions under which binary analysis covers simulation. Although the special case of a feedback-free circuit constructed with 1- and 2-input gates and started in a stable state was solved in [10, 11], the answer is not known in general. For simplicity, we continue to restrict our attention to feedback-free circuits with stable initial states.

Let N and $\mathbf{N}$ be the two networks modeling a circuit. We run Algorithm Ã for $\mathbf{N}$ started in stable total state $\tilde{a} \cdot b$, with the input tuple $\tilde{a}$ changing to a. We also perform binary analysis for N with initial total state $a \cdot b$. We would like to know whether for any state $\mathbf{s}^h$ resulting from Algorithm Ã, there exists a path π from b in $G_a(b)$ such that $\mathbf{s}^h \leq \boldsymbol{\sigma}^\pi$.

7.1. Complete network model

For the result in Section 6, the network model with state variables associated only to gates is sufficient. For the converse of that result we need to take wire delays into account. The following example shows that Algorithm Ã can produce states that are not covered by binary analysis, if delays on the input wires are not taken into account.

Consider the circuit in Figure 11 with associated networks N and $\mathbf{N}$. The excitations in N are $S_1 = X_1 \wedge X_2, \quad S_2 = s_1, \quad S_3 = s_1 \vee s_2$, and those in $\mathbf{N}$ are $\mathbf{S}_1 = \mathbf{X}_1 \otimes \mathbf{X}_2, \quad \mathbf{S}_2 = \mathbf{s}_1, \quad \mathbf{S}_3 = \mathbf{s}_1 \oplus \mathbf{s}_2$.

We run Algorithm Ã for $\mathbf{N}$ started in stable total state $\tilde{a} \cdot b$, where $\tilde{a} = 01$ and $b = 000$; we change the input $\tilde{a}$ to $a = 10$. The result is in Table 5.

Graph $G_{10}(000)$ resulting from the binary analysis of N has only one state, namely 000, since total state $10 \cdot 000$ is stable. Hence, there is no path in $G_{10}(000)$ whose history covers states $\mathbf{s}^1, \mathbf{s}^2$, or $\mathbf{s}^3$ of the simulation. This shows that there exist networks whose binary analysis does not cover simulation.

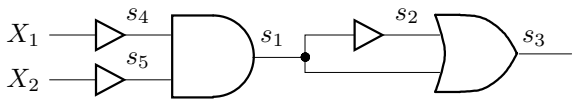

FIGURE 12. Circuit of Figure 11 with input-gates.

TABLE 6. Simulation for circuit of Figure 12.

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{s}_4$	$\mathbf{s}_5$	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	state
01	10	0	1	0	0	0	$\mathbf{s}^0$
01	10	01	10	0	0	0	$\mathbf{s}^1$
01	10	01	10	010	0	0	$\mathbf{s}^2$
01	10	01	10	010	010	010	$\mathbf{s}^3$
01	10	01	10	010	010	01010	$\mathbf{s}^4$

One problem in this example is the static hazard 010 on s_1 predicted by simulation, but not by binary analysis. As in [5], we add input-gate variables to fix this problem.[2] We add an input-gate variable for each input port. The new network is in Figure 12.

The excitations in N are now $S_1 = s_4 \wedge s_5$, $S_2 = s_1$, $S_3 = s_1 \vee s_2$, $S_4 = X_1$, $S_5 = X_2$, and those in $\mathbf{N}$ are $\mathbf{S}_1 = \mathbf{s}_4 \otimes \mathbf{s}_5$, $\mathbf{S}_2 = \mathbf{s}_1$, $\mathbf{S}_3 = \mathbf{s}_1 \oplus \mathbf{s}_2$, $\mathbf{S}_4 = \mathbf{X}_1$, $\mathbf{S}_5 = \mathbf{X}_2$.

For $b = 01000$, the result of the simulation is shown in Table 6. We now write state s as $s = (s_4, s_5, s_1, s_2, s_3)$. This time, in $G_{10}(01000)$ we find path

$$\pi = \underline{0}\,\underline{1}000, 1\underline{1}\,\underline{0}00, 10\underline{1}\,\underline{0}\,\underline{0}, 100\underline{1}\,\underline{1}, 10000$$

that covers states $\mathbf{s}^0, \mathbf{s}^1, \mathbf{s}^2, \mathbf{s}^3$. However, we show next that there is no path that covers state $\mathbf{s}^4$.

We know from Theorem 4.5 that the result of Algorithm $\tilde{\text{A}}$ is the same as the result of Algorithm A for networks containing input-gate variables, and, in particular, for $\mathbf{N}$. Hence, we can apply the propositions and corollaries of Section 6 for the result of Algorithm $\tilde{\text{A}}$ and graph $G_{10}(01000)$ in our example.

Since $\mathbf{s}_1^4 = 010$ and $\mathbf{s}_2^4 = 010$, from Corollary 6.3 we know that, for any path π from 01000 in $G_{10}(01000)$, we have $\boldsymbol{\sigma}_1^\pi \leq 010$ and $\boldsymbol{\sigma}_2^\pi \leq 010$. The graph of Figure 13a shows all possible orders in which (s_1,s_2) may change along paths starting from 01000 in $G_{10}(01000)$. Any path from $(0, 0)$ in this graph shows a possible scenario. Note that we do not claim all such scenarios actually take place in $G_{10}(01000)$. If we replace each pair (s_1, s_2) of values in this graph with the corresponding value of $s_1 \vee s_2$, we obtain the graph of Figure 13b which shows all possible orders in which $s_1 \vee s_2$ may change along paths from the initial state in

[2]Input-gate variables are equivalent to wire variables. We choose to call them input-gate variables to emphasize the fact that they are associated with the inputs.

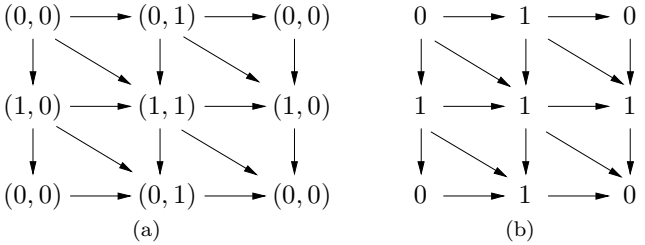

FIGURE 13. Possible changes of s_1, s_2, and $s_1 \vee s_2$.

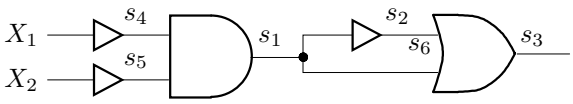

FIGURE 14. Circuit of Figure 12 with a wire variable.

$G_{10}(01000)$. Since $S_3 = s_1 \vee s_2$, this graph gives us the possible excitation histories of s_3. Observe that the longest history of S_3 in this graph, 01010, can be obtained in only two cases:

$$(0,0) \longrightarrow (0,1) \longrightarrow (0,0) \longrightarrow (1,0) \longrightarrow (0,0),$$
or
$$(0,0) \longrightarrow (1,0) \longrightarrow (0,0) \longrightarrow (0,1) \longrightarrow (0,0).$$

With s_3 initially stable, by Corollary 3.6 we know that $\boldsymbol{\sigma}_3^{\pi} \leq 01010$. Suppose there exists a path π from the initial state in $G_{10}(01000)$ with $\boldsymbol{\sigma}_3^{\pi} = 01010$. The two cases above show that π must have a prefix π' with (i) $\boldsymbol{\sigma}_1^{\pi'} = 010$ and $\boldsymbol{\sigma}_2^{\pi'} = 0$, or (ii) $\boldsymbol{\sigma}_1^{\pi'} = 0$ and $\boldsymbol{\sigma}_2^{\pi'} = 010$. Case (ii) is impossible, as explained next.

By Corollary 3.6, $\boldsymbol{\sigma}_2^{\pi'} \leq \boldsymbol{\Sigma}_2^{\pi'}$. Hence case (ii) implies that $\boldsymbol{\Sigma}_2^{\pi'} \geq 010$. But $S_2 = s_1$ means $\boldsymbol{\Sigma}_2^{\pi'} = \boldsymbol{\sigma}_1^{\pi'}$. Therefore $010 \leq \boldsymbol{\sigma}_1^{\pi'}$, which contradicts the fact that in case (ii) $\boldsymbol{\sigma}_1^{\pi'} = 0$.

Thus only case (i) is possible, with $\boldsymbol{\sigma}_1^{\pi'} = 010$, and $\boldsymbol{\sigma}_2^{\pi'} = 0$. It follows that the last state of path π', say s^l, is such that $s_1^l = 0$, and $s_2^l = 0$. In such a state neither one of these variables is unstable and cannot become unstable, since all changes on s_1 have been exhausted. So s_1 and s_2 do not change any more along π. Hence π has $\boldsymbol{\sigma}_3^{\pi} \leq 010$, which contradicts our supposition. This shows that there does not exist a path π from 01000 in $G_{10}(01000)$ with $\boldsymbol{\sigma}_3^{\pi} \geq 01010$. Thus, there is no path from 01000 in $G_{10}(01000)$ that covers $\mathbf{s}^4$.

The problem is solved if we add a wire variable s_6 on the input wire of the OR gate coming from the buffer. We now have the circuit of Figure 14 with $S_6 = s_2$ and $S_3 = s_1 \vee s_6$. The other excitations are unchanged. The simulation with $b = 010000$ is in Table 7 (the inputs are not shown). State s is now $s = (s_4, s_5, s_1, s_2, s_6, s_3)$.

TABLE 7. Simulation for circuit of Figure 14.

$\mathbf{s}_4$	$\mathbf{s}_5$	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_6$	$\mathbf{s}_3$	state
0	1	0	0	0	0	$\mathbf{s}^0$
01	10	0	0	0	0	$\mathbf{s}^1$
01	10	010	0	0	0	$\mathbf{s}^2$
01	10	010	010	0	010	$\mathbf{s}^3$
01	10	010	010	010	010	$\mathbf{s}^4$
01	10	010	010	010	01010	$\mathbf{s}^5$

TABLE 8. Path with desired history.

s_4	s_5	s_1	s_2	s_6	s_3
$\underline{0}$	$\underline{1}$	0	0	0	0
1	$\underline{1}$	$\underline{0}$	0	0	0
1	0	$\underline{1}$	$\underline{0}$	0	$\underline{0}$
1	0	0	$\underline{1}$	$\underline{0}$	$\underline{1}$
1	0	0	0	$\underline{1}$	$\underline{0}$
1	0	0	0	0	$\underline{1}$
1	0	0	0	0	0

In $G_{10}(010000)$ we find path π shown in Table 8 with

$$\boldsymbol{\sigma}^\pi = (01, 10, 010, 010, 010, 01010),$$

which satisfies $\mathbf{s}_i^h \leq \boldsymbol{\sigma}_i^\pi$, for all $i \in [6], h \in [5] \cup \{0\}$.

The examples above motivate us to augment our network model in order to account for wire delays.

We construct the *complete* counterpart of a network N modeling a circuit as follows. For each input X_i, we add an *input-gate* variable s_i; we represent the input gate by a triangle. We also consider each fork as a *fork gate*, and add a variable for each fork output; we represent a fork gate by a rectangle. Finally, we add a variable for each wire. The excitations of the added variables are identity functions, and the excitations of the original variables are updated appropriately to the new set of variables.

Example 7.1. Figure 15a shows a gate circuit consisting of an inverter and an OR gate. It has input variable X_1 and state variables s_a and s_b with excitations $S_a = \overline{X_1}$, and $S_b = X_1 \vee s_a$. We add input-gate variable s_1, fork-gate variables s_3 and s_4, wire variables s_2, s_5, s_7, and s_8, and we relabel s_a as s_6 and s_b as s_9. The resulting complete circuit is in Figure 15b. The new excitations are: $S_1 = X_1,\ S_2 = s_1,\ S_3 = S_4 = s_2,\ S_5 = s_3,\ S_6 = \overline{s_5},\ S_7 = s_6,\ S_8 = s_4,\ S_9 = s_7 \vee s_8$.

Conjecture 7.2. For any feedback-free complete network the results of binary analysis cover those of the simulation with Algorithm A.

For a proof of this conjecture for feedback-free circuits constructed with 1- and 2-input gates see [10, 11].

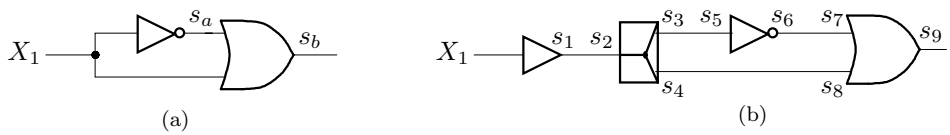

FIGURE 15. Circuit with complete version.

8. Conclusions

Simulation in the algebra of transients provides an efficient method of counting signal changes and detecting hazards in digital circuits. We have shown that simulation covers classical binary analysis in any gate circuit. Our work generalizes the results previously obtained for ternary simulation to simulation with the infinite algebra of transients. The converse result, that binary analysis covers simulation if a sufficient number of delays is taken into account, remains an open problem in the general case.

Acknowledgements. This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871. Most of this work was done while the second author was at the University of Waterloo. A short version of some of the results reported in this paper has appeared previously in [4].

References

[1] Ann. Symp. Asynchronous Circuits and Systems, Proc. 9th *(ASYNC '03), IEEE Comp. Soc.* (2003).

[2] J.A. Brzozowski and Z. Ésik, Hazard algebras. *Formal Methods in System Design* **23** (2003) 223–256.

[3] J.A. Brzozowski, Z. Ésik and Y. Iland, *Algebras for hazard detection. Beyond Two: Theory and Applications of Multiple-Valued Logic*, edited by M. Fitting and E. Orłowska. Physica-Verlag (2003) 3–24.

[4] J.A. Brzozowski and M. Gheorghiu, Simulation of gate circuits in the algebra of transients. Implementation and Application of Automata. *Lect. Notes Comput. Sci.* **2608** (2003) 57–66.

[5] J.A. Brzozowski and C.-J.H. Seger, *Asynchronous circuits.* Springer-Verlag (1995).

[6] J.A. Brzozowski and C.-J.H. Seger, A characterization of ternary simulation of gate networks. *IEEE Trans. Comp.* **C-36** (1987) 1318–1327.

[7] J.A. Brzozowski and M. Yoeli, On a ternary model of gate networks. *IEEE Trans. Comp.* **C-28** (1979) 178–183.

[8] E.B. Eichelberger, Hazard detection in combinational and sequential circuits. *IBM J. Res. Dev.* **9** (1965) 90–99.

[9] J.D. Garside, AMULET3 revealed, in *Proc. 5th Ann. Symp. Asynchronous Circuits and Systems (ASYNC '99)*, *IEEE Comp. Soc.* (1999) 51–59.

[10] M. Gheorghiu, *Circuit simulation using a hazard algebra*. MMath Thesis, Department of Computer Science, University of Waterloo, Waterloo, ON, Canada (2001).
[11] M. Gheorghiu and J.A. Brzozowski, Simulation of feedback-free circuits in the algebra of transients. *Int. J. Found. Comput. Sci.* **14** (2003) 1033–1054.
[12] J. Kessels and P. Marston, Designing asynchronous standby circuits for a low-power pager, in *Proc. 3rd Ann. Symp. Asynchronous Circuits and Systems (ASYNC '97), IEEE Comp. Soc.* (1997) 268–278.
[13] E.J. McCluskey, *Transients in combinational logic circuits. Redundancy techniques for computing systems*, edited by R.H. Wilcox and W.C. Mann. Spartan Books (1962) 9–46.
[14] D.E. Muller and W.C. Bartky, A theory of asynchronous circuits, in *Proc. Int. Symp. on Theory of Switching, Annals of Comp. Lab.* Harvard University **29** (1959) 204–243.
[15] C.-J.H. Seger and J.A. Brzozowski, Generalized ternary simulation of sequential circuits. *Theor. Inf. Appl.* **28** (1994) 159–186.
[16] I.E. Sutherland and J. Ebergen, Computers without Clocks. *Sci. Amer.* **287** (2002) 62–69.
[17] S.H. Unger, *Asynchronous sequential switching circuits.* John Wiley & Sons (1969).

To access this journal online:
www.edpsciences.org