EXACT DELAY COMPUTATION WITH TIMED BOOLEAN FUNCTIONS

by

William K.C. Lam, Robert K. Brayton, and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/57

8 May 1992

COVER PAGE

EXACT DELAY COMPUTATION WITH TIMED BOOLEAN FUNCTIONS

by

William K.C. Lam, Robert K. Brayton, and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/57

8 May 1992

TITLE PAGE

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

EXACT DELAY COMPUTATION WITH TIMED BOOLEAN FUNCTIONS

by

William K.C. Lam, Robert K. Brayton, and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/57

8 May 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Exact Delay Computation With Timed Boolean Functions*

William K.C. Lam   Robert K. Brayton   Alberto L. Sangiovanni-Vincentelli
Department of EECS, University of California, Berkeley

*This project is supported by Fannie and John Hertz Foundation.

Contents

Abstract

In this paper, we propose a new technique to represent timing behaviors of digital circuits. This technique, Timed Boolean Functions (TBF), conveniently integrates functionality and timing information of circuits in a form resembling ordinary Boolean functions. Timed Boolean Functions elucidate temporal interactions of various signals in the circuits, and hence, make timing analysis more straightforward. We present some timing properties of combinational circuits, and clarify events that violate the monotone speed-up property. Using Timed Boolean Functions, we give efficient algorithms to compute the *exact* delays of combinational circuits for two cases: 1) the case where gate delays track well, and 2) the general case where gate delays are uncorrelated. In the general case, the problem of computing the exact delay is formulated as a new computational problem, mixed Boolean linear programming, and a lower-bound-progressive-updating algorithm is given which solves this problem efficiently. The algorithms consider a subset of paths at one time; only the paths potentially responsible for the delay of the circuit are considered. Finally, the computation of the carry delay of a 4-bit ripple bypass adder is used to illustrate how Timed Boolean Functions can be represented implicitly with circuits and how some core computations can be translated into test generation and Boolean SAT problems.

1 Introduction

Analyzing timing behavior of logic circuits can be confusing, possibly because of the lack of an intuitive tool to represent the circuits' timing behavior. All existing representation methods have a major drawback that the circuits' timing information and functionalities are not directly and conveniently integrated; hence, temporal interactions of various signals in the circuits are hard to visualize.

In this paper, we propose an algebraic approach, Timed Boolean Functions, to represent circuits' functionalities as well as timing information. This representation captures a circuit's functionality and timing behavior with simple equations. Once a circuit is represented by a Timed Boolean Function, all timing behaviors of the circuit are captured; therefore, all timing properties of the circuit can be verified via algebraic operations on the Timed Boolean Function.

A Timed Boolean Function has three important features. First, its form is similar to ordinary Boolean functions; hence, many techniques in existing logic optimization and synthesis can be used with Timed Boolean Functions with little, if any, modification. Thus, problems in the temporal domain can be translated to the functional domain. Second, certain timing-related problems, for example, path sensitization, hazard detection, timed test generation, wave pipelining, can be systematically formulated with Timed Boolean Functions, and reduced to a basic computation problem: mixed Boolean linear programming, in which a quantity f is optimized over a set of points in the Boolean space; at each point of the Boolean space, the value of f is the optimal value of a linear programming problem associated with the point in the Boolean space. Finally, Timed Boolean Functions enable visualizing temporal interactions of signals in the circuits.

In this paper, we apply Timed Boolean Functions to compute the exact delay of circuits when the inputs are excited by a pair of vectors. Most of proposed solutions to this problem consider only the restricted case where the gate delays are specified by fixed constants. These proposed solutions can be divided into two categories: heuristic and algebraic approaches. Examples of heuristic approaches are [BI88] and [MB89]. Heuristic approaches trade accuracy for speed; only upper bounds are computed. Examples of algebraic approaches are [DKM92] and [HPS91]. (Both [DKM92] and [HPS91] appeared during the preparation of this paper.) When gate delays are specified by lower and upper bounds, except for explicit exhaustive search, all proposed solutions (at the time of this writing) give only an upper bound on the delay of a circuit. In our algebraic approach, we give efficient algorithms to compute the exact delays of circuits when the gate delays are specified by fixed constants or in min-max forms.

The organization of this paper is as follows.

1. Timed Boolean Functions are defined, and some properties are illustrated.
2. Some general timing properties of combinational circuits are proved. In particular, events that violate monotone speed-up properties are clarified.
3. Exact delay computation is formulated as a mixed Boolean linear programming problem and efficient algorithms are presented.

2 Timed Boolean Functions

Definition 1 *1. A waveform space W is a collection of mappings f: $R \mapsto \{0,1\}$. In particular, the unit step function $U(t)$ is defined as follows:*

$$U(t) = \begin{cases} 1 & \textit{if } t > 0 \\ 0 & \textit{if } t < 0 \\ undefined & t=0 \end{cases}$$

2. Timed Boolean Functions (TBF) are defined recursively as follows.

- *$F(v) = v$, $v \in W$, is a TBF.*
- *If $G : W^{n_1} \mapsto W, H : W^{n_2} \mapsto W$ are TBF's, then $F = \overline{G}, F = G \cdot H, F = G + H$ are TBF's.*

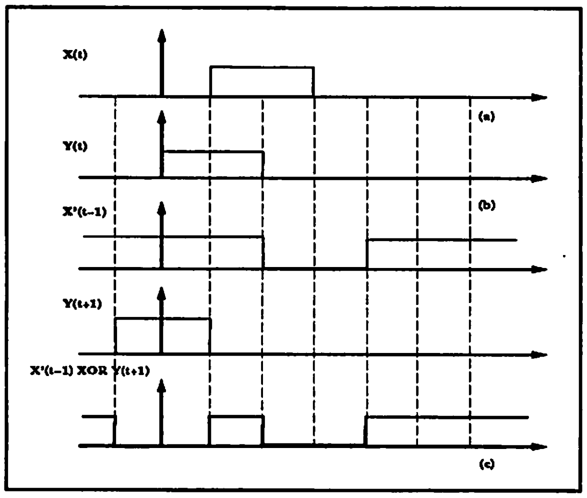

Figure 1: Representing Waveforms by TBF

Example 1 *Let x, $y \in W$ be the waveforms shown in Figure 1(a) and 1(b); then the Timed Boolean Function $f(a,b)(t) = \bar{a}(t-1) \oplus b(t+1)$ represents the waveform shown in Figure 1(c) if a=x, b=y.*

Example 2 *Interpolation with $U(t)$. For any waveform $w(t)$, there exists a Timed Boolean Function f with only one timed Boolean variable such that $w(t)$=$f(U)(t)$. That is, any waveform can be generated by a single step function $U(t)$. Let*

$$w(t) = b_i; \tau_{i-1} \leq t < \tau_i, \ i=1,2,\ldots, \ b_i \in \{0,1\}$$

Then,

$$f(U)(t) = \sum_i b_i \cdot U(t-\tau_{i-1})\overline{U}(t-\tau_i)$$

represents the waveform $w(t)$.

2.1 Modeling Timing Behavior with Timed Boolean Function

Before representing a circuit by a TBF, each component of the circuit needs to be modeled by a TBF. In this section, we demonstrate the modeling capability of TBF's. It will be shown that delay information can be directly incorporated into TBF's, in contrast to previous approaches where delay information is kept separate from circuit representation. The advantage is that temporal interactions among signals are easier to visualize. Here, we only illustrate through examples the modeling process for some commonly encountered gates.

1. Gates characterized by a single delay for each input-output pair. The complex gate shown in Figure 2(a) has three inputs; input x_i has a delay τ_i to the output. This gate

is modeled with the TBF:

$$y(t) = x_1'(t - \tau_1) + x_2(t - \tau_2) + x_3(t - \tau_3).$$

2. Buffer with different rising and falling delays. Let τ_r and τ_f be the rising and falling delays, respectively. If $\tau_r > \tau_f$, then the buffer can be modeled as:

   $$y(t) = x(t - \tau_r) \cdot x(t - \tau_f).$$

   and if $\tau_r < \tau_f$, the buffer can be modeled as:

   $$y(t) = x(t - \tau_r) + x(t - \tau_f).$$

3. Gates with different rising and falling delays for each input-output pair. Rising delay is the delay when the output is rising, likewise for falling delay. Each input is modeled by a buffer with different rising and falling delays; and the "functional block" assumes zero delay. The overall TBF for the gate is obtained through the usual functional composition. An example of an OR gate is shown in Figure 2(b). Input 1 has a rising delay of 1 and a falling delay of 2, while input 2 has a rising delay of 4 and a falling delay of 3. The buffer modeling input 1 is

   $$x_1(t-1) + x_1(t-2).$$

   The buffer modeling input 2 is represented by

   $$x_2(t-4) \cdot x_2(t-3).$$

   Therefore, the OR gate is

   $$x_1(t-1) + x_1(t-2) + x_2(t-4) \cdot x_2(t-3).$$

   A common problem in digital circuit design is the pulse shrinkage or dilation. The pulse shrinkage (dilation) effect occurs when a pulse passes through a chain of gates with unequal rising and falling delays; the pulse width becomes narrower (wider) at the end of the chain. With the above modeling technique, this pulse shrinkage or dilation effect is captured.

2.2 Circuit Formulation with Timed Boolean Function

Once all components of a circuit are represented, the TBF for the circuit can be derived by identifying the timed variables corresponding to the ports connected to the same net. We illustrate this with an example.

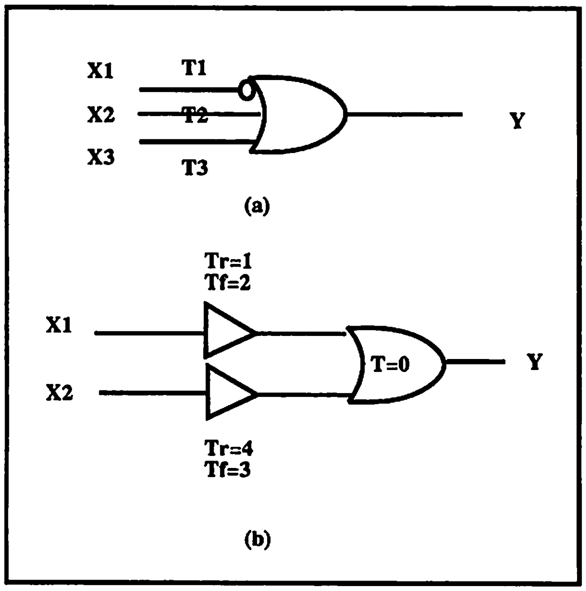

Figure 2: Modeling with TBF

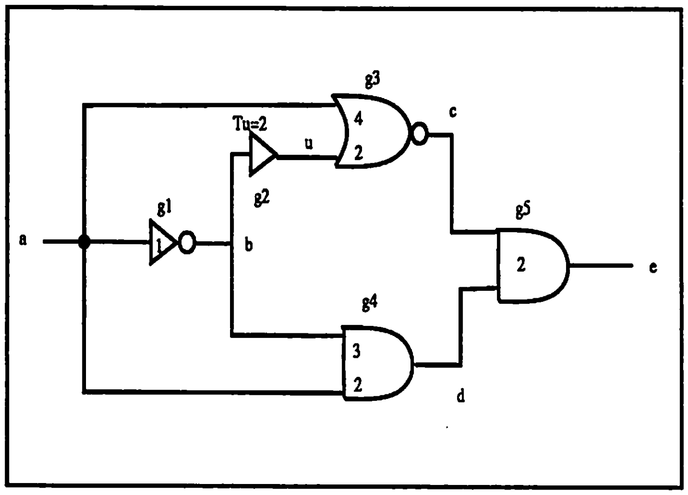

Figure 3: An Example For Delay Computation

Example 3 *The circuit in Figure 3 is taken from [MB89] with wire delays redistributed to gates. For each gate, the delays are labeled next to the input pins.*

First, each gate is represented by a TBF, as follows.

$$\begin{aligned} b(t) &= \overline{a}(t-1) \\ u(t) &= b(t-\tau_u) \\ d(t) &= b(t-3)a(t-2) \\ c(t) &= \overline{a}(t-4)\overline{u}(t-2) \\ e(t) &= c(t-2)d(t-2) \end{aligned}$$

where τ_u=*2*.

We can also flatten the above equations to a two level representation as follows.

$$\begin{aligned} d(t) &= \overline{a}(t-4)a(t-2) \\ c(t) &= \overline{a}(t-4)\overline{u}(t-2) \end{aligned}$$

Therefore,

$$e(t) = \overline{a}(t-6)a(t-5-\tau_u)a(t-4).$$

For $\tau_u = 2, e(t) = a(t-7)\overline{a}(t-6)a(t-4)$.

Comments:

1. When each circuit component is represented by a TBF having the time argument of the form $t - k_i$, k_i a constant, then the TBF for the entire circuit also has only the time arguments of the form $t - k_i$.

2. The Equivalent Normal Form (ENF) is a special case of a TBF; because if the delays from the inputs to the output of the ith gate are all equal to d_i, then the timed variables of the TBF for the circuit are of the form

$$x(t - \sum_{j}^{n} d_{i_j})$$

Replacing each such form by $x_{\{i_j, j=1,\ldots,n\}}$ gives the ENF. Basically, the ENF subscripts a variable according to the path followed by that variable. The TBF displays the delays along the path as arguments.

2.3 Evaluating Timed Boolean Function on Input Waveforms

When properties of input waveforms are known, for instance, the times the input waveforms switch, TBF's can be evaluated accordingly.

Example 4 *Assume the variables in the TBF $f(t) = a(t-1)\overline{a}(t-2)b(t-3)$ take on the waveforms that switch only at $t = 0$. Let $a(0^-)$ be the Boolean value $a(t)$ takes for $t < 0, a(0^+)$ (or simply a), the value $a(t)$ takes for $t > 0$. Similarly for $b(0^-)$ and $b(0^+)$. Then, for $t < 1$, $a(t-1), \overline{a}(t-2), b(t-3)$ become $a(0^-), \overline{a}(0^-), b(0^-)$, respectively. Therefore, $f(t) = 0$ for $t < 1$. For $1 < t < 2$, $a(t-1), \overline{a}(t-2), b(t-3)$ become $a(0^+), \overline{a}(0^-), b(0^-)$, respectively. Therefore, $f(t) = a(0^+)\overline{a}(0^-)b(0^-)$, for $1 < t < 2$, an ordinary Boolean function. Similar evaluations can be done for $2 < t < 3, 3 < t$.*

This evaluation procedure is formalized as follows. Let

$$f(x,t) = \sum_i \prod_j x_{ij}(g_{ij}(t))$$

be a TBF, and the waveforms of x_{ij} switch at times $\{\tau_{ijk}, k = 1, 2, ...\}$. Denote the value of $x_{ij}(t)$ for $\tau_{ijk} \leq t < \tau_{ij(k+1)}$ by a Boolean variable $x_{ij}(k)$; $\tau_{ij(-\infty)}, \tau_{ij\infty}$ are fictitious transition times at $-\infty, \infty$. Partition the time axis by the points of $\{g_{ij}^{-1}(\tau_{ijk})\}$ into intervals $\{I_l\}$. For each interval I_l, there exists some k such that $g_{ij}(I_l) \subseteq [\tau_{ijk}, \tau_{ij(k+1)}]$. Denote this k by $k(l)$. Therefore, for $t \in I_l$, $x_{ij}(g_{ij}(t)) = x_{ij}(k(l))$. Hence,

$$f(x,t) = f(x) = \sum_i \prod_j x_{ij}(k(l)), \text{for } t \in I_l, \forall l$$

an ordinary Boolean function. In the above example, $\{\tau_{ijk}\} = \{0\}, \{g_{ij}^{-1}(\tau_{ijk})\} = \{1,2,3\}$, $\{I_l\} = \{(-\infty,1),(1,2),(2,3),(3,+\infty)\}$, and the symbols 0^- and 0^+ were used to represent $\{k(l)\}$.

2.4 Decision Diagram for Timed Boolean Function

TBF's can be represented with BDD's. In this section, we consider the decision diagrams for TBF's when the input is a pair of vectors switching at $t = 0$. Treat $\{k_i\}$ as binary variables. k_i takes the value of 0 when $t < k_i$, otherwise, 1. A possible good variable ordering is to order k_i's before the normal Boolean variables.

Example 5 *Take the TBF from example 3, $e(t) = a(t-7)\overline{a}(t-6)a(t-4)$. For $t < 4, e(t) = a(0^-)\overline{a}(0^-)a(0^-) = 0$. Similarly, $e(t) = 0 for 4 \leq t < 6,\ 6 \leq t < 7, 7 < t$. Therefore, the decision diagram is as shown in Figure 4(a). The interpretation of the diagram is that for a k_i node, the left branch represents the TBF for $t < k_i$, the right branch for $t > k_i$. When the encountered node is a Boolean variable, the usual BDD interpretation applies. Upon reducing the decision diagram in Figure 3(a), we get the constant function ZERO, as expected; because all paths in this circuit with the specified delays are false. See [MB89].*

If $\tau_u = 0, e(t) = \overline{a}(t-6)a(t-5)a(t-4)$.

$$\begin{array}{ll} t < 5, & e(t) = 0; \\ 5 < t < 6, & e(t) = \overline{a}(0^-)a(0^+) = \overline{a}(0^-)a; \\ 6 < t, & e(t) = 0. \end{array}$$

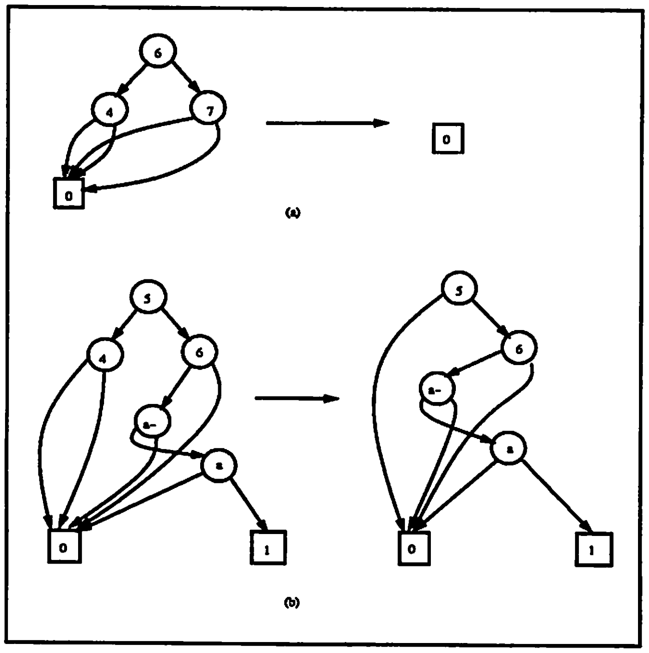

Figure 4: Decision Diagram For TBF

The decision diagrams before and after reduction are shown in Figure 4(b). Extracting the function from the reduced diagram, we get $e(t) = \overline{a}(t-6)a(t-5)$.

Thus, the first part of the BDD sets up time intervals in which the function is a normal Boolean function and this points to a regular BDD in $x(0^-)$ and $x(0^+)$. It can be shown that the above decision diagram is canonical. And two TBF's with identical canonical forms represent the same timing behavior. A way to simplify a design while preserving its timing behavior is to reduce its TBF. Decision diagrams for input waveforms switching at times other than zero can be similarly derived.

3 Event Properties of Combinational Circuits

Definition 2 *1. The derivative with respect to time of a TBF f(t),* $\frac{\partial f(t)}{\partial t}$*, is defined as:*

$$\frac{\partial f(t)}{\partial t} = \lim_{\epsilon \to 0}\{f(t+\epsilon) \oplus f(t-\epsilon)\}$$

2. Let $y(t)$ *be a TBF. Define* $\|y(t)\|_\tau = |\{\tau_i : \frac{\partial y(t)}{\partial t}|_{t=\tau_i} \neq 0\}|$*, i.e. the number of transition times. Let* $\|\xi\|_d$ *be the number of different topological delays from circuit* ξ*'s inputs to its output, i.e. the number of distinct path delays from inputs to output.*

3. *If all the timed variables in a TBF are of the form*

$$x(t-\sum_{l=1}^{n_i} d_{j_l}) = x(t-k_i)$$

where d_i is ith gate's delay variable, $\{k_i\}$ are called the time constants of the TBF. A time constant k_i is effective if there exist input waveforms switching at t=0 that will produce a transition at t=k_i, i.e. $\frac{\partial f(t)}{\partial t}|_{t=k_i} \neq 0$. $\Sigma_i d_i$ is called a delay sum. The set of all delay sums that add up to k_i is denoted by $[k_i]$. For example, for $d_1 = 1, d_2 = 2, d_3 = 1$, the TBF $f(t) = x(t-d_1)+x(t-d_2)+x(t-d_2-d_3)$ becomes $x(t-1)+x(t-2)+x(t-3)$ which has time constants 1, 2 and 3, and $[1] = \{d_1\}, [2] = \{d_2\}, [3] = \{d_2 + d_3\}$. Time constants 2 is not effective, because $f(2^-) = f(2^+) = x(0^-) + x(0^+)$, therefore, $\frac{\partial f(t)}{\partial t}|_{t=2} = 0$. Time constant 1 is effective, because $f(1^-) = x(0^-)$, and $f(1^+) = x(0^-) + x(0^+)$; therefore, $\frac{\partial f(t)}{\partial t}|_{t=1} \neq 0$. Similarly, time constant 3 is effective.

Comments:
$\frac{\partial f(t)}{\partial t}|_{t=\tau} \neq 0$ if and only if there exist inputs that produce a transition at t=τ. $\frac{\partial f(t)}{\partial t}|_{t=\tau}$ is an ordinary Boolean function. A simple application of $\frac{\partial f(t)}{\partial t}$ is seen in the problem of deciding whether there is a transition at time τ at the output of a circuit for a set of input waveforms. A possible solution is to simulate the output waveform for the set of input waveforms. A better solution is to evaluate $\frac{\partial f(t)}{\partial t}|_{t=\tau}$, (as will be seen later, $\frac{\partial f(t)}{\partial t}|_{t=\tau}$ can be represented by a circuit). Then, decide whether $\overline{\frac{\partial f(t)}{\partial t}|_{t=\tau}}$ is a tautology. If it is, there is no transition at τ.

The following theorem gives an upper bound on the number of output transitions, given the number of input transitions..

Theorem 1 *Let circuit ξ have output y and inputs x_i, i=1,...,n. Then*

$$\|y(t)\|_\tau \leq \|\xi\|_d \cdot \|\{x_i(t)\}\|_\tau$$

and if the TBF for ξ has time constants $\{k_i\}$, then

$$\|y(t)\|_\tau \leq | \{k_i\} | \cdot \|\{x_i(t)\}\|_\tau$$

Proof. Let $\{m_{ij}\}$ be the transition times of the input waveform $x_i(t)$'s. A necessary condition for the output $y(t)$ to have a transition at t is that there exist i, j, l, such that $t - k_l = m_{ij}$. The number of distinct such t is equal to the number of distinct values of $m_{ij} + k_l$, which is bounded by the number of combinations of k_l and m_{ij}. This is equal to $| \{k_i\} | \cdot \|\{x_i(t)\}\|_\tau$. But $| \{k_i\} | \leq \|\xi\|_d$. Hence, the claims follow. □

Comments:

1. We conjecture that if $f(t)$ is prime and irredundant, then there exist $x_i(t)$'s such that the equality holds.

2. If $\|\{x_i(t)\}\|_\tau \leq \sum_i \|x_i(t)\|_\tau$, then the equality holds when all inputs $\{x_i(t)\}$ change at different times.

Theorem 2 *If each of the inputs $\{x_i(t)\}$ has only one transition, which occurs at t=0, then,*

1.

$$\|y(t)\|_\tau \leq \|\xi\|_d$$

$$\|y(t)\|_\tau \leq | \{k_i\} | .$$

2. Transitions at the output can occur only at $t = k_i$.

Proof. Follows from the proof of theorem 1.□

Example 6 *From example 3, when $\tau_u = 2, e(t) = a(t-4)\overline{a}(t-6)a(t-7)$. If input a(t) makes a transition only at $t = 0$, then by the above theorem, transitions at the output can occur only at $t = 4, 6, 7$. But $\frac{\partial e(t)}{\partial t} = 0$ at $t = 4, 6, 7$. Therefore, e(t)=0. When $\tau_u = 0, e(t) = \overline{a}(t-6)a(t-5)a(t-4)$. $\frac{\partial e(t)}{\partial t} = 0$ at $t = 4$, and $\frac{\partial e(t)}{\partial t} \neq 0$ at $t = 5, 6$. Thus, there are transitions at $t = 5, 6$, as expected. See [MB89]. Note that this analysis required no waveforms to be plotted to determine where the transitions are.*

In the following sections we will assume that all input waveforms have transitions only at $t = 0$.

4 Delay Properties of Combinational Circuits with Tracking

4.1 Delay Model for Circuits with Tracking Delays

The delays of a manufactured circuit are very difficult to control, because the physical properties that determine the delays, for example, oxide thickness, conductivity, and mobility, are sensitive to fabrication parameters like lithography precision, diffusion temperature, and etching rates, which are not precisely controlled. Therefore, delays of circuits from different wafers may differ substantially. However, gate delays within the same chip usually track well, increasing or decreasing by about the same ratio; this is because physical properties of the devices on the chip are subject to the similar fabrication conditions (due to the small size of a chip), and hence are closely matched. Although small, there are still local variations among the gate delays on the same chip. Therefore, delay variations are mainly caused by these two factors: global variation and local fluctuation. Let r be the delay ratio of a chip with respect to a referenced chip, and let ϵ be the local gate delay fluctuation within a chip.

Then the ith gate delay $d_i \in [r \cdot d_i^0(1-\epsilon), r \cdot d_i^0(1+\epsilon)]$, where d_i^0 is the ith gate delay of the referenced circuit. This bound on d_i is equivalent to:

$$\left|\frac{d_i}{d_j} - \frac{d'_i}{d'_j}\right| \leq \epsilon$$

where d_i is the ith delay of a manufactured circuit, d'_i, the ith delay of another instance of the same circuit. A small ϵ means gate delays track well.

Definition 3 *We say that a manufacturing process has tracking coefficient ϵ if the delays associated with the gates of one circuit $\{d_i\}$ and the delays associated with another manufactured instance of the same circuit $\{d'_i\}$ obey the inequality*

$$\left|\frac{d_i}{d_j} - \frac{d'_i}{d'_j}\right| \leq \epsilon$$

If the delay of a gate is given in terms of

$$d_i^{min} \leq d_i \leq d_i^{max}$$

then

$$\left|\frac{d_i^{min}}{d_j^{min}} - \frac{d_i'^{min}}{d_j'^{min}}\right| \leq \epsilon$$

Similarly for d_i^{max}.

4.2 Delay Properties of Circuits with Tracking Delays

The delay of a combinational circuits is not necessarily monotone; that is, the delay may increase even if the delays of the circuit's components decrease; this is because some false paths may become true as the component delays decrease. Example 3 is such a circuit: the delay increases from 0 to 6 as the gate delay τ_u decreases from 2 to 0.

For practical situations, we assume the delay model discussed in the above section. In a manufacturing process with tracking coefficient ϵ, a time constant k_i in a design may give rise $|\ [k_i]\ |$ time constants in a manufactured circuit; because process variation may cause each delay sum in $[k_i]$ to add up to a slightly different time constant; and there are $|[k_i]|$ delay sums. Let $\{k_{ij}\}$ denote the set of time constants in the manufactured circuit, resulting from k_i due to process variations.

Theorem 3 *Let the TBF $\xi_d(t)$ denote the output function of a design, $\xi_m(t)$, the output function of a manufactured circuit of the design, and r, the ratio of the nominal delays in the manufactured circuit to those in the design. Then*

1. $\xi_d(t) = \xi_m(r \cdot t)$, *for* $t \notin [(1-\frac{\epsilon}{r})k_i, (1+\frac{\epsilon}{r})k_i]$.

2. $\xi_m(t)$ *has at most* $\mid [k_i] \mid$ *transitions in the interval* $[(1-\frac{\epsilon}{r})k_i, (1+\frac{\epsilon}{r})k_i]$.

Proof. If $t \notin [(1-\frac{\epsilon}{r})k_i, (1+\frac{\epsilon}{r})k_i]$, and $k_1 > k_2 > ... > k_m > t > k_n > ... > k_p$. Consider the timed variables in $\xi_d(t)$. The timed variables $x(t-k_i), i \leq m$ become $x(0^-)$, while $x(t-k_i), i \geq n$ become $x(0^+)$, which is the same as the Boolean variable x. Now, consider the timed variables in $\xi_m(t)$. Because the values of $\{k_{ij}\}$ are lower bounded by $(r-\epsilon)k_i$ and upper bounded by $(r+\epsilon)k_i$, and $t \notin [(1-\frac{\epsilon}{r})k_i, (1+\frac{\epsilon}{r})k_i]$, $\{k_{1j}\}, \{k_{2j}\}, ..., \{k_{mj}\} > r \cdot t > \{k_{nj}\}, ..., \{k_{pj}\}$. Therefore, the timed variables in $\xi_m(r \cdot t)$, $x(r \cdot t - k_{ij}), i \leq m$ become $x(0^-)$, while $x(t-k_{ij}), i \geq n$ become $x(0^+)$, which is equal to the Boolean variable x. It can be seen that the two sets of timed variables, $\{x(t-k_i), i \leq m\}$ in $\xi_d(t)$ and $\{x(t-k_{ij}), i \leq m\}$ in $\xi_m(t)$, are the same. So are $\{x(t-k_i), i \geq n\}$ in $\xi_d(t)$ and $\{x(t-k_{ij}), i \geq n\}$ in $\xi_m(t)$. Therefore, $\xi_d(t) = \xi_m(\tau \cdot t)$ for t $\notin [(1-\frac{\epsilon}{r})k_i, (1+\frac{\epsilon}{r})k_i]$.

In the interval $[(1-\frac{\epsilon}{r})k_i, (1+\frac{\epsilon}{r})k_i]$, each delay sum of k_i may add up to a slightly different time constant than k_i, due to process variation. Since there are $[k_i]$ distinct delay sums, each k_i may give rise up to $\mid [k_i] \mid$ distinct time constants due to process variation. Because each time constant may cause at most one transition, and transitions can only occur at $t = k_i$; by theorem 2, hence, there are at most $[k_i]$ transitions in the interval $[(1-\frac{\epsilon}{r})k_i, (1+\frac{\epsilon}{r})k_i]$. □

An interpretation of above theorem is that the output waveform of ξ_m is a scaled version of the output waveform of ξ_d with possibly groups of glitches appearing at the location $r \cdot k_i$. The width of the group of glitches at k_i is bounded by $2\epsilon k_i$. The presence of the groups of glitches is caused by manufacturing process tracking variation. Graphically, transitions or potential transitions at $r \cdot k_i$ split into bands of glitches, as shown in Figure 5.

An interesting consequence is that **the events that cause delays of circuits to violate monotone speedup in a good tracking process are only glitches.** So, adding a glitch eliminating device at the outputs can make the delays of the circuits monotonic, and possibly shorten the delays.

Theorem 4 *In the case of perfect tracking, the delays of combinational circuits obey monotonic speedup.*

Proof. Perfect tracking means $\epsilon = 0$. Therefore, there is no glitch due to process tracking variation. The waveforms of the manufactured circuits are scaled versions of those of the designs; hence, obey the monotone speedup property. □

5 Computing the Exact Delays of Combinational Circuits

In this section, we present algorithms for computing the exact delays of combinational circuits, where the delays of a circuit's components are specified with lower and upper bounds.

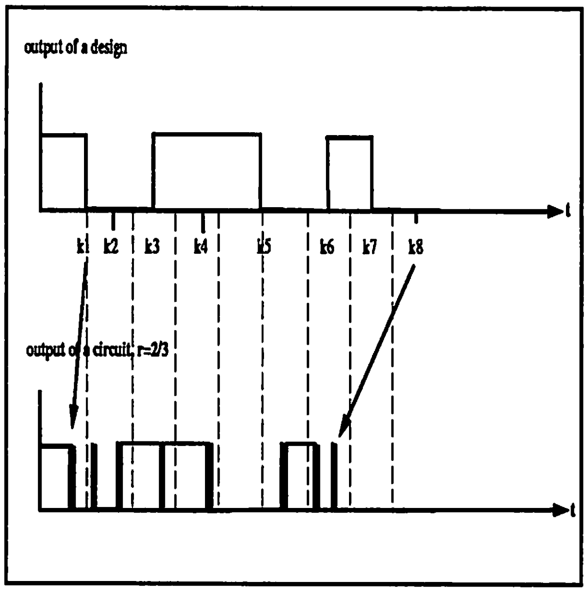

Figure 5: Splitting of Time Constants Into Bands

We consider first the case where the component delays track with coefficient ϵ, then the general case where the component delays are not correlated.

Our approach examines the possibility of transitions only at the instants of time capable of producing a transition, namely, at the time constants of the circuit, instead of investigating sensitization of paths. The advantages are: first, a subset of paths are considered simultaneously, because a time constant may correspond to a subset of paths. Other approaches that consider a subset of paths at one time are [DKM92] and [MB89]. Second, since the number of time constants is roughly proportional to the number of levels of the circuit, the number of time constants is relatively few. Third, only the paths potentially responsible for the delay of the circuit are considered.

5.1 Delays of Circuits with Tracking Coefficient ϵ

We assume that for each component in the circuit, the delay of the component is specified by $[d_i^{min}, d_i^{max}]$, and $\frac{d_i^{min}}{d_i^{max}} = \rho$ is equal to the same constant for all i. Further, for simplicity, we also assume the following definition for tracking coefficient ϵ.

Definition 4 *A manufactured circuit's delays are scaled by a factor p with respect to a reference circuit, with tracking coefficient ϵ, if the ratios of the gate delays of the manufactured circuit to those of the reference circuit are between p and $p - \epsilon$.*

5.1.1 Distribution of Time Constants

Before presenting the algorithm, we consider the distribution of the time constants of manufactured circuits, as the components' delays vary from d_i^{max} to d_i^{min} with tracking coefficient ϵ. Consider first the circuit with each component taking its maximum delay, called the max circuit. This will be used as a reference circuit. Plot the time constants of the max circuit, as shown in Figure 6. The solid vertical lines represent the time constants $\{k_i\}$ that will cause a transition for some inputs, i.e. effective time constants, while the dashed lines represent the time constants that will not cause a transition for all inputs. With respect to the max circuit, as the scaling factor p decreases from 1, the k_i's move toward the origin, and at the same time, each k_i splits into a band of possible time constants resulting from variation of gate delays within the tracking coefficient ϵ. The width of the band at k_i is $\epsilon \cdot k_i$. At small enough value of p, a band will merge with its neighboring bands, but the order the bands started with remains intact as p decreases; that is, bands do not cross each other, because they move with the same ratio p. Eventually, they all collapse to the axis at the origin at $p = 0$.

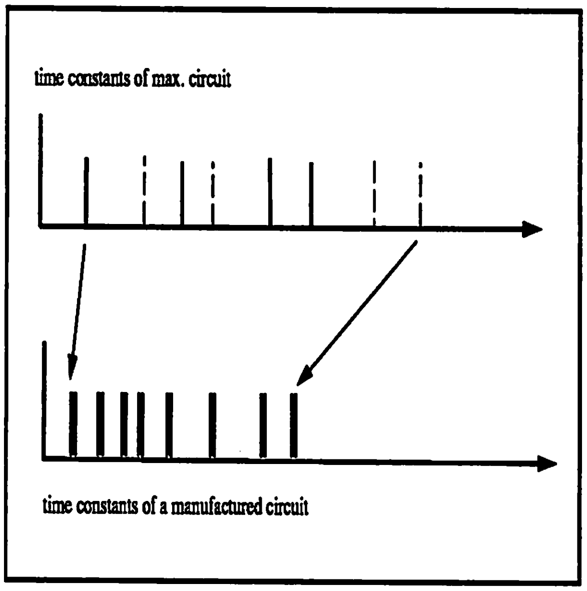
time constants of max. circuit
time constants of a manufactured circuit

Figure 6: Variation of Time Constants in Manufacturing

5.1.2 Computing Delay with Tracking ϵ

By theorem 2, transitions can only occur at $t = k_i$. In the first stage of computation, the algorithm starts with the max circuit and looks for the greatest effective time constant by deciding whether $\frac{\partial f(t)}{\partial t}\,|_{t=k_i} \neq 0$ starting from the greatest time constant k_1. If k_e is the greatest effective time constant found in this stage, then the delay of the circuit is between k_1 and k_e. Hence, the search range for the next stage is from k_e to k_1; so set the lower bound for the delay, D^*, equal to k_e. Note that k_i's $> D^*$ are ineffective.

This stage of computation ends on finding the first effective k_i or none. Call this procedure the **zero order computation.**

Now, we consider only the k_i's $> D^*$, and let the scaling factor p = 1. Since gate delays may vary within the range $[(1-\epsilon)\cdot d_i^{max}, d_i^{max}]$. Each k_i splits into a band of time constants, $\{k_{ij}\}$. The goal in this stage is to decide whether there is an effective k_{ij}.

First partition the time axis into intervals by points $\{(p-\epsilon)\frac{k_i}{m}, p\frac{k_i}{m} > D^*, \forall i, m = 1,2,...\}$. In this case, $p = 1$. Label the intervals from right left by $I_1, I_2, ...$ For an interval I, if $[(p-\epsilon)k_i, pk_i] \supseteq I$, then the $\{k_{ij}\}$ can vary randomly in I. Let $k(I)$ be the set of all time constants in I, i.e. union of $\{k_{ij}\}$ that $[(p-\epsilon)k_i, pk_i] \supseteq I$. An I is effective if $k(I)$ contains an effective time constant.

To determine whether a time constant $k^i \in k(I)$, is effective, we need to first assume the relative magnitudes of $k(I)$, e.g. $k^1 > k^2 > ...$ Then, determine whether $\frac{\partial f(t)}{\partial t} \mid_{t=k^i} = 0$. If $\frac{\partial f(t)}{\partial t} \mid_{t=k^i} \neq 0$, then verify the ordering assumption of $k(I)$ is feasible by determining whether the inequalities resulting from the ordering are satisfiable. If the inequalities are satisfiable, k^i is effective. If there is an effective time constant, set this time constant to be the lower bound D^*.

Because time constants $k(I)$ can vary randomly in I. there are $\mid k(I) \mid!$ possible orders. At the first glance, it seems there are $\mid k(I) \mid!$ cases needed to be considered before it can be concluded that there is no transition in the interval I for all possible orders of time constants in $k(I)$, for all input waveforms. The following lemma asserts that only $2^{|k(I)|}$ cases needed to be considered. Usually, $\mid k(I) \mid$ is small for large I; and the I's examined in delay computation are greater than D^*, hence, are usually large.

Lemma 1 *Let $\{x_{ij}(t-k^i)\}$ be the timed variables of $f(x)$ with $k^i \in k(I)$. Then, the TBF $f(x)$ has no transition possible in the interval I regardless of the relative magnitudes of $k(I)$, if the values of $f(t)$ evaluated at $\{x_{ij}(t-k^i)\} \in \{\{x_{ij}(0^+)\}, \{x_{ij}(0^-)\}\}$ are equal. Thus, the number of cases to be examined to conclude that $k(I)$ are ineffective for all orders is $2^{|k(I)|}$.*

Proof. Let $f(t)$ be the output function of the circuit, and $\{x_{ij}(t-k^i), i = 1,2,...\}$ be the set of timed variables with time constant $k^i \in k(I)$. All $k(I)$ are ineffective in the order $k^1 > k^2 > k^3 > ...$, if $\frac{\partial f(t)}{\partial t} \mid_{t=k} = 0, \forall k \in k(I)$, or equivalently, $f(k^{1+}) = f(k^{1-}) = f(k^{2-}) = f(k^{3-}) = ...$ At $t = k^{1\pm}, k^{j-}, j \neq 1$, the variables $\{x_{ij}(t-k^i)\}$ become either $\{x_{ij}(0^+)\}$ or

$\{x_{ij}(0^-)\}$, according to the following table.

$t =$	$\{x_{ij}(t-k^i)\} =$
k^{1+}	$\{x_{ij}(0^+)\}$
k^{1-}	$\{x_{1j}(0^-)\}, \{x_{2j}(0^+)\}, ..., \{x_{mj}(0^+)\}$
k^{2-}	$\{x_{1j}(0^-)\}, \{x_{2j}(0^-)\}, \{x_{3j}(0^+)\}, ..., \{x_{mj}(0^+)\}$
$\vdots$	$\vdots$
k^{m-}	$\{x_{1j}(0^-)\}, ..., \{x_{mj}(0^-)\}$

That is, $x(t-k)$ becomes $x(0^+)$ at $t > k, x(0^-)$ at $t < k$. It can be seen that for all the orders of $k(I), \{x_{ij}(t-k^i)\}$ is either $\{x_{ij}(0^+)\}$ or $\{x_{ij}(0^-)\}$. Therefore, $k(I)$ are ineffective for all the orders if the values of $f(t)$ are equal for $\{x_{ij}(t-k^i)\} \in \{\{x_{ij}(0^+)\}, \{x_{ij}(0^-)\}\}, i = 1, ..., m$. Thus, there are $2^{|k(I)|}$ cases to consider.□

Let $f(\infty)$ be derived from $f(t)$ by replacing each $x(t-k_i)$ with $x(0^+)$, i.e. $f(t)$ at $t = \infty$, and $\{x_{ij}(t-k^i), j = 1, 2, ...\}$ be the timed variables with time constants $k^i \in k(I)$. Lemma 1 implies the following more efficient method to find D^* Instead of assuming an ordering of $k(I)$, evaluating $\frac{\partial f(t)}{\partial t}$, and verifying the ordering assumption, we assume the sign of $t-k^i$, i.e. $t-k^i > 0$ or $t-k^i < 0$. Let $\sigma = (\sigma_i, \sigma_i, ...)$ be a choice of signs of $t-k^i$'s. Then, determine whether $f(\sigma) = f(\infty)$, where $f(\sigma)$ is derived from $f(t)$ by replacing each $x_{ij}(t-k^i)$ with $x_{ij}(0^+)$, if σ_i is $+$, and with $x_{ij}(0^-)$, if σ_i is -; for timed variables, $x(t-k_s)$, $k_s \notin k(I)$, $x(t-k_s)$ becomes $x(0^-)$ if k_s is above I, $x(0^+)$, if below. If $f(\sigma) \neq f(\infty)$, then a lower bound is the maximum value of a linear programming problem which includes inequalities arising from σ, e.g. $\sigma_i = +$ gives the inequality $t-k^i > 0$. σ is feasible if $f(\sigma) \neq f(\infty)$ and the linear programming problem does not give null result, i.e. the inequality in the linear programming problem are satisfiable. More precisely,

$$D^* = \max_{\sigma} \tau(\sigma) \tag{1}$$

$$f(\sigma) \neq f(\infty)$$

$$\tau(\sigma) = \max t$$

$$t > \sum_k^{n_i} d_{l^i_k} \text{if } \sigma_i = +$$

$$t < \sum_k^{n_i} d_{l^i_k} \text{if } \sigma_i = -$$

$$i = 1, 2, ...$$

$$(1-\epsilon) d_i^{max} \leq d_i \leq d_i^{max}$$

$$t > D_0^*$$

where D_0^* is the lower bound from the zero order computation, σ is a vector of signs of $t-k^i, i=1,2,3,\ldots$. For the linear programming to be meaningful, the strict inequality $a<b$ is replaced by $a \le b+\Delta$, Δ is an arbitrary small positive number. Likewise for $a>b$. From here on, this replacement applies in any linear programming with $<$ and $>$.

Example 7 *In the TBF $f(x_1(t-k_1), x_{21}(t-k_{21}), x_{22}(t-k_{22}), x_{23}(t-k_{23}), x_3(t-k_3))$, $k_1 > \{k_{2i}, i=1,2,3\} > k_3$. Suppose that $\{k_{2i}, i=1,2,3\}$ is the band of k_2, and from the zero order computation, k_1 and k_2 are ineffective, k_3 is effective; so $D_0^* = k_3$; and that $(1-\epsilon)k_2 > k_3$; thus, in the interval $I=[(1-\epsilon)k_2, k_2]$, $k(I)=\{k_{2i}, i=1,2,3\}$. We want to determine whether there is an effective time constant in $k(I)$ for some ordering of $k(I)$. Let $\sigma=(\sigma_1,\sigma_2,\sigma_3)$ denote a choice of sign of $t-k_{21}, t-k_{22}, t-k_{23}$. For $\sigma=(-,+,-)$, $f(\sigma)=f(x_1(0^-), x_{21}(0^-), x_{22}(0^+), x_{23}(0^-), x_3(0^+))$, and $f(\infty)=f(x_1(0^+), x_{21}(0^+), x_{22}(0^+), x_{23}(0^+), x_3(0^+))$. Suppose $f(\sigma)\neq f(\infty)$; and the delay sums are $k_{21}=d_1+d_2+d_3, k_{22}=d_2+d_4, k_{23}=d_1+d_3+d_4$. Then, latest transition for this σ is calculated as follows:*

$$\begin{array}{l} max\ t \\ t < d_1+d_2+d_3 \\ t > d_2+d_4 \\ t < d_1+d_3+d_4 \\ (1-\epsilon)d_i^{max} \le d_i \le d_i^{max} \\ t > D_0^*. \end{array}$$

Note that the delay sums for the bands above and below I are not involved in the linear programming, because their satisfiability is implied by the satisfiability of the inequalities arising from I. As a result, the linear programming is relatively simple. This sign combination is feasible if above linear programming does not give a null result, meaning the inequalities are satisfiable.

If there are m non-empty intervals, then, the number of cases to be examined to conclude the intervals are ineffective is

$$\sum_{i=1}^{m} 2^{n_i}$$

where $n_i = |\ k(I_i)\ |$, and $I_i > D^*$.

The greatest lower bound, D^*, is the maximum value of the linear programming problem over all choices of σ.

This procedure together with the zero order computation is called the **first order computation**, which will be used as a fast means to determine a lower bound on the delay.

Now continue decreasing the scaling factor p until some bands above the lower bound D^* start to merge. Let p_1 be the value of p at which some bands $> D^*$ start to merge at above D^*, and $p_2 < p_1$, be the value of p at which some other bands $> D^*$ start to merge at above D^*. For $p \in [p_1, p_2]$, the overlapping of the merging bands forms new intervals. The method to decide whether a new interval is effective is similar to that in the first order computation. Symbolically,

$$\begin{aligned} D^* &= \max_{\sigma} \tau(\sigma) \qquad (2)\\ f(\sigma) &\neq f(\infty)\\ \tau(\sigma) &= \max\ t\\ t &> \sum_{k}^{n_i} d_{l^i_k} \text{ if } \sigma_i = +\\ t &< \sum_{k}^{n_i} d_{l^i_k} \text{ if } \sigma_i = -\\ i &= 1, 2, \ldots\\ (p-\epsilon) d_i^{max} &\leq d_i \leq p d_i^{max}\\ p_2 &\leq p \leq p_1 \end{aligned}$$

where σ is the sign vector of the timed variables in the new interval. The above procedure is repeated with decreasing p and increasing lower bound D^*, until all bands of time constant above D^* either do not merge or merge below D^* when $p = \rho$. Then the delay of the circuit is D^*.

In summary,

Algorithm For Computing Delay With Tracking ϵ:

1. Let $d_i = d_i^{max}$. Starting from the largest time constant k_1, find the first k_i such that $\frac{\partial f(t)}{\partial t} \mid_{t=k_i} \neq 0$. Set $D^* = k_i$. This is the zero order computation.

2. Partition the time axis into intervals by points $\{(1-\epsilon)k_i, k_i > D^*, \forall i\}$. Label the intervals from right to left by $I_1, I_2, \ldots$ Search for a lower bound D^* in the intervals, starting from I_1. For an interval I_i, Let σ be a vector of signs of the timed variables in I_i. A lower bound D^* is computed as:

$$\begin{aligned} D^* &= \max_{\sigma} \tau(\sigma)\\ f(\sigma) &\neq f(\infty)\\ \tau(\sigma) &= \max\ t\\ t &> \sum_{k}^{n_i} d_{l^i_k} \text{ if } \sigma_i = +\\ t &< \sum_{k}^{n_i} d_{l^i_k} \text{ if } \sigma_i = - \end{aligned}$$

$$i = 1, 2, \ldots$$
$$(1-\epsilon)d_i^{max} \leq d_i \leq d_i^{max}$$

Step 1 and 2 consists of the **first order computation.**

3. Calculate p_1 at which some bands $> D^*$ start to merge above D^* and $p_2 < p_1$ at which some other bands $> D^*$ start to merge above D^*. For $p \in [p_1, p_2]$, the overlapping of the merging bands forms new intervals. Starting from the greatest new interval above D^*, determine a lower bound D^* in the optimization problem (2).

4. If all bands of time constant above D^* either do not merge or merge below D^* when $p = \rho$, then the delay of the circuit $= D^*$; else, go to step 3.

5.2 Validating Condition for First Order Computation

In this section, we derive the conditions under which the first order analysis is sufficient to produce the delay of circuit. Suppose that we have performed the first order computation to get a lower bound D^*, and found that the bands $\{k_{1j}\}, \ldots, \{k_{nj}\} > D^*$ are ineffective. Since D^* is a lower bound, we will ignore the bands $< D^*$ and consider only the bands $> D^*$.

Definition 5 *The disjoint succeeding band of band $B(k_i) = [(p-\epsilon)k_i, pk_i]$ is the greatest band(s) $B(k_j) = [(p-\epsilon)k_j, pk_j]$, i.e. the greatest k_j, satisfying the property:*

1. $k_i > k_j$

2. $B(k_i) \bigcap B(k_j) = \phi$

The lower bound from the first order computation is the true delay if either 1) at any $p \geq \rho$, no bands and their disjoint succeeding bands $> D^*$ merge, or 2) the bands and their disjoint succeeding bands $> D^*$ that merge at $p^* \geq \rho$ merge below D^*. Let $B(k_i)$ be a disjoint succeeding band of band $B(k_j)$, then,

Condition 1) gives:

$$(k_j - k_i) \cdot \rho \geq k_i \cdot \epsilon$$

equivalently,

$$\frac{k_i}{k_j} \leq 1 - \frac{\epsilon}{\rho}$$

Conditions 2) gives:

$$p^* \cdot k_i \leq D^*$$

at

$$p^* = \frac{\epsilon}{1 - \frac{k_i}{k_j}}$$

at which bands of k_i and k_j start to merge. Equivalently,

$$\frac{1}{k_i} - \frac{1}{k_j} \geq \frac{\epsilon}{D^*}$$

Therefore, the lower bound from the first order computation is the true delay of the circuit if at least one of the following conditions is true:

1.
$$\frac{k_i}{k_j} \leq 1 - \frac{\epsilon}{\rho}$$

2.
$$\frac{1}{k_i} - \frac{1}{k_j} \geq \frac{\epsilon}{D^*}$$

Above conditions can be graphically illustrated in terms of the intervals between disjoint succeeding bands, as shown below. Let $l_i = k_j - k_i$. The validating conditions give:

1. $k_i \leq (\frac{\rho}{\epsilon} - 1) \cdot l_i$
2. $k_i \leq \sqrt{\frac{l_i D^*}{\epsilon} + \frac{l_i^2}{4}} - \frac{l_i}{2}$

For a given k_i, the validating values of l_i are the shaded area in Figure 7.

Comments:

1. Figure 7 shows that the intervals between time constants should be farther as the time constants get farther from the lower bound. The order of magnitudes for valid intervals is illustrated in the following example.

   Example 8 *Let the tracking coefficient* $\epsilon = 1\%$, $\rho = \frac{d_i^{min}}{d_i^{max}} = 50\%$, $D^* = 100$ *from the first order computation, and the ineffective bands of time constants above* D^* *be* $k_4 = 102, k_3 = 104, k_2 = 106, k_1 = 108$. *Then, minimum of* $1/k_i - 1/k_j$ = *1/106 - 1/108* = $1.75 \times 10^{-3} > \epsilon/D^* = 10^{-3}$. *Therefore, the first order computation gives the true delay of circuit. The delay of the circuit is therefore* $D^* = 100$. *The intervals between the time constants* $> D^*$ *are less than 2% of the time constants; hence, the first order computation is valid even for very close time constants.*

2. If the lower bound from the first order computation is not the true delay, then some bands $> D^*$merge above D^* at $p^* \geq \rho$. If willing to trade accuracy for speed, we assume all merging bands at $p^* \geq \rho$ produce a feasible time constant; hence, an upper bound for the delay is $p^* \cdot k_m > D^*$, where k_m is the greatest band merged at $p^* \geq \rho$.

3. Bands close together may be considered as a single band in the first order computation, so that other bands are separated far enough for the first order computation to be valid.

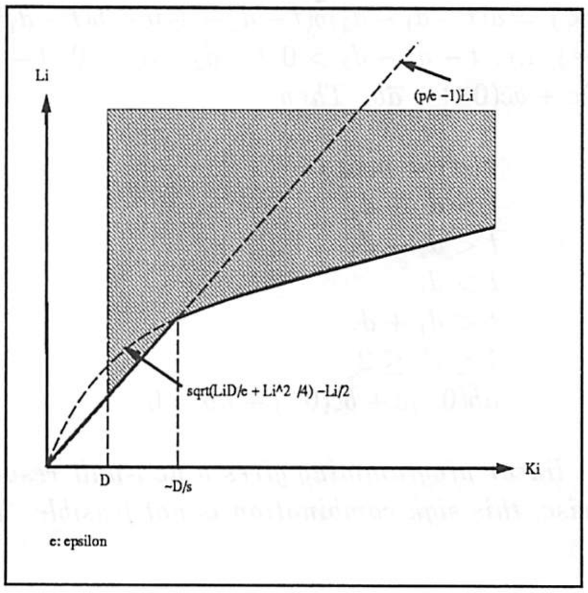

Figure 7: Valid Region For First Order Analysis

5.3 Computing Delay of the General Case: Mixed Boolean Linear Programming

In the general case, gate delays vary independently within the interval $[d_i^{min}, d_i^{max}]$. This situation may rise from delays of chips on circuit boards, and delays of modules on multi-module chips, in which the components are fabricated separately; so their delays are not correlated. Computing the delay of circuit can be formulated as a mixed Boolean linear programming problem, as follows.

$$\text{Delay} = \max t$$
$$\frac{\partial f(t)}{\partial t} \neq 0$$
$$d_i^{min} \leq d_i \leq d_i^{max}, \forall i$$

Let $\sigma = (\sigma_{i1}, ..., \sigma_{ij}, ...)$ be a sign vector, where σ_{ij} is a choice of the sign of $t - \sum_k^{n_{ij}} d_{l_k^{ij}}$ which is a timed variable in $f(t)$. Then,

$$\begin{aligned}
&Delay = \max_\sigma \tau(\sigma) \\
&\tau(\sigma) = \max t \\
&f(\sigma) \neq f(\infty) \\
&t > \textstyle\sum_k^{n_{ij}} d_{l_k^{ij}}, \text{if } \sigma_{ij} = + \\
&t < \textstyle\sum_k^{n_{ij}} d_{l_k^{ij}}, \text{if } \sigma_{ij} = - \\
&d_i^{min} \leq d_i \leq d_i^{max}
\end{aligned}$$

Example 9 *Let $f(t) \oplus f(\infty) = a(t-d_1-d_2)\bar{b}(t-d_2-d_3)c + b(t-d_1)\bar{c}(t-d_1-d_3) + \bar{a}b, 1 \leq d_i \leq 2$. For $\sigma = (+,-,+,-)$, i.e. $t-d_1-d_2 > 0, t-d_2-d_3 < 0$, $t-d_1 > 0, t-d_1-d_3 < 0$, then $f(\sigma) \oplus f(\infty) = a\bar{b}(0^-)c + b\bar{c}(0^-) + \bar{a}b$. Then,*

$$
\begin{aligned}
&\tau(\sigma) = max\ t \\
&t > d_1 + d_2 \\
&t < d_2 + d_3 \\
&t > d_1 \\
&t < d_1 + d_3 \\
&1 \leq d_i \leq 2 \\
&a\bar{b}(0^-)c + b\bar{c}(0^-) + \bar{a}b \neq 0
\end{aligned}
$$

If $f(\sigma) \oplus f(\infty) \neq 0$ and the linear programming gives a non-null result, then $\tau(\sigma)$ is a lower bound for the delay; otherwise, this sign combination is not feasible. The delay of the circuit is $max_\sigma \tau(\sigma)$.

The number of all the possibilities of signs of $t - \sum d_{n_i}$ is exponential in the number of timed variables with different delay sums. The number of timed variables can be reduced progressively in solving the mixed Boolean linear programming problem, as discussed in the following section.

5.4 Solving Mixed Boolean Linear Programming

Observe that $f_{t>\tau}(t)$, $f(t)$ restricted to $t > \tau$, is no more complicated than $f(t)$. Because $f_{t>\tau}(t)$ is derived from $f(t)$ by replacing $x(t-k)$ by x if $k < \tau$. For example, $f(t) = a(t-3)\bar{b}(t-2)\bar{c}(t-1)c(t-2) + \bar{a}(t-1)b(t-2)c(t-3)b$; then $f_{t>2}(t) = a(t-3)\bar{b}\bar{c}c + \bar{a}bc(t-3) = \bar{a}bc(t-3)$. Hence, once a lower bound D^* is computed, $f(t)$ is replaced by $f_{t>D^*}(t)$.

Therefore, use the first order computation to compute a lower bound D^*. Replace $f(t)$ by $f_{t>D^*}(t)$. For a combination of the signs of the timed variables in $f_{t>D^*}(t)$, compute a lower bound, update D^*, recompute $f_{t>D^*}(t)$, and repeat for another sign combination of timed variables in $f_{t>D^*}(t)$. Each time D^* is updated and $f_{t>D^*}(t)$ is recomputed, timed variables $x(t-\Sigma d_i)$ with $\Sigma d_i < D^*$ become ordinary Boolean variables x; Hence, in order to reduce most timed variables, the largest D^* is desired as soon as possible, implying early choices of sign combination should contain as many "+" as possible.

In summary,

Algorithm For Computing Delay Of The General Case

1. Use first order computation to obtain a lower bound D^*.
2.

$$\text{Delay} = \max t$$

$$\frac{\partial f_{t>D^*}(t)}{\partial t} \neq 0$$
$$d_i^{min} \leq d_i \leq d_i^{max}$$
$$t > D^*$$

Specifically,

(a) For a combination, σ, of signs of timed variables in $f_{t>D^*}(t)$, do:

$$\begin{aligned}&\tau(\sigma) = max\, t\\ &f_{t>D^*}(t) \neq f(\infty)\\ &t > \textstyle\sum_k^{n_{ij}} d_{l_k^{ij}} \text{if } \sigma_{ij} = +\\ &t < \textstyle\sum_k^{n_{ij}} d_{l_k^{ij}} \text{if } \sigma_{ij} = -\\ &d_i^{min} \leq d_i \leq d_i^{max}\\ &t > D^*\end{aligned}$$

If σ is feasible, then $D^* = \tau(\sigma)$ and compute $f_{t>D^*}(t)$.

(b) If $f_{t>D^*}(t)$ still has timed variables with unchecked sign combinations, select a sign combination with most "+" and go to step (a), else the delay of the circuit is D^*.

5.5 Delay Specific Enumeration

In a TBF with n timed variables, there are up to 2^n possible combinations of signs of $t - \sum_i d_{n_i}$. However, some combinations are impossible due to constraints imposed by the minimum and maximum values of $\sum_i d_{n_i}$. For instance, the following combinations can not happen: $t - d_1 - d_2 > 0, t - d_3 - d_4 < 0, 1 \leq d_{1,2} \leq 2, 5 \leq d_{3,4} \leq 6$. Therefore, taking into account of the specific values of delays reduces the number of sign combinations.

The procedure for delay specific enumeration is as follows. For each delay sum $S_i = \sum_i d_{n_i}$, plot its range. S_i is minimum (maximum) when all $d'_{n_i}s$ take their minimum (maximum) values. To find the delay, decrease t from the maximum of all delay sums. At t, the sign of $t - \sum_i d_{n_i}$ is "+" if $t > \max \sum_i d_{n-i}$, "-", if $t < \min \sum_i d_{n-i}$, either "+" or "-", otherwise.

Example 10 *In Figure 8, there are 4 timed variables with distinct delay sums, S_i. The shaded areas are the ranges of $S_i = \sum_i d_{n_i}$. At $t = 7, t - S_2$ can be either "+" or "-", because $\min S_2 < t < \max S_2$, while $t - S_1$, $t - S_3$, and $t - S_4$ are all "-"; that is, $t - S_1 > 0, t - S_2 < 0, t - S_3 > 0, t - S_4 > 0$, and $t - S_1 > 0, t - S_2 < 0, t - S_3 > 0, t - S_4 > 0$. At t=5.5, both $t - S_1$ and $t - S_2$ can be either "+" or "-"; thus there are four possible combinations. But the two not yet enumerated combinations are when $t - S_1$ is "-" and $t - S_2$ can be either "+" or "-". Specifically, $t - S_1 < 0, t - S_2 < 0, t - S_3 > 0, t - S_4 > 0$*

and $t - S_1 < 0, t - S_2 > 0, t - S_3 > 0, t - S_4 > 0$. *Similarly at t=2.5, 1.5, ... All together, there are total of 8 combinations instead of* $2^4 = 16$.

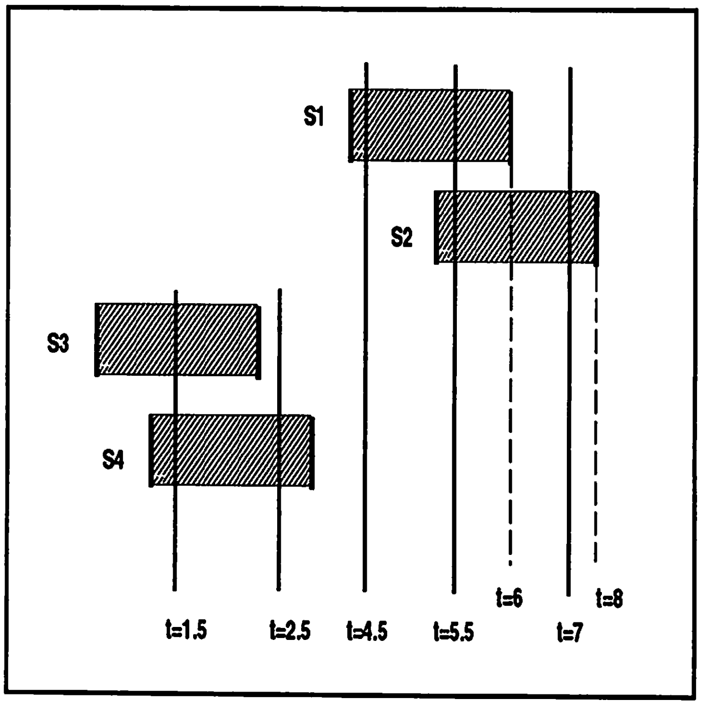

Figure 8: Delay Specific Enumeration

A advantage of delay specific enumeration is that sign combinations leading to larger t can be clearly seen. This is beneficial because in the delay computation a great upper bound is desired as early as possible in the enumerating process. For instance, in Figure 8, the combination $t - S_1 > 0, t - S_2 < 0, t - S_3 > 0, t - S_4 > 0$, corresponding to $t = 7$, is preferred over $t - S_1 < 0, t - S_2 < 0, t - S_3 > 0, t - S_4 > 0$, corresponding to $t = 4.5$.

Another benefit of delay specific enumeration is the simplification of linear programming. Referring to Figure 8, if want to examine a sign combination in region $6 \leq t \leq 8$, linear inequalities $t > 6, t - S_2 < 0$ can be used, instead of $t - S_2 < 0, t - S_1 > 0, t - S_3 > 0, t - S_4 > 0$. be used.

Of course, a combination in delay specific enumeration still needs to be determined feasible by linear programming.

6 An Example: 4-bit Ripple Bypass Adder

In this example, we apply the above algorithm to find the carry output delay of an 4-bit ripple bypass adder, and show how TBF's can be expressed implicitly with circuits and how the timed derivative $\frac{\partial f(t)}{\partial t}$ (as well as $f(\sigma) \oplus f(\infty)$) can be computed using existing techniques in test generation and Boolean SAT. An 4-bit ripple bypass adder is shown in Figure 9. The range of component delays is shown in parenthesis next to each gate. Gate g_0 models the delay from the previous stage. The sum bits are ignored.

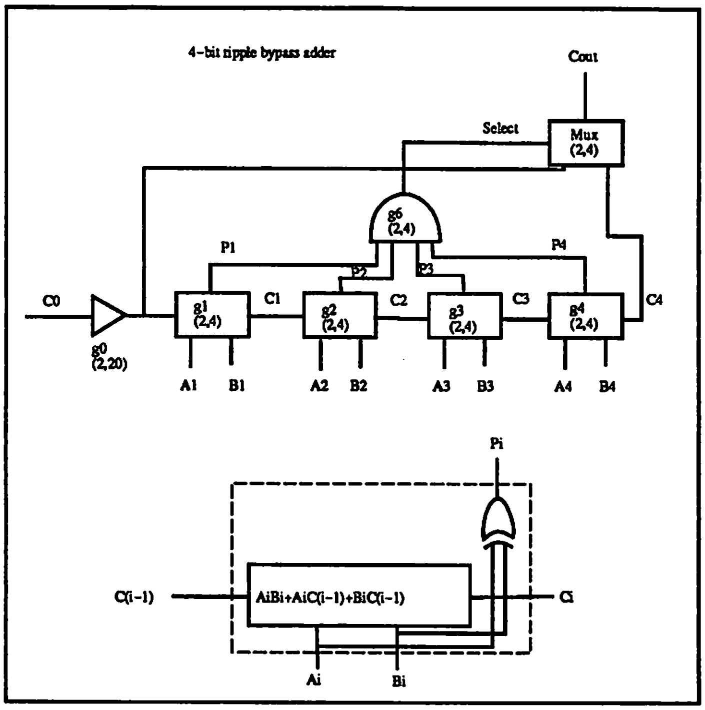

Figure 9: 4-bit Ripple Bypass Adder

1. First order computation. Let each gate takes its maximum delay. The time constants, $k_1, ..., k_n$, in decreasing order, are the delays from the input C_0 to the output C_{out}. $k_1 = 40, k_2 = 24, ...$

2. Evaluate $\frac{\partial f(t)}{\partial t}$. $f(40^+)$ can be represented by the original circuit. For $f(40^-)$, only one path has delay ≥ 40, the path being from C_0 to C_{out} without bypassing; so, the timed variable of this path becomes $C_0(0^-)$, while all other timed variables remain the ordinary Boolean variables. The circuit representing $f(40^-)$ is shown in Figure 10(a). Therefore, $\frac{\partial f(t)}{\partial t}|_{t=40}$ is represented by the circuit in Figure 10(b). Deciding whether $\frac{\partial f(t)}{\partial t}|_{t=40} \neq 0$ can be achieved by using Boolean SAT algorithm [Lar92] or by treating it as a test generation problem for y stuck at 0. In both cases, efficient algorithms are available. The result is that y is not stuck at 0 testable. So, k_1 is ineffective; proceed to $k_2 = 24$, which is the delay from C_0 to C_{out} through bypassing. Because all paths from C_0 to C_{out} have delays ≥ 24 while all other paths have delays < 24, therefore, at

$t = 24^-$, C_o becomes $C_0(0^-)$ while other input variables remain as ordinary Boolean variables. It can be seen that the circuit represent $\frac{\partial f(t)}{\partial t}|_{t=24}$ is that shown in Figure 11 (a) and (b). For $C_0 = 1, C_0(0^-) = 0$, and $A_i \oplus B_i = 1, y = 1$. Therefore, $\frac{\partial f(t)}{\partial t}|_{t=24} \neq 0$. Hence, a lower bound for the delay, D^*, is 24.

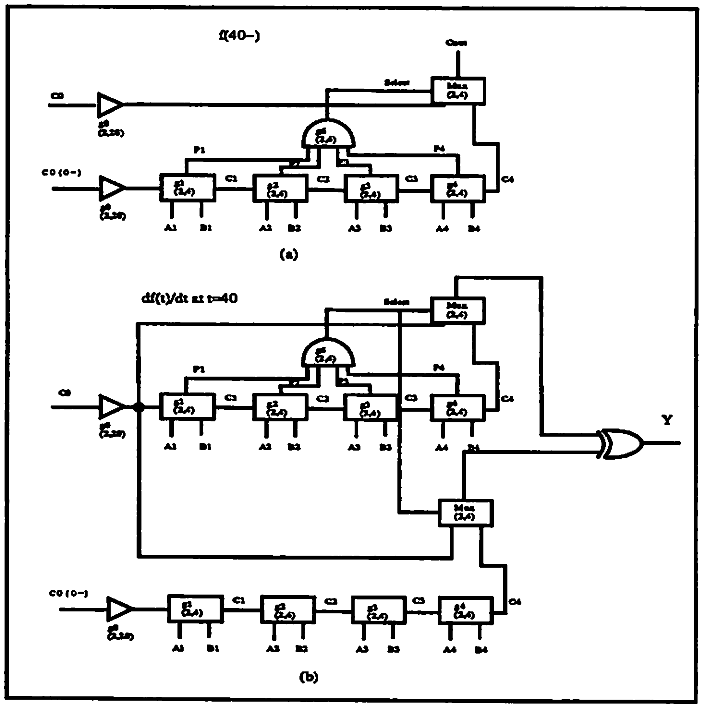

Figure 10: Circuits For TBF Computation

Evaluate $f_{t>24}(t)$. Since the path from C_0 to C_{out} without bypassing is the only path having delay > 24, hence, all inputs except C_0 remain as ordinary Boolean variables. The timed variable is $C_0(t - d_0 - ... - d_5)$; and its sign combinations are already considered. Since there is only one band above D^*, no merging above D^* is possible; thus, first order computation is done. $f_{t>24}(t)$ has only one timed variable, namely, $C_0(t - d_0 - ... - d_5)$.

3. Higher order computation. The only sign combinations are $t - d_0 - ... - d_5 > or < 0$, which have been considered in the previous first order computation. So, no higher order computation is needed. Therefore, the carry delay of the circuit is 24. The path corresponding to k=40, g0,g1,...,g5, is false.

Comments:

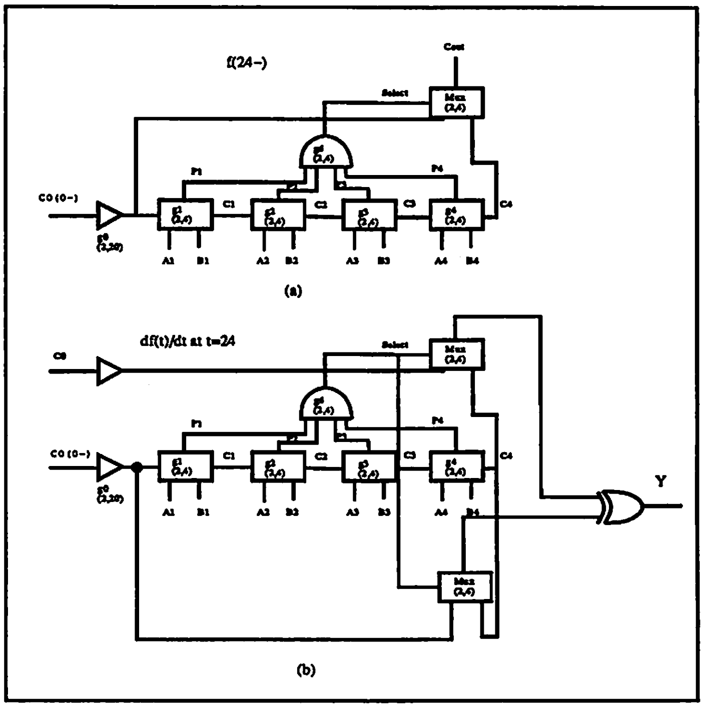

Figure 11: Circuits For TBF Computation

1. The TBF for the adder has 18 timed variables; thus has 2^{18} possible sign enumerations. With the lower bound updating technique, only 2 enumerations were examined in calculating the delay.
2. Although there are many paths, few have large time constants.
3. Using the lower bound from first order computation, only a few timed variables remain in the restricted Timed Boolean Function.
4. Timed Boolean Functions can be expressed implicitly with circuits.
5. Timed derivatives can be evaluated using techniques in test generation, and Boolean SAT.

7 Conclusion

In this paper, we proposed a new technique to represent timing behaviors of digital circuits. This technique, Timed Boolean Functions (TBF), conveniently integrates functionality and timing information of circuits in a form resembling ordinary Boolean functions. Timed

Boolean Functions elucidate temporal interactions of various signals in the circuits, and hence, make timing analysis more straight forward. We presented some timing properties of combinational circuits, and clarified events that violate the monotone speed-up property. Using Timed Boolean Functions, we gave efficient algorithms to compute the *exact* delays of combinational circuits for two cases: 1) the case where gate delays track well, and 2) the general case where gate delays are uncorrelated. In the general case, the problem of computing the exact delay is formulated as a new computational problem, mixed Boolean linear programming, and a lower bound progressive updating algorithm is given which solves this problem efficiently. The algorithms consider a subset of paths at one time; only the paths potentially responsible for the delay of the circuit are considered. Finally, the computation of the carry delay of a 4-bit ripple bypass adder is used to illustrate how Timed Boolean Functions can be represented implicitly with circuits and how some core computations can be translated into test generation and Boolean SAT problems.

References

[BI88] D. Brand and V. Iyengar. Timing analysis using functional analysis. *IEEE Transactions on Computers*, Oct. 1988.

[Bro90] Frank M. Brown. *Boolean Reasoning: the Logic of Boolean Equations.* Kluwer Academic Publishers, 1990.

[DKM91] S. Devadas, K. Keutzer, and S. Malik. Delay computation in combinational logic circuits: Theory and algorithms. *Proc. of the ICCAD*, Nov. 1991.

[DKM92] S. Devadas, K. Keutzer, and S. Malik. Certified timing verification and transition delay of a logic circuit. *Proc. of the Design Automation Conference, (to appear)*, June, 1992.

[HP68] Fredrick Hill and Gerald Peterson. *Introduction to Switching Theory and Logical Design.* John Wiley and Sons, Inc., 1968.

[HPS91] S. Huang, T. Parng, and J. Shyu. A new approach to solving false path problem in timing analysis. *Proc. of the ICCAD*, Nov. 1991.

[JS10] Yun-Cheng Ju and Resve A. Saleh. Incremental techniques for the identification of statically sensitizable critical paths. *28th ACM/IEEE Design Automation Conference*, pages 541–5465, 1910.

[Lar92] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, Jan. 1992.

[MB89] P. McGeer and R. Brayton. Provably correct critical paths. *The Proceedings of the Decennial Caltech VLSI Conference*, 1989.

[Tau82] Herbert Taub. *Digital Circuit and Microprocessors.* McGraw-Hill Book Company, 1982.

[Ung69] Stephen H. Unger. *Asynchronous Sequential Switching Circuits.* John Wiley and Sons, Inc., 1969.