Hazard Algebras *

(Extended Abstract)

J. Brzozowski
Dept. of Computer Science
University of Waterloo
Waterloo, ON, Canada
brzozo@uwaterloo.ca

Z. Ésik
Dept. of Computer Science
University of Szeged
Szeged, Hungary
esik@inf.u-szeged.hu

Abstract

We introduce algebras capable of representing, detecting, identifying, and counting static and dynamic hazard pulses on any wire in a gate circuit. These algebras also permit us to count the number of signal changes on any wire. This is of interest to logic designers for two reasons: each signal change consumes energy, and unnecessary multiple signal changes slow down the circuit operation. We describe efficient circuit simulation algorithms based on our algebras and illustrate them by several examples. Our method generalizes Eichelberger's ternary simulation and several other algebras designed for hazard detection.

1 Introduction

The problem of hazards, i.e., unwanted short pulses on the outputs of gates in logic circuits, is of great importance. In an asynchronous circuit a hazard pulse may cause an error in the circuit operation. Synchronous circuits are protected from such errors, since all actions are controlled by a common clock, and all combinational circuits stabilize before the clock pulse arises. However, an unwanted change in a signal increases the energy consumption in the circuit. From the energy point of view it is necessary not only to detect the presence of unwanted signal changes, but also to count them, in order to obtain an estimate of the energy consumption. Such unwanted changes also add to the computation time. In this paper we address the problem of counting hazards and signal changes in gate circuits.

One of the earliest simulation methods for hazard detection is ternary simulation. A two-pass ternary simulation method was introduced by Eichelberger in 1965 [6], and later studied by others [3]. Ternary simulation is capable of detecting static hazards and oscillations, but does not detect dynamic hazards. A quinary algebra was proposed by Lewis in 1972 [10]

*This research was supported by Grant No. OGP0000871 from the Natural Sciences and Engineering Research Council of Canada, and Grant No. T30511 from the National Foundation of Hungary for Scientific Research.

for the detection of dynamic hazards. A survey of various simulation algebras for hazard detection was given in 1986 by Hayes [9]. We show that several of these algebras are special cases of the hazard algebra presented here.

The remainder of the paper is structured as follows. In Section 2 we discuss our model for representing transients in gate circuits. Using this model, in Section 3 we define Algebra C capable of counting an arbitrary number of signal changes on any wire in a gate circuit. To make the algebra more applicable, in Section 4 we modify it to Algebra C_k, where k is any positive integer; such an algebra is capable of counting and identifying up to $k-1$ signal changes. In Section 5 we describe circuit simulation algorithms based on Algebras C and C_k, and we illustrate our method by several examples. In Section 6 we extend our definitions to arbitrary Boolean functions. Complexity issues are then treated in Section 7, and Section 8 concludes the paper.

We omit all the proofs; they can be found in [1].

2 Transients

We use waveforms to represent changing binary signals. In particular, we are interested in studying transient phenomena in circuits. For this application we consider waveforms with a constant initial value, a transient period involving a finite number of changes, and a constant final value. Waveforms of this type are called *transients*.

Figure 1 gives four examples of transients. With each such transient we associate a binary word, i.e., a sequence of 0s and 1s, in a natural way. In this binary word a 0 (1) represents a maximal interval during which the signal has the value 0 (1). Such an interval is called a *0-interval* (*1-interval*). Of course, no timing information is represented by the binary word. This is to our advantage, however, since we assume that the changes can happen at any time and that the intervals between successive changes can vary arbitrarily.

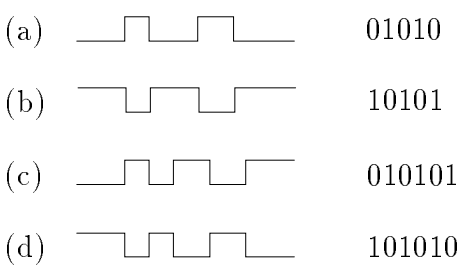

Figure 1: Transients: (a) constant 0 with static hazards; (b) constant 1 with static hazards; (c) change from 0 to 1 with dynamic hazards; (d) change from 1 to 0 with dynamic hazards.

We use $\vee$, $\wedge$, and $^-$ for the Boolean OR, AND, and NOT operations, respectively. For the present, we assume that our circuits are constructed with 2-input OR gates, 2-input AND gates and inverters. Given a transient at each input of a gate, we wish to find the longest possible transient at the output of that gate. For this reason, we need to assume that no

pulses are lost, as might be the case if the delay of the gate were inertial. In this sense we are using the ideal delay model.

We now examine how transients are processed by OR gates, AND gates, and inverters. The case of the inverter is the easiest one. If $t = a_1 \ldots a_j$ is the binary word of a transient at the input of an inverter, then its output has the transient $\overline{t} = \overline{a_1} \ldots \overline{a_j}$. For example, in Fig. 1, the first two transients are complementary, as are the last two.

For the OR and AND gates, we assume that the changes in each input signal can occur at arbitrary times. The following proposition permits us to find the largest number of changes possible at the output of a gate.

PROPOSITION 2.1 *If the inputs of an* OR *gate have m and n 0-intervals respectively, then the maximum number of 0-intervals in the output signal is 0 if $m = 0$ or $n = 0$, and is $m + n - 1$, otherwise.*

Example 1 *Figure 2 shows waveforms of two inputs X_1 and X_2 and output y of an* OR *gate. The input transients are* 010 *and* 1010*, and the output transient is* 101010*. Here, the inputs have two* 0*-intervals each, and the output has three* 0*-intervals, as predicted by Proposition 2.1.* □

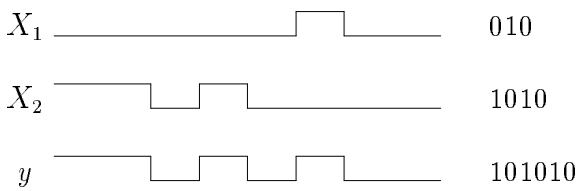

Figure 2: 0-intervals in OR GATE.

By an argument similar to that for Proposition 2.1, we obtain:

PROPOSITION 2.2 *If the inputs of an* AND *gate have m and n 1-intervals respectively, then the maximum number of 1-intervals in the output signal is 0 if $m = 0$ or $n = 0$, and is $m + n - 1$, otherwise.*

These two results will be used in the next section to define operations on transients. In Section 6, it is shown how to extend any Boolean function to transients.

3 Change-Counting Algebra

Let $T = 0(10)^* \cup 1(01)^* \cup 0(10)^*1 \cup 1(01)^*0$; this is the set of all nonempty words over 0 and 1, in which no two consecutive letters are the same. As explained above, elements of T are called transients.

We define the *(signal) change-counting algebra* $C = (T, \oplus, \otimes, \bar{}, 0, 1)$. For any $t \in T$ define $z(t)$ (z for zeros) and $u(t)$ (u for units) to be the number of 0s in t and the number of 1s in t, respectively. Let $\alpha(t)$ and $\omega(t)$ be the first and last letters of t, and let $l(t)$ denote the length of t. For example, if $t = 10101$, then $z(t) = 2$, $u(t) = 3$, $\alpha(t) = \omega(t) = 1$, and $l(t) = 5$.

Operations $\oplus$ and $\otimes$ are binary operations on T intended to represent the worst-case OR-ing and AND-ing of two transients at the inputs of a gate. They are defined as follows:

$$t \oplus 0 = 0 \oplus t = t, \;\; t \oplus 1 = 1 \oplus t = 1,$$

for any $t \in T$. If w and w' are words in T of length > 1, and their *sum* is denoted by $t = w \oplus w'$, then t is that word in T that begins with $\alpha(w) \vee \alpha(w')$, ends with $\omega(w) \vee \omega(w')$, and has $z(t) = z(w) + z(w') - 1$, by Proposition 2.1. For example, $010 \oplus 1010 = 101010$, as illustrated in Fig. 2.

Next, define

$$t \otimes 1 = 1 \otimes t = t, \;\; t \otimes 0 = 0 \otimes t = 0,$$

for any $t \in T$. Consider now the *product* of two words $w, w' \in T$ of length > 1, and denote this product by $t = w \otimes w'$. Then t is that word in T that begins with $\alpha(w) \wedge \alpha(w')$, ends with $\omega(w) \wedge \omega(w')$, and has $u(t) = u(w) + u(w') - 1$, by Proposition 2.2. For example, $0101 \otimes 10101 = 01010101$.

The *(quasi-)complement* $\bar{t}$ of a word $t \in T$ is obtained by complementing each letter in t. For example, $\overline{1010} = 0101$. Finally, the constants 0 and 1 are the words 0 and 1 of length 1.

A *commutative bisemigroup* is an algebra $C = (S, \oplus, \otimes)$, where S is a set, and $\oplus$ and $\otimes$ are associative and commutative binary operations on S, i.e., $(S, \oplus)$ and $(S, \otimes)$ are both commutative semigroups with the same underlying set. Thus a commutative bisemigroup satisfies equations L1, L2, L1′, L2′ in Table 1, where the laws are listed in dual pairs. A commutative bisemigroup is *de Morgan* if it has two constants 0 and 1, and a unary operation $\bar{}$ satisfying L3–L6, L3′, L4′, and L6′. All the laws of Table 1 are also satisfied by Boolean algebras, but several laws of Boolean algebras are not necessarily satisfied by de Morgan bisemigroups. Thus, de Morgan bisemigroups are generalizations of Boolean algebras.

Table 1: Laws of change-counting algebra.

L1	$x \oplus y = y \oplus x$	L1′	$x \otimes y = y \otimes x$
L2	$x \oplus (y \oplus z) = (x \oplus y) \oplus z$	L2′	$x \otimes (y \otimes z) = (x \otimes y) \otimes z$
L3	$x \oplus 1 = 1$	L3′	$x \otimes 0 = 0$
L4	$x \oplus 0 = x$	L4′	$x \otimes 1 = x$
L5	$\overline{\overline{x}} = x$		
L6	$\overline{x \oplus y} = \overline{x} \otimes \overline{y}$	L6′	$\overline{x \otimes y} = \overline{x} \oplus \overline{y}$

PROPOSITION 3.1 *The change-counting algebra* $C = (T, \oplus, \otimes, ^-, 0, 1)$, *is a commutative de Morgan bisemigroup, i.e., it satisfies the equations of Table 1.*

We note some further properties of the operations in C. For any two binary words t and t', 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 the set of binary words, i.e., it is reflexive, antisymmetric and transitive. We use $\leq$ to denote the prefix order.

The prefix order restricted to T is represented by the inequalities below together with the reflexive and transitive laws:

$$0 \leq 01 \leq 010 \leq 0101 \leq 01010 \leq \ldots,$$

$$1 \leq 10 \leq 101 \leq 1010 \leq 10101 \leq \ldots.$$

A function $f(x_1, \ldots, x_n)$ defined on a partially ordered set, ordered by $\leq$, is *monotonic* if is *monotonic* with respect to a partial order $\leq$ if and only if

$$x_1 \leq x'_1, \ldots, x_n \leq x'_n \text{ implies } f(x_1, \ldots, x_n) \leq f(x'_1, \ldots, x'_n).$$

For example, we know that $01 \otimes 10 = 010$. Since $\otimes$ is monotonic, as stated below, and since $01 \leq 0101$ and $10 \leq 10101$, we know that 010 is a prefix of the result $0101 \otimes 10101$. In fact, that result is 01010101.

PROPOSITION 3.2 *The* $\oplus, \otimes$ *and* $^-$ *operations are monotonic with respect to the prefix order.*

The next result shows that the length of a word t cannot be decreased by multiplying it by another word $s \neq 0$, or by adding another word $s \neq 1$.

COROLLARY 3.3 *Suppose that* $t, s \in T$. *If* $s \neq 1$, *then* $l(t) \leq l(t \oplus s)$. *If* $s \neq 0$, *then* $l(t) \leq l(t \otimes s)$.

For any word w, let w^{-1} denote the mirror image of w. It is clear that $w \in T$ iff $w^{-1} \in T$. Moreover,

$$\begin{aligned} (t \oplus s)^{-1} &= t^{-1} \oplus s^{-1} \\ (t \otimes s)^{-1} &= t^{-1} \otimes s^{-1} \\ \overline{t^{-1}} &= (\overline{t})^{-1}, \end{aligned}$$

for all $t \in T$. Since also $(t^{-1})^{-1} = t$, we have:

PROPOSITION 3.4 *The function* $w \mapsto w^{-1}$, $w \in T$, *defines an automorphism of* C, *i.e., a one-to-one and onto mapping* $T \to T$ *which preserves the operations* $\oplus$, $\otimes$, $^-$, *and constants* 0 *and* 1.

It follows now that the operations also preserve the suffix order shown below.

$$0 \preceq 10 \preceq 010 \preceq 1010 \preceq 01010 \preceq \ldots,$$

$$1 \preceq 01 \preceq 101 \preceq 0101 \preceq 10101 \preceq \ldots.$$

4 Counting Changes to a Threshold

Since the underlying set T of Algebra C is infinite, an arbitrary number of changes can be counted. An alternative is to count only up to some threshold $k \geq 0$, and consider all transients with length k or more as equivalent.

Relation $\sim_k$ in algebra $C = (T, \oplus, \otimes, ^-, 0, 1)$ is defined as follows: For $t, s \in T$, $t \sim_k s$ if either $t = s$ or t and s are both of length $\geq k$.

PROPOSITION 4.1 *Relation $\sim_k$ is a congruence relation on C, by which we mean that it is an equivalence relation on T such that for all $t, s, w \in T$, $t \sim_k s$ implies $(w \oplus t) \sim_k (w \oplus s)$, and $\overline{t} \sim_k \overline{s}$; this then implies that $(w \otimes t) \sim_k (w \otimes s)$ whenever $t \sim_k s$.*

The equivalence classes of the quotient algebra $C_k = C/\sim_k$ are of two types. Each transient t with $l(t) < k$ is in a class by itself, and all the words of length $\geq k$ constitute a class. The operations on equivalence classes are as follows. The complement of the class containing t is the class containing $\overline{t}$. The sum (product) of the class containing t and the class containing t' is the class containing $t \oplus t'$ ($t \otimes t'$). Thus, the quotient algebra C_k is a commutative de Morgan bisemigroup with $2k - 1$ elements.

Table 2: OR and AND operations for $k = 2$.

$\oplus$	0	Φ	1
0	0	Φ	1
Φ	Φ	Φ	1
1	1	1	1

$\otimes$	0	Φ	1
0	0	0	0
Φ	0	Φ	Φ
1	0	Φ	1

Example 2 *For $k = 2$, the $\oplus$ and $\otimes$ operations are shown in Table 2. These are the operations of the well known 3-element ternary algebra [3].* □

Example 3 *The* OR *and* AND *operations for the case $k = 3$ are shown in Table 3. This is the quinary algebra introduced in 1972 by Lewis [10], and studied also in [4, 5, 9]. Note that both binary operations are idempotent, i.e., laws L7 and L7′ hold; hence, the algebra is a bisemilattice [4].* □

We now discuss some further properties of Algebras C_k.

PROPOSITION 4.2 *Suppose that $k \geq 2$. Then $\sim_{k-1}$ is the smallest congruence relation strictly containing $\sim_k$.*

COROLLARY 4.3 *Each C_k with $k \geq 2$ is subdirectly irreducible.*

Table 3: OR and AND operations for $k = 3$.

$\oplus$	0	01	Φ	10	1
0	0	01	Φ	10	1
01	01	01	Φ	Φ	1
Φ	Φ	Φ	Φ	Φ	1
10	10	Φ	Φ	10	1
1	1	1	1	1	1

$\otimes$	0	01	Φ	10	1
0	0	0	0	0	0
01	0	01	Φ	Φ	01
Φ	0	Φ	Φ	Φ	Φ
10	0	Φ	Φ	10	10
1	0	01	Φ	10	1

The last result means that C_k is not a subalgebra of a direct product of algebras with fewer elements. This implies, for example, that C_k cannot be expressed in any nontrivial way as an algebra of ordered triples [9], where each component of the triple is evaluated in its own algebra.

As we did for T, we define two partial orders on T_k; these are derived from the prefix and suffix orders. The class consisting of t is $\leq$ the class consisting of t' if and only if t is a prefix of t', and every class is $\leq \Phi$. The second partial order is similarly defined using suffixes. For example, the Hasse diagrams of the two orders on T_4 are shown in Fig. 3.

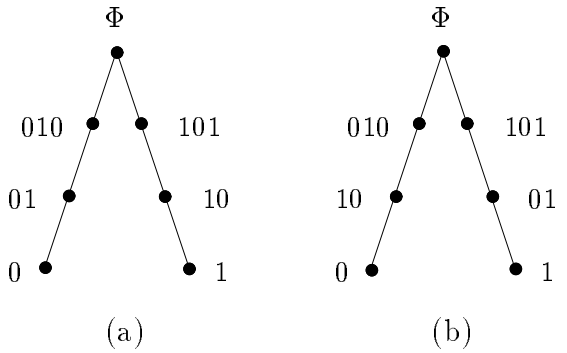

Figure 3: Partial orders on T_4: (a) $\leq$; (b) $\preceq$.

As was the case in C, the three operations $\oplus$, $\otimes$, and $^-$ are monotonic with respect to both partial orders. This property plays an important role in the simulation algorithms described in the next section.

5 Circuit Simulations

The binary analysis of a logic circuit with n gates may have as many as 2^n states. If one can be satisfied with partial information about the circuit behavior, then multivalued simulation is often an efficient method for finding that information.

For the detection of hazards, ternary algebra has been used since 1948 [8]. A two-pass ternary simulation method was introduced by Eichelberger in 1965 [6], and later studied by many authors [3]. The following example illustrates how static hazards are detected by ternary simulation.

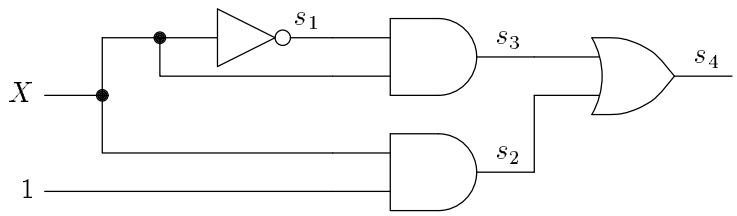

Figure 4: Circuit with hazards.

Table 4: Ternary simulation.

	X	s_1	s_2	s_3	s_4
initial state	0	1	0	0	0
	Φ	1	0	0	0
	Φ	Φ	Φ	Φ	0
result A	Φ	Φ	Φ	Φ	Φ
	1	Φ	Φ	Φ	Φ
	1	0	1	Φ	Φ
result B	1	0	1	0	1

Example 4 *Consider the circuit shown in Fig. 4. Let s_i denote the output of a gate, and S_i, the excitation of that gate. In the Boolean model we have the following excitation equations:*

$$S_1 = \overline{X},\ S_2 = 1 \wedge X,\ S_3 = X \wedge s_1,\ S_4 = s_2 \vee s_3.$$

Refer now to Table 4. The values of the input variable X and the state variables $s_1, \ldots, s_4$ are shown in rows as the simulation progresses. We begin in the initial state 01000*, which is stable. We wish to study the behavior of the circuit when the input changes from* 0 *to* 1 *and is kept constant at* 1*.*

Ternary simulation consists of two algorithms, A and B. In Algorithm A, the input changes to the uncertain or unknown value Φ. Instead of Boolean functions, we now use the ternary extensions of these functions as they are defined in Algebra C_2. Thus we use the excitation equations

$$S_1 = \overline{X},\ S_2 = 1 \otimes X,\ S_3 = X \otimes s_1,\ S_4 = s_2 \oplus s_3.$$

After X changes to Φ, Gates 1, 2, and 3, become unstable in the ternary model. In ternary simulation all unstable variables are changed at the same time. This results in the second

row of Algorithm A. Now Gate 4 becomes unstable and changes, to yield the third row of Algorithm A.

In the case of C_2 the two partial orders $\leq$ and $\preceq$ coincide and are given by

$$0 \leq \Phi,\ 1 \leq \Phi,\ 0 \leq 0,\ 1 \leq 1,\ \Phi \leq \Phi.$$

This partial order is known as the uncertainty *partial order [3].*

In our example, the input became more uncertain by changing from 0 to Φ. Since the gate operations preserve this order, the gate outputs can only become more uncertain. Thus the sequence of states produced by algorithm A is monotonically increasing, and the process terminates in at most n steps if there are n gates in the circuit. Intuitively, in Algorithm A we introduce uncertainty in the circuit inputs and we see how this uncertainty spreads throughout the circuit.

In Algorithm B we start in the state produced by Algorithm A, but now we change the inputs to their final values from Φ, thus reducing uncertainty. In our example, X becomes 1. We again use the ternary excitation functions to see whether the uncertainty will be removed from any gates. This time Algorithm B produces a monotonically decreasing sequence of states, which must terminate in no more than n steps. The final result is the vector 10101, showing that each gate reaches a binary value after the transient is over. Of course, this is the answer we expect since the circuit is feedback-free.

It is clear from the circuit diagram that s_1 changes from 1 to 0 and s_2 changes from 0 to 1 without any hazards. By Boolean analysis, it is easily verified that a static hazard may be present in s_3. It takes a little more effort to show that a dynamic hazard is present in s_4. Specifically, the variables may change in the following sequence $s_3, s_4, s_1, s_3, s_4, s_2, s_4$, if the gate delays have appropriate values [3].

This example illustrates that ternary simulation is capable of detecting static hazards. Gate s_3 is 0 in the beginning and at the end of the transition, but it is Φ at the end of Algorithm A; this indicates a static hazard [3]. Our example also clearly shows that ternary simulation is not capable of detecting dynamic hazards. Gates s_2 and s_4 both change from 0 to Φ to 1, yet s_2 has no dynamic hazard, while s_4 has one. □

In the examples that follow, we show how the accuracy of the simulation improves when we use Algebra C_k as k increases.

Example 5 *In Table 5 we repeat the simulation, this time using quinary extensions of gates as defined in Table 3. Instead to changing X to Φ, we change it to 01 in Algorithm A, since this is the transient we wish to study. From the result of Algorithm A we see that s_1 changes from 1 to 0 and s_2 changes from 0 to 1, both without hazards. The static hazard in s_3 is detected as before, as is the dynamic hazard in s_4. This example shows that quinary simulation is capable of detecting both static and dynamic hazards.* □

Example 6 *We repeat the simulation now using Algebra C_4 with seven values. Refer to Table 5. This time Algorithm A not only reveals that there is a static hazard in s_3, but also identifies it as 010. Note, however, that the dynamic hazard is still not identified.* □

Table 5: Quinary and septenary simulation.

	X	s_1	s_2	s_3	s_4
initial state	0	1	0	0	0
	01	1	0	0	0
	01	10	01	01	0
	01	10	01	Φ	01
result A	01	10	01	Φ	Φ
	1	10	01	Φ	Φ
	1	0	1	10	Φ
result B	1	0	1	0	1

	X	s_1	s_2	s_3	s_4
initial state	0	1	0	0	0
	01	1	0	0	0
	01	10	01	01	0
	01	10	01	010	01
result A	01	10	01	010	Φ
	1	10	01	010	Φ
	1	0	1	10	Φ
result B	1	0	1	0	1

Table 6: Nonary simulation.

	X	s_1	s_2	s_3	s_4
initial state	0	1	0	0	0
	01	1	0	0	0
	01	10	01	01	0
	01	10	01	010	01
result A	01	10	01	010	0101
	1	10	01	010	0101
	1	0	1	10	0101
result B	1	0	1	0	1

Example 7 *We repeat the simulation using Algebra C_5 with nine values. Refer to Table 6. This time Algorithm A identifies the dynamic hazard as* 0101.

Observe that the same table results if we simulate our circuit in Algebra C_k with any $k \geq 5$, or in Algebra C. Note also, that for $k \geq 4$ Algorithm B is no longer necessary, since the entire history of worst-case signal changes is recorded on each wire. □

Example 8 *Consider the circuit of Fig. 5. Algorithm A using Algebra C does not terminate, because there is a nontransient oscillation in the* OR *gate and the wire delay in the feedback loop. For more details see [3]. Simulation in Algebra C_k does, of course, terminate. In Table 7 we show the simulation with C_4. The nontransient oscillation is detected by the presence of Φ in the result of Algorithm B. This example also shows that multiple input changes can be handled by our simulation.* □

In a number of simulators [5, 9], the signal values are ordered triples containing the initial, transient, and final values of a signal. Such algebras can also be described in our framework; for details see [1].

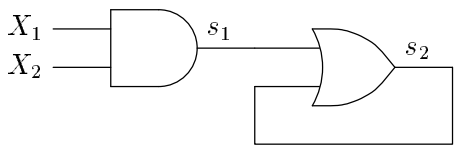

Figure 5: Circuit with nontransient oscillation.

Table 7: Simulation of circuit with nontransient oscillation.

	X_1	X_2	s_1	s_2
initial state	1	0	0	0
	10	01	0	0
	10	01	010	0
	10	01	010	010
$resultA$	10	01	010	Φ
	0	1	010	Φ
$resultB$	0	1	0	Φ

6 Extensions of Boolean Functions

In this section we show that simulation in Algebra C or C_k is not limited to 2-input OR and AND gates and inverters. Let $B = \{0, 1\}$. We now show how to extend any Boolean function $f : B^n \to B$ to a function $\hat{f} : T^n \to T$. We use the notation $[n]$ to mean $\{1, \ldots, n\}$.

Consider a simple example first. Suppose that a 2-input gate performs Boolean function f, and that the two inputs have transients 01 and 101, respectively. We want to find the maximum number of changes that can appear at the output of the gate, assuming that the input changes may occur at any time. Initially, the gate 'sees' the first letters of the two transients, i.e., 0 and 1; hence the output of the gate is $f(0, 1)$. One possibility is that the second transient occurs while the first input has the value 0. Then the gate would see the consecutive ordered pairs $(0, 1)$, $(0, 0)$, $(0, 1)$, and finally $(1, 1)$. Another possible order would be $(0, 1)$, $(0, 0)$, $(1, 0)$, and $(1, 1)$, and the third possible order would be $(0, 1)$, $(1, 1)$, $(1, 0)$, and $(1, 1)$. Note that we do not need to consider sequences like $(0, 1), (1, 0), (1, 1)$, in which both inputs change in some steps, since each such sequence is a subsequence of one of the three sequences above. The corresponding output sequences in each of these cases would be $f(0, 1), f(0, 0), f(0, 1), f(1, 1), f(0, 1), f(0, 0), f(1, 0), f(1, 1)$, and $f(0, 1), f(1, 1), f(1, 0), f(1, 1)$. If f is the OR function, we would have the sequences 1011, 1011, and 1111, respectively. The corresponding transients would be 101, 101, and 1. Since we are looking for the longest possible transient, we have 101 as the result of the operation $01 \oplus 101$, which, of course, agrees with our definition in Section 3. Similarly, if f is the AND function, we get the result 0101, and if it is the XOR function, we have 1010.

This example is generalized as follows. Suppose that $x = (x_1, \ldots, x_n)$, where $x_i \in T$, for each $i \in [n]$. Define the directed graph $D(x)$ to have as vertices n-tuples $y = (y_1, \ldots, y_n)$,

where each y_i is a prefix of length > 0 of x_i, for each $i \in [n]$. There is an edge from vertex $y = (y_1, \ldots, y_n)$ to vertex $y' = (y'_1, \ldots, y'_n)$ iff y and y' differ in exactly one coordinate, say i, and $y'_i = y_i a$, where $a \in B$. For the example given above, we have the graph of Fig. 6.

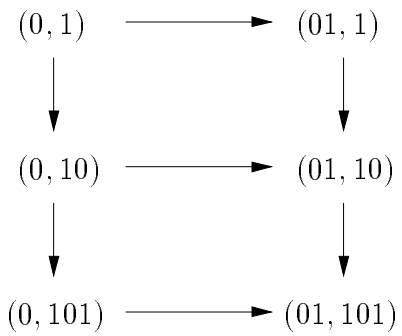

Figure 6: Graph $D(01, 101)$.

Let $x = b_1 \ldots b_n \in T$. Call a word y an *expansion* of x if y results from x by replacing each letter b_i, $i \in [n]$ by $b_i^{m_i}$, for some $m_i > 0$. Note that any nonempty word y over B is the expansion of a unique word $x \in T$. We call x the *contraction* of y.

Returning now to graph $D(x)$, we label each vertex $y = (y_1, \ldots, y_n)$ of $D(x)$ with the value $f(a_1, \ldots, a_n)$, where a_i is the last letter of y_i, i.e., $a_i = \omega(y_i)$, for each $i \in [n]$.

We define function $\hat{f}$ to be that function from T^n to T which, for any n-tuple $(x_1, \ldots, x_n)$ of transients, produces the longest transient when $x_1, \ldots, x_n$ are applied to the inputs of a gate performing Boolean function f. The following proposition is now clear from the definition of $\hat{f}$ and the graph $D(x)$.

PROPOSITION 6.1 *The value of $\hat{f}(x_1, \ldots, x_n)$ is the contraction of the label sequence of those paths in $D(x)$ from $(\alpha(x_1), \ldots, \alpha(x_n))$ to $(x_1, \ldots, x_n)$ which have the largest number of alternations between* 0 *and* 1.

As an example, we now give a complete description of the extension of the n-input XOR function $f : B^n \to B$ to function $\hat{f} : T^n \to T$. For this function, no adjacent vertices have the same label. Hence, the length of the transient $\hat{f}(x_1, \ldots, x_n)$ is the length of any path in $D(x)$ from $(\alpha(x_1), \ldots, \alpha(x_n))$ to $(x_1, \ldots, x_n)$ (all such paths have the same length). One easily verifies that $y = \hat{f}(x_1, \ldots, x_n)$ is that word in T satisfying the conditions

$$\begin{aligned} \alpha(y) &= f(\alpha(x_1), \ldots, \alpha(x_n)) \\ \omega(y) &= f(\omega(x_1), \ldots, \omega(x_n)) \\ l(y) &= 1 + \sum_{i=1}^{n} (l(x_i) - 1) = 1 - n + \sum_{i=1}^{n} l(x_i). \end{aligned}$$

Extensions of the n-input OR, AND, NOR and NAND functions are similarly obtained; see [1] for details. Note that function composition does not preserve extensions in

general. For a two input XOR gate with inputs 01 and 101 the output is 1010. Suppose now that the XOR gate is constructed with OR gates, AND gates and inverters. Then

$$(\overline{01} \otimes 101) \oplus (01 \otimes \overline{101}) = (10 \otimes 101) \oplus (01 \otimes 010) = 1010 \oplus 010 = 101010.$$

Thus, the hazard properties of the XOR function are quite different from those of the network of OR gates, AND gates and inverters.

PROPOSITION 6.2 *$\hat{f}$ is monotonic with respect to the prefix and suffix orders.*

It follows from the definition of $\hat{f}$ that the length of $\hat{f}(x_1, \ldots, x_n)$ is bounded by

$$1 + \sum_{i=1}^{n} (l(x_i) - 1) \quad = \quad 1 - n + \sum_{i=1}^{n} l(x_i).$$

When $n = 1$, this bound can be achieved for the identity function and function $^-$, and for $n > 1$, for the n-input XOR function.

PROPOSITION 6.3 *Suppose that f depends on each of its arguments and none of the x_i is a single letter. Then the length of $\hat{f}(x_1, \ldots, x_n)$ is at least the maximum of the lengths of the x_i.*

PROPOSITION 6.4 *Let $k \geq 2$, and let $\sim_k$ be the equivalence relation on T, defined before. Then $\sim_k$ is a congruence in the sense that, for any Boolean function $f : B^n \to \{0, 1\}$, and $x_1, \ldots, x_n, x'_1, \ldots, x'_n \in T$, if $x_1 \sim_k x'_1, \ldots, x_n \sim_k x'_n$, then $\hat{f}(x_1, \ldots, x_n) \sim_k \hat{f}(x'_1, \ldots, x'_n)$.*

Since $\sim_k$ is a congruence relation on T, it is meaningful to extend any Boolean function to T_k. It follows that the extension preserves the prefix and suffix order on T_k.

The XOR table for $k = 3$ is shown in Table 8.

Table 8: Operation of a XOR gate for $k = 3$.

XOR	0	01	Φ	10	1
0	0	01	Φ	10	1
01	0	Φ	Φ	Φ	10
Φ	Φ	Φ	Φ	Φ	Φ
10	10	Φ	Φ	Φ	01
1	1	10	Φ	01	1

7 Complexity Issues

In this section we give an estimate of the worst case performance of the simulation algorithms.

Using Algebra C, Algorithm A may not terminate unless the circuit is feedback-free. So suppose that we are given an m-input feedback-free circuit with n gates. We assume that each gate is an OR or AND gate, or an inverter, or more generally, a gate with a bounded number of inputs such that the extension of the binary gate function to T can be computed in linear time when each transient t is represented by the triple $(\alpha(t), z(t), \omega(t))$. Here, we assume that $z(t)$, the number of 0s in t is stored in binary.

Suppose the circuit is in a stable state and some of the inputs are supposed to change. If an input is to change from 0 to 1 (1 to 0), we set it to 01 (10). It takes $O(m)$ time to record these changes. Since the circuit contains no cycles, we may assume that the gates are given in topological order with corresponding state variables $s_1, \ldots, s_n$. Initially, each s_i has a binary value. Algorithm A then takes n steps. In step i, variable s_i is set to a new value according to its excitation. For example, when the ith gate is an OR gate which takes its inputs from the jth and kth gates, where $j, k < i$, then s_i is set to $s_j \oplus s_k$, the sum of the current values of s_j and s_k. The $\oplus$ operation is that of Algebra C. Using the triple representation of transients, the middle component of the new value of s_i is at most twice the maximum of the middle components of s_j and s_k, so that its binary representation is at most one longer than the maximum of the lengths of the middle components of s_j and s_k. We see that updating the value of s_i in the ith step takes $O(n)$ time. Thus, in n steps, we can set all n state variables to their final values in $O(n^2)$ time, showing that Algorithm A runs in $O(m + n^2)$ time. Since topological sort can be done within this time limit, the same bound applies if the gates are given in an arbitrary order.

By essentially the same algorithm, we can also compute the total number of signal changes in the entire circuit in $O(m + n^2)$ time. This total can be used as an indication of the worst-case energy consumption for a given input change of the circuit.

If the gates are unsorted, we need to update each state variable in each step. Since Algorithm A terminates in no more than n steps, and since in each step, each state variable can be updated in $O(n)$ time, this second method leads to the time bound of $O(m + n^3)$. However, if we use Algebra C_k, for a fixed k, then at each step, the value of any state variable can be stored in constant space and updated in constant time by a simple table look-up. In this case, Algorithm A terminates in $O(n)$ steps for all circuits, including those with feedback, since T_k is finite, and the subsequent values of each state variable form a monotonically increasing sequence with respect to the prefix order. Monotonicity follows from the fact, proved in [1] that the extension of any Boolean function preserves the prefix order on T_k. Thus, using Algebra C_k, Algorithm A runs in $O(m+n^2)$ time, even for circuits with feedback.

If we use Algorithm A with Algebra C for a feedback-free circuit, then Algorithm B becomes unnecessary. The last letter of the final value of each state variable gives the response of the circuit to the intended change in the input. Moreover, the final value of each state variable is the transient that describes all of the (unwanted) intermediate changes

that can take place in worst case at the respective gate. The same holds if we use C_k for any circuit, which now may contain cycles, such that upon termination of Algorithm A, each state variable assumes a value other than Φ. However, if the final value of a state variable is Φ, Algorithm B does become necessary. It will stop in no more than n steps, since the subsequent values of each state variable now form a monotonically decreasing sequence with respect to the suffix order. The total time required by Algorithm B is $O(n^2)$.

8 Conclusions

We conclude the paper with a short summary of our results. Our main contribution is a general treatment of signal changes and hazards that encompasses the existing methods and permits a systematic study and comparison of these approaches. Some detailed properties of our method are highlighted below.

- **Hazards** We have presented a general theory of simulation of gate circuits for the purpose of hazard detection, identification, and counting.
- **Energy Estimation** The same simulation algorithms can be used to count the number of signal changes during a given input change of a circuit. This provides an estimate of the worst-case energy consumption of that input change.
- **Efficiency** If a circuit has m inputs and n gates, our simulation algorithms run in $O(m + n^2)$ time.
- **Accuracy** By choosing the value of the threshold k one can count signal changes and hazards to any degree of accuracy.
- **Feedback-Free Circuits**
  - In Algebra C, Algorithm A always terminates, and Algorithm B is not required.
  - In Algebra C_k, Algorithm A produces a result without Φs if k is sufficiently large. In that case, Algorithm B is not needed.
  - Simulation with (initial, middle, final) values terminates [1].
- **Circuits with Feedback**
  - In Algebra C, Algorithm A may not terminate.
  - Simulation in Algebra C_k always terminates.
  - Simulation with (initial, middle, final) values may not terminate; hence, it is not suitable for such circuits [1].
- **Multivalued Algebras** Many known algebras are included as special cases in our theory:
  - Ternary algebra is isomorphic to Algebra C_2.
  - The quinary algebra of Lewis [10] is isomorphic to Algebra C_3.

- We can also represent the 6-valued algebra H_6 of Hayes [9] and the 8-valued algebra of Breuer and Harrison [2] and Fantauzzi [7, 9] as quotients of C.

- **Decision Problems [1]**
  - For a given circuit in a stable initial state, a given input change, and an integer k, it is decidable in polynomial time whether there are k or more (unwanted) signal changes on the output of any given gate, or in the entire circuit, during that input change.
  - For a given circuit in a stable initial state and an integer k, it is decidable in nondeterministic polynomial time whether there exists an input change that would cause k or more (unwanted) signal changes on the output of any given gate, or in the entire circuit.

Characterizations of the simulation results described here in terms of the results of binary analysis, extensions to circuits started in unstable states, and other related results will be presented in a companion paper currently in preparation.

Acknowledgement
The authors thank Steve Nowick for his careful reading of our paper and for his constructive comments.

References

[1] J. A. Brzozowski and Z. Ésik, "Hazard Algebras," Maveric Research Report 00-2, University of Waterloo, Waterloo, ON, http://maveric.uwaterloo.ca/publication.html, 27 pp., July 2000.

[2] M. A. Breuer and L. Harrison, "Procedures for Eliminating Static and Dynamic Hazards in Test Generation," *IEEE Trans. on Computers*, vol. C-23, pp. 1069–1078, October 1974.

[3] J. A. Brzozowski and C-J. H. Seger, *Asynchronous Circuits*, Springer-Verlag, 1995.

[4] J. A. Brzozowski, "De Morgan Bisemilattices," *Proc. 30th Int. Symp. on Multiple-Valued Logic*, Portland, OR, pp. 173–178, IEEE Computer Society Press, Los Alamitos, CA, May 2000.

[5] S. Chakraborty and D. L. Dill, "More Accurate Polynomial-Time Min-Max Simulation," *Proc. 3rd Int. Symp. on Advanced Research in Asynchronous Circuits and Systems*, Eindhoven, The Netherlands, pp. 112-123, IEEE Computer Society Press, Los Alamitos, CA, April 1997.

[6] E. B. Eichelberger, "Hazard Detection in Combinational and Sequential Switching Circuits," *IBM J. Research and Development*, vol. 9, pp. 90–99, March 1965.

[7] G. Fantauzzi, "An Algebraic Model for the Analysis of Logic Circuits," *IEEE Trans. on Computers*, vol. C-23, pp. 576–581, June 1974.

[8] M. Goto, "Application of Three-Valued Logic to Construct the Theory of Relay Networks" (in Japanese), *Proceedings of the Joint Meeting of IEE, IECE, and I.of Illumination E. of Japan*, 1948.

[9] J. P. Hayes, "Digital Simulation with Multiple Logic Values," *IEEE Trans. on Computer-Aided Design*, vol. CAD–5, no. 2, April 1986.

[10] D. W. Lewis, *Hazard Detection by a Quinary Simulation of Logic Devices with Bounded Propagation Delays,* MSc Thesis, Electrical Engineering, Syracuse University, Syracuse, NY, January 1972.