CIRCUIT DELAY MODELS AND THEIR EXACT COMPUTATION USING TIMED BOOLEAN FUNCTIONS

by

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

Memorandum No. UCB/ERL M93/6

19 January 1993

CIRCUIT DELAY MODELS AND THEIR EXACT COMPUTATION USING TIMED BOOLEAN FUNCTIONS

by

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

Memorandum No. UCB/ERL M93/6

19 January 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

CIRCUIT DELAY MODELS AND THEIR EXACT COMPUTATION USING TIMED BOOLEAN FUNCTIONS

by

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

Memorandum No. UCB/ERL M93/6

19 January 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Circuit Delay Models and Their Exact Computation Using Timed Boolean Functions *

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

Abstract

In this paper, we introduce a new circuit delay model, delay by sequences of vectors, which captures the essence of viability and floating delays. Then, we classify delays of circuits according to both the delay models of the gates making up the circuits *and* the family of inputs to the circuits. In this classification, we give sufficient conditions under which floating delay is the same as delay by sequences of vectors; these sufficient conditions are true for most practical circuits. This implies that the assumption of arbitrary node values used in the floating delay model is not too conservative. Thus, delay by sequences of vectors (hence, viability and floating delays), transition delay, and cycle time delay have coherent definitions under the *same* framework.

Next, we study the problem of computing the exact circuit delays under both bounded and unbounded gate delay models, for some of which only upper bounds are known. By using a new formulation technique, called Timed Boolean Function, we formulate the problem of computing the exact delays as a mixed Boolean linear programming problem for which we give efficient algorithms to compute the *exact* delays of combinational circuits for transition delay and delay by sequences of vectors.

The algorithms consider a subset of paths at one time and only the paths potentially responsible for the delay of a circuit are considered. Moreover, the core computation of the algorithms are composed of two computationally efficient algorithms: linear programming and BDD manipulations.

We next compute floating (or viability) delays with the bounded gate delay model and show that delays by sequences of vectors and floating (or viability) delays are invariant under both bounded and unbounded gate delay models. Finally, we address the effect of gate delay lower bounds on delays of circuits. We demonstrate the effectiveness of the method by giving exact delay results for all ISCAS benchmark circuits (except C6188).

*This project is supported by Fannie and John Hertz Foundation and SRC under contract 92-DC-008, whose supports are gratefully acknowledged.

1 Introduction

Research for high performance systems inevitably invokes the problem of delay computation. The pioneering work in [MB89] pointed out pitfalls in static timing analysis and took the innovative approach of computing delays of circuits that observe the so-called monotonic speed-up property: speeding up gate delays in a circuit never slows down the delay of the circuit. Since then there has been much research on delay computation. There are two major types of delay models for combinational circuits. The first type consists of viability delay [MB89] and floating delay [CD90], the second type, transition delay [CD90], [DKM92]. In the first type of delay, node values in a circuit are assumed to be arbitrary before they are determined by an input vector; and the delay of the circuit is the earliest settling time under the input vector. The gate delay model for this type of delay is implicitly the unbounded delay model, i.e. a gate's delay can vary from an upper bound to zero; thus, viability or floating delays under the more general bounded gate delay model (i.e. a gate's delay varies within an interval) have not been studied. On the other hand, for transition delay, a second input vector is applied to a circuit after the circuit has settled under the first vector. The delay of the circuit is the arrival time of the last output transition referenced to the input transition. In this type of delay, gates in a circuit can have either bounded or unbounded delay model. Although transition delay, or 2-vector delay, may not be appropriate for some applications, e.g. asynchronous circuits, in which inputs are not just pairs of vectors, it is a good estimate for cycle times of finite state machines under some conditions, [DKM92], [LBSV92b].

These two types of circuit delay model are not coherent. While transition delays explicitly consider a specific family of input, namely, pairs of vectors, viability and floating delays do not, but assume arbitrary node values. Further, the viability and floating delays are commonly used as upper bounds for cycle times of finite state machines, even thought it is not obvious (no known published results) whether these upper bounds are valid for cycle times. Because viability and floating delays assume implicitly "last input vectors", which may be ambiguous in the cycle time scenario. In fact, short paths in a finite state machine should be considered in using viability and floating delays as upper bounds for cycle times. In this paper, we classify models of circuit delays by both the delay models of the gates in the circuits and the families of input waveforms to the circuits. With this classification, the essence of viability and floating delays is captured by "delay by sequences of vectors". So, under this classification the two types of circuit delay models can be regarded as delays under two different families of input. And viability and floating delays with the bounded gate delay model can be studied with the same methodology as transition delay with the bounded gate delay model. Furthermore, cycle times for finite state machines can also be regarded as a class of delay in this classification with the input family being periodic signals.

Once the meaning of delay is clarified, we investigate the problem of computing exact delays, not just relatively tight upper bounds. Since viability or floating delays with the bounded gate delay model have not been studied, it is not known whether the existing computational techniques [MB89], [CD90] [DKM91] provide the exact delays in this case. The computation for transition delay with the bounded gate delay model in [DKM92] give only upper bounds.

Since computing the exact delays of circuits involve not only logical functionalities but also topological constraints (i.e. bounds on gate delays) of the circuits, an effective representation

mechanism should integrate functionality and timing information to elucidate their interactions. In this paper, we propose an algebraic representation, called Timed Boolean Functions. With this representation, a circuit's logic and timing, hence all its timing behaviors, are captured with simple equations. Therefore, all timing properties of the circuit can be verified via algebraic operations on the Timed Boolean Functions.

With TBF's, we formulate the problem of exact delay computation as a mixed Boolean linear programming problem[1], for which we present algorithms to compute the exact transition delays and delays by sequences of vectors.

2 What is delay?

Before discussing delay computation, we clarify the meaning of delay; how should we define delay so that the definition will reflect our intuitive conception of delay? A common conception is that the delay of a circuit should be the time of the latest output transition (referenced to the last input transition) over all possible input signals and gate delay variations within the specifications. There are two drawbacks to this definition. First, a last input transition is assumed; this assumption may not hold in situations like finite state machines in which the inputs to combinational circuits are periodic. Second, this definition may be too pessimistic by assuming all possible input signals in situations where input signals have a regular pattern, e.g. next state signals to the combinational circuits in finite state machines are periodic, whereas it may be reasonable in asynchronous applications.

Therefore, the delay of a circuit depends not only on the delays of gates in the circuit but also the nature of the input signals. In previous research, the effect of input signals on the delay of a circuit is not clearly distinguished; hence, delays computed assuming arbitrary inputs are only estimates for specific applications. For instance, floating delays are used as estimates for cycle times of finite state machines. Here we propose to classify the delay of a circuit according to the types of input signals as well as the delay model of the gates in the circuit. In an asynchronous environment, input signals to a circuit may be trains of pulses with arbitrary separations between pulses; then the delay of the circuit should be the latest output transition over all input sequences of pulses. That is, we consider sensitization by sequences of pulses. As will be seen later, this definition of delay is equivalent to the floating delay in the literature for most practical conditions. For input signals being a pair of vectors, we call the arrival time of the lastest output transition the transition delay [DKM92] or 2-vector delay. This delay is useful for estimating the cycle times of finite state machines, [DKM92], [LBSV92b]. Finally, when the input signals are sequences of periodic vectors, we call the delay of the combinational circuit its cycle time delay, in accordance with customary use of the term "cycle time".

2.1 Formalization of Delay Models

We would like to generalize the notion of delay models so that the notions of gate delay models and of circuit delays become coherent. First, consider delay models for single simple gates, e.g.

[1] We note that this is different than mixed (0-1) integer programming

AND, OR, INVERTER. From our previous discussion on input waveforms, it is easy to see that for single simple gates the delays with inputs being a sequence of vectors or a pair of vectors are the same; hence, delays of simples gate are well understood without referring to input excitations. Due to manufacturing variations, the delays of simple gates are usually specified by intervals, e.g. $[d^{min}, d^{max}]$, or distribution functions. Therefore, the delay models of simple gates are well defined with $[d^{min}, d^{max}]$ or distribution functions. In this paper we only discuss the first type of delay model.

The delay models of circuits or complex gates can be derived recursively from the same notion of gate delay models. A circuit is a connection of gates (or sub-circuits) with well defined delay models. The delay model of the circuit depends on the circuit topology, delay models of the gates in the circuit, and the family of inputs to the circuit.

Definition 1 *1. Let circuit C be a connection of gates with delay model $\mathcal{M}_g$. and $\mathcal{I}$, the family of inputs to C (assume that circuit C has already settled before the applications of $\mathcal{I}$). Then, the circuit delay model is: $[D^{min}(C,\mathcal{M}_g, \mathcal{I}), D^{max}(C, \mathcal{M}_g, \mathcal{I})]$ for interval type, D^{min} is the earliest arrival time of the last output transition and D^{max}, the latest arrival time of the last output transition.*

2. For $\mathcal{I} \in \{\omega^-, \omega^+, 2, P\}$, where ω^- symbolizes sequences of vectors applied at $t \leq 0$ with the last vector at $t = 0$, ω^+, sequences of vectors applied at $t \geq 0$ with the first vector at $t = 0$, 2, a pair of vectors, and P, periodic sequences of vectors, we define the following models.

- *$D^{max}(C, \mathcal{M}_g, \omega^-)$: the latest arrival time of the last output transition, when a sequence of arbitrary number of vectors of arbitrary intervals between vectors is applied to the inputs at $t \leq 0$, and the last vector is applied at $t = 0$. $D^{min}(C, \mathcal{M}_g, \omega^-)$ is the same as $D^{max}(C, \mathcal{M}_g, \omega^-)$ except it is the earliest arrival time of the last output transition.*
- *$D^{max}(C, \mathcal{M}_g, \omega^+)$: the latest arrival time of the first output transition, when a sequence of arbitrary number of vectors of arbitrary intervals between vectors is applied to the inputs at $t \geq 0$, and the first vector is applied at $t = 0$. $D^{min}(C, \mathcal{M}_g, \omega^+)$ is the same as $D^{max}(C, \mathcal{M}_g, \omega^+)$ except it is the earliest arrival time of the first output transition.*
- *$D^{max}(C, \mathcal{M}_g, 2)$: the latest arrival time of the last output transition, when a pair of vectors are applied with the first vector applied at $t = -\infty$, the second vector, at $t = 0$. $D^{min}(C, \mathcal{M}_g, 2)$ is the earliest arrival time of the first output transition under the same setting.*
- *$D^{max}(C, \mathcal{M}_g, P)$: minimum cycle time of an FSM. This will be defined more precisely in another paper. For now $D^{min}(C, \mathcal{M}_g, P)$: not defined.*

3. The gate delay models considered are: $\mathcal{M}_g \in \{[d_i^{min}, d_i^{max}], [d_i^{max}, d_i^{max}], [0, d_i^{max}]\}$, commonly referred as bounded, fixed, unbounded delay models, respectively.

In this paper only $D^{max}(C, \mathcal{M}_g, \mathcal{I}), \mathcal{I} \in \{2, \omega^-\}$ is of interest in computing the delay of a circuit; hence, it will be referred to simply as the delay of circuit, and written as $D(\mathcal{M}_g, \mathcal{I})$, if the circuit in discussion is understood.

3 Exact Delay Computation: Functional and Topological Feasibility

Computing the exact delays of a circuit consists of two basic steps. First, choose input vectors to sensitize the longest sensitizable path. This step deals with the logic functionality of the circuit. Second, choose gate delay assignments within bounds so that the sensitization is realizable and path lengths are maximized if realizable. This step deals with topological constraints of the circuit. The above two steps are iterated until the longest sensitizable and realizable path(s) are found.

Previous methods for computing delays involve the first step only and assume all sensitizable paths are realizable. Thus, only upper bounds on delays are provided.

The following example illustrates the two basic steps for computing the exact 2-vector delay.

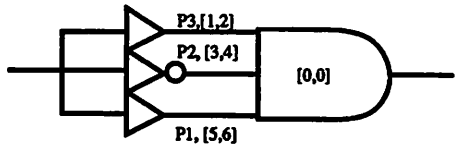

Figure 1: Example for Delay Computation

Example 1 *Referring to Figure 1, the delays of the gates are shown in the intervals. To find the delay for a falling input transition, we start with P_1. If P_1 propagates the last falling output transition, then the values of the side inputs at the AND gate should be non-controlling, implying the sensitization condition that at the time the buffer on P_1 switches the buffer on P_3 must still maintain the first input vector, which is* 1*, while the inverter maintains the second input vector so that the inverter's output is* 1*. Thus, $|P_3| > |P_1|$ and $|P_2| < |P_1|$. Now, we need to check whether the sensitization is realizable, namely, whether $|P_3| > |P_1|$ and $|P_2| < |P_1|$ are feasible. It is easy to see that $|P_3| > |P_1|$ and $|P_2| < |P_1|$ are not feasible for the bounds given, thus, P_1 is not the longest sensitizable and realizable path for falling transition. This process continues with other paths until the longest sensitizable and realizable path is found.*

In order to deal with the complexity of exact delay computation, a systematic and implicit method should be devised to manage functional and topological computations. To achieve this, we introduce a new formulation machinism, Timed Boolean Functions (TBF's), to characterize circuits.

4 Timed Boolean Function

Definition 2 *1. A waveform space W is a collection of mappings f: $R \mapsto \{0,1\}$.*

2. A Timed Boolean Function (TBF) is defined recursively as follows.

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

Example 2 *Let $x, y \in W$ be the waveforms shown in Figure 2(a) and 2(b); then the TBF $f(a,b)(t) = \overline{a}(t-1) \oplus b(t+1)$ represents the waveform shown in Figure 2(c) if a=x, b=y.*

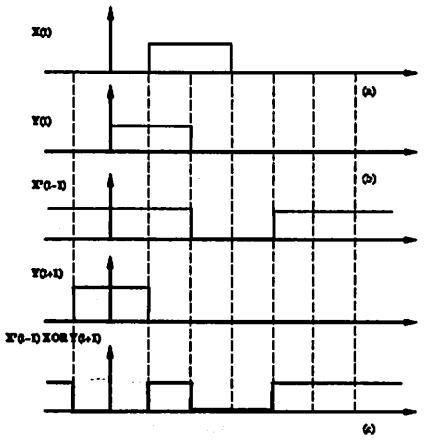

Figure 2: Representing Waveforms by TBF

4.1 Modeling Timing Behavior with TBFs

Before representing a circuit by a TBF, each component of the circuit needs to be modeled by a TBF.

Here, we only illustrate through examples the modeling process for some commonly encountered gates.

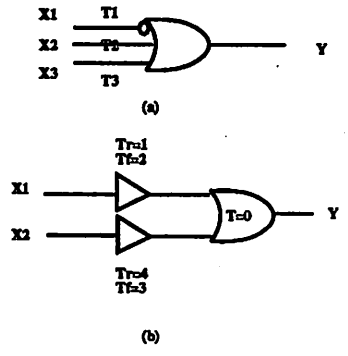

Figure 3: Modeling With TBF

1. Gates characterized by a single delay for each input-output pair. The complex gate shown in Figure 3(a) has three inputs; input x_i has a delay τ_i to the output. This gate is modeled with the TBF:

$$y(t) = \overline{x}_1(t-\tau_1) + x_2(t-\tau_2) + x_3(t-\tau_3).$$

2. Buffers 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 (falling) delay is the delay when the output is rising (falling). Each input is modeled by a buffer with different rising and falling delays; 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 3(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

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

and input 2 is

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

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 pulse shrinkage or dilation. Pulse shrinkage (dilation) effects occur 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 effect is captured.

Once each gate of a circuit is modeled, the TBF for each node in terms of primary inputs can be obtained by composing the gate model's TBF with the TBF's of the fanins of that gate. When the TBF of a circuit output is found, the circuit's behavior at any time can be calculated from the TBF. Suppose $f(t, x_1, ..., x_n, d_1, ..., d_m)$ is the TBF of an output of a circuit, where $x_1, ..., x_n$ are the primary inputs, and $d_1, ..., d_m$ are the delays of gates in the circuit. The value of the output at $t = k$ is given by $f(k, x_1, ..., x_n, d_1, ..., d_m)$.

5 Formulation for Exact Delay Computation

With TBF's, exact delay computation can be formulated nicely as a mixed Boolean linear programming problem. The delay of a circuit is the maximum arrival time of the last output transition over all gate delay assignments and a family of input vectors. Mathematically, let $f(t, x_1, ..., x_n, d_1, ..., d_m)$ be a TBF of the circuit, where $x_1, ..., x_n$ are the primary inputs, d_i is the ith gate's delay variable whose range is $[d_i^{min}, d_i^{max}]$. At $t = \infty$, all nodes in the circuit have settled down to their steady state values; hence the function $f(\infty, x_1, ..., x_n, d_1, ..., d_m)$ is the steady state function of the circuit,

and is equal to the static logical function. If the last output transition of a circuit occurs at $t = v$, then the output value of the circuit at $t = v^+$ is equal to the value of the circuit's static logical function, and the output value at $t = v^-$ is not equal to this, but is equal to $f(v^-, x_1, ..., x_n, d_1, ..., d_m)$. Therefore, the time of the last output transition is the maximum t such that:

$$f(t, x_1, ..., x_n, d_1, ..., d_m) \neq f(\infty, x_1, ..., x_n, d_1, ..., d_m)$$

With bounds on gate delays, the exact delay computation of a circuit can be formulated as a mixed Boolean linear programming problem as follows.

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

$$\begin{array}{rcl} f(t, x_1, ..., x_n, d_1, ..., d_m) & \neq & f(\infty, x_1, ..., x_n, d_1, ..., d_m) \\ d_i^{min} \leq & d_i & \leq d_i^{max} \end{array}$$

The above formulation applies for both 2-vector delay as well as delay by sequences of vectors. The semantics of this mixed Boolean linear programming is illustrated by the following example.

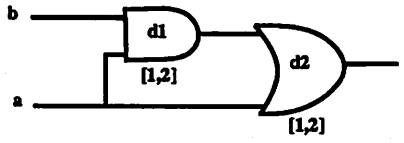

Figure 4: Example for Mixed Boolean Linear Programming

Example 3 *Refer to Figure 4. Assume the input signals are a pair of vectors switching simultaneously at* $t = 0$. *Let* $(a(0^-), b(0^-))$ *be the vector applied from* $-\infty$ *to* 0, (a, b), *the vector applied at* $t = 0$. *The TBF of the circuit is:*

$$f(t, a, b, d_1, d_2) = a(t - d_2) + a(t - d_1 - d_2)b(t - d_1 - d_2)$$

The static logical function is:

$$f(\infty, a, b, d_1, d_2) = a + ab = a$$

The ranges of the gate delay variables are $[1, 2]$. *The TBF variables can take on either the input vector applied before time 0 or after time 0. For example, TBF variable* $a(t - d_2)$ *equals to* $a(0^-)$ *if* $t < d_2$ *or equals to* a *if* $t > d_2$. *Any assignment of TBF variables to their respective before or after values* $x(0^-)$ *or* x *induces an associated linear programming problem. For instance, the Boolean assignment to the TBF variables:*

$$\begin{array}{rcl} a(t - d_2) & = & a \\ a(t - d_1 - d_2) & = & a(0^-) \\ b(t - d_1 - d_2) & = & b(0^-) \end{array}$$

induces following linear programming problem:

$$\max t$$

$$\begin{aligned} t - d_2 &> 0 \\ t - d_1 - d_2 &< 0 \\ 1 \leq d_1 &\leq 2 \\ 1 \leq d_2 &\leq 2 \end{aligned}$$

The maximum is $t = 4$. For this Boolean assignment, the TBF becomes

$$a + a(0^-)b(0^-)$$

which is not equal to the static function a. Since the longest topological path length is also 4, the maximum t from the linear programming is indeed the delay of the circuit. Therefore, finding the delay of a circuit is equivalent to finding a Boolean assignment to TBF variables such the induced linear program gives the maximum t and the TBF function evaluated at the assignment is not equal to the static logical function.

6 Solving Mixed Boolean Linear Programming for 2-vector Delay

We assume the pair of vectors $(x(0^-), x(0^+))$, (or $(x(0^-), x)$ for short) is applied to input x at $t = 0$.

Notation: The term $\sum_i d_j$ is the sum of gate delays along the ith path. We denote $\sum_i d_j$ by k_i, called a time constant, and $\sum_i d_j^{max}$ by k_i^{max}, and $\sum_i d_j^{min}$ by k_i^{min}. The longest topological length is denoted by L.

The idea in solving the mixed Boolean linear programming problem is to vary t from L. At each point t, the functional inequality $f(t, x_1, ..., x_n, d_1, ..., d_m) \neq f(\infty, x_1, ..., x_n, d_1, ..., d_m)$ is checked. If this functional inequality holds and the associated linear program is feasible, the delay of the circuit is lower bounded by the solution of the linear program. Otherwise, t is decreased further. The details of this search are discussed below.

6.1 Valuation of TBF variables

The argument of a TBF variable is a function of t and some d_i's. For instance, the argument of the TBF variable $x(t-k)$ is $t-k$. In this paper, the arguments of all TBF variables are of this form. $k = \sum d_i$, is the sum of gate delays along a path, and has minimum and maximum values of $k^{min} = \sum d_i^{min}$ and $k^{max} = \sum d_i^{max}$, respectively. For a specific value of t, we call the valuation of $x(t-k)$ positive if $t > k^{max}$ because $x(t-k)$ evaluates to $x(0^+)$, or simply x. Similarly, the valuation is negative If $t < k^{min}$ because $x(t-k)$ evaluates to $x(0^-)$. If $k^{min} < t < k^{max}$, the valuation can be either $x(0^+)$ or $x(0^-)$, depending on delay assignments to d_i. In this case, we call the valuation, delay dependent.

6.2 Searching for the maximum t

To find the maximum t of the mixed Boolean linear programming, we divide the search domain $[0, L]$ into intervals with the points $\{k_i^{max}\}$. Consider an interval $[a, b] = [k_i^{max}, k_j^{max}]$ containing no k_q^{max} but possibly some k_q^{min}. For the TBF variable $x(t - k_q)$ and such an interval $[a, b]$, only the following three cases can happen: 1) $k_q^{min} > b$, 2) $k_q^{max} < a$, 3) $k_q^{min} \in [a, b]$. For case 1), $x(t - k_q)$ evaluates to $x(0^-)$, or negative, whenever $t \in [a, b]$. For case 2), $x(t - k_q)$ evaluates to x, or positive, whenever $t \in [a, b]$. For case 3), valuation of $x(t - k_q)$ is delay dependent at $t = b^-$. Therefore, all the possible TBF valuations for $t \in [a, b]$ are included in the TBF valuations at $t = b^-$. Hence, the TBF of a circuit is equal to its static function for $t \in [a, b]$ if and only if the TBF is equal to its static function at $t = b^-$. That is, instead of examining all the points in any such interval not containing any k_q^{max}, we only need to examine the end of the interval at b^-.

At the initial search value $t = L$, the valuations of all TBF variables are positive; hence, the TBF function is equal to its static logical function. Then t is decreased into the next search interval. To compare the TBF of the circuit to its static function, the phase (positive or negative) of a delay dependent valuation must be resolved. Each resolution induces a linear constraint; for instance, choosing a positive phase induces the inequality $t > \sum d_i$. Naively, each resolution of any delay dependent valuations should be tried, and the resolved TBF compared with its static function. If they are unequal and the induced constraints are feasible, the maximum t of the associated linear programming problem (using the induced constraints and the delay bounds) for this instance of resolution is a candidate for the delay of the circuit. The delay of the circuit is the maximum of the candidate delays of all the resolutions. In a later section, a technique using BDD's to implicitly enumerate the resolutions is presented. If, for every resolution, either the TBF's of the circuit is equal to its static function or the resolution is not feasible, then the last output transition of the circuit can not happen in the interval. Therefore, t is decreased to the next interval, and the equality and feasibility checking processes are repeated.

7 Computational Issues

Here, we discuss efficient techniques to represent TBF's by multi-level networks and to enumerate implicitly TBF resolutions using BDD's.

7.1 TBF Networks

Explicitly representing TBF's results in substantial memory usage. Thus, a TBF for a circuit is represented by a multi-level network constructed from the original network of the circuit. The original network can be regarded as a representation of a TBF with TBF variables of only positive valuation, i.e. TBF evaluated at $t = L$. A TBF network at an arbitrary t is just a generalization of this view. In a TBF network, paths are partitioned into three groups, the positive group consisting of paths whose maximum lengths are less than t, the negative group whose minimum lengths are greater than t, and the delay dependent group where t lies between the paths' maxima and minima. Recalling the relationship between paths and TBF variables, the positive group of paths correspond to the TBF variables with positive valuations. Analogously for negative and delay dependent paths.

In the negative group, a path that was connected to primary input x is now connected to a new primary input $x(0^-)$. Similarly in the delay dependent group, a path that was connected to primary input x is now connected to a new primary input x^*. Primary inputs for paths in the positive group remain the same.

The exact construction of the TBF network from the original one is similar to that of an independent work in [MSS+91]. In general, gates are duplicated to separate paths through the circuit into three groups corresponding to positive, negative, and delay dependent valuations.

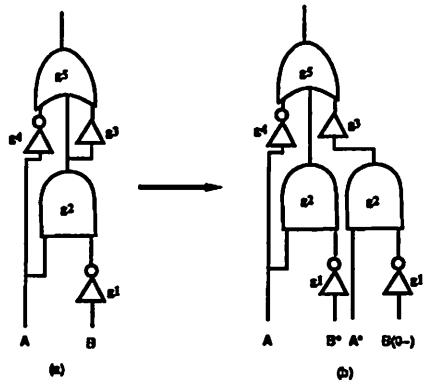

Figure 5: TBF Network

Example 4 *Refer to Figure 5. Figure 5(a) is the original network. Each gate's maximum delay is 1 and minimum delay is 0.9. At $t = 2.8$, path $\{g1 - g2 - g3 - g5\}$ is negative, because its minimum length is 3.6. Paths $\{g1 - g2 - g5, g2 - g3 - g5\}$ are delay dependent, because t lies between their respective minima and maxima. The other paths are positive. The TBF network at $t = 2.8$ is shown in Figure 5(b). It is easy to check that the network in Figure 5(b) represents the TBF of the original network at $t = 2.8$.*

Once a TBF network is constructed to represent a TBF of a circuit at t, BDD for the TBF network is compared with the BDD of the original circuit. However, before the BDD for the TBF network can be constructed, the delay dependent inputs, e.g. A^* and B^* in example 5, must be resolved.

7.2 Delay Dependence Resolution

For a TBF with n delay dependent valuations, there are 2^n instances resolutions; For instance, the TBF $f(t, x(0^+), x(0^-), x^*)$ has delay dependent valuation x^*, and the two resolutions are $x^* = x(0^+)$ and $x^* = x(0^-)$. Hence, for large n, it is infeasible to try each resolution independently. Here, we present a technique of using BDD's to implicitly enumerate all resolutions. The idea is to modify the TBF's so that the new TBF's select the resolutions automatically. This is achieved by replacing each delay dependent valuation, say x^*, with $s_x x(0^+) + \overline{s}_x x(0^-)$. Then, if the resolvent s_x is 1, the resolution $x(0^+)$ is selected, otherwise, $x(0^-)$ is selected. With this replacement, the example TBF

becomes

$$f(t, x(0^+), x(0^-), x^*) = f(t, x(0^+), x(0^-), s_x x(0^+) + \overline{s}_x x(0^-))$$

This modified TBF is then compared with its static function by comparing their BDD's. If they are not equal, the exclusive-OR of their BDD's is computed. Now for each **cube** of the XOR BDD, the feasibility of the cubes of resolution variables are checked. If a resolvent s_x has a literal of 1 in the cube, this means the positive resolution, i.e. $x(0^+)$, was selected, hence it induces the constraint $t - \sum d_i > 0$. A literal of 0 induces $t - \sum d_i < 0$. A literal of 2 means either positive or negative resolution can be selected, i.e $t - \sum d_i$ is either > 0 or < 0, hence, no constraint is induced. From experiments with ISCAS benchmarks, this enumeration technique proves to be very efficient.

7.3 Algorithm for Computing Exact 2-vector Delays

Let $f(t)$ be a TBF of a circuit, K_i's, the time constants. The 2-vector delay of the circuit can be found as follows.

```
        Sort K_i^max in descending order.
Delay_found=0;
while(!Delay_found){
        t=next K_i^max;
        Evaluate TBF f(t);
        for each TBF variable x(t-k){
                x(t-k) → x(0^-) if k^min > t;
                x(t-k) → x(0^+) if k^max < t;
                x(t-k) → s_x x(0^+) + s̄_x x(0^-) if k^max > t > k^min;
        }
        if (f(t) ≠ f(∞)){
                if(∃ cubes in BDD(f(t)) ⊕ BDD(f(∞)) feasible)
                        Delay_found=1;
                        Delay= maximum of linear
                                programming over all feasible cubes.
                }
        }
}
```

The above algorithm for computing the exact 2-vector delay using the bounded gate delay model has been implemented and the results are shown in section 12.

8 Floating Delay and Delay by Sequences of Vectors

The viability and floating delays assume the unbounded gate delay model, thus, it is not known how theses delays will be affected by lower bounds on gate delays. In this section, we relate floating delay to delay by sequences of vectors and show that for most practical circuits, delay by sequences

of vectors, viability, and floating delay are the *exact* delays under both bounded and unbounded gate delay models.

The floating delay [CD90], as well as viability delay, is essentially path sensitization by a single vector with the condition that the logical values at the nodes in the circuit are assumed unknown until the input vector has propagated. Unknown node values are assumed because input vectors preceding the sensitization vector may cause the node values to be arbitrary.

The setting for delay by sequences of vectors is: each gate's delay may vary between a lower and an upper bound, the delay is the latest arrival time of the last output transition, when a sequence of an arbitrary number of vectors with arbitrary intervals between vectors is applied to the inputs and the last vector is applied at $t = 0$.

Arbitrary input sequences can actually happen in practical situations but whether nodes in a circuit can take on arbitrary values is not obvious. A natural question is then: is floating delay too conservative for practical circuits, or how do delay by sequences of vectors and floating delay compare?

To compare the floating delay with the delay by sequences of vectors, we observe that floating delay is always greater or equal to delay by sequences of vectors, because of the assumption of arbitrary values at nodes. Therefore, if for each path sensitizable under the floating delay model there exists a sequence of vectors such that the path propagates the last output transition, then floating delay is equal to delay by sequences of vectors. Whether floating delay is equal to delay by sequences of vectors depends on the gate delay models involved, because there exist circuits with gates of fixed delays such that their delays by sequences of vectors are less than their floating delays. An example is shown below.

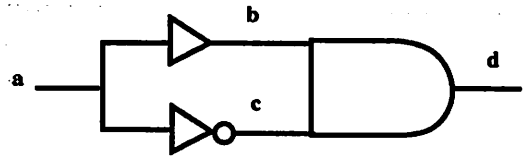

Figure 6: Circuit Illustrating Floating Delay not Equal to Delay by Sequences of Vectors

Example 5 *Referring to Figure 6, the delay for each gate is a fixed constant 1. Node b and node c always have opposite values; hence, the delay by sequences of vectors is* 0. *However, the floating delay is* 2. *Practical gates may not have the identical delays; they may differ by a small amount. Then, the output of the circuit in Figure 6 can only be glitches of width equal to the delay difference between the inverter and the buffer, assuming zero inertial delays on all gates. But practical gates on the other hand have finite inertial delays; so the narrow glitches will not appear at the output. Thus, small variation in gate delays may be rendered as fixed constants by finite inertial delays. Therefore, for circuits with well controlled delays, floating delays may be larger than delays by sequences of vectors. However, in this paper, we will not discuss the effect of inertial delays.*

Since practical circuits do not have fixed gate delays due to variations in the manufacturing process, then, will variable gate delays constitute equivalence of floating delay and delay by sequences of vectors? To resolve this, we examine whether a sensitizable path under the floating

delay model propagates the last input transition under some sequence of vectors. If such a path propagates the last transition, the side inputs on each gate along the path must be initialized by the sequence of vectors such that at the moment the transition arrives a gate all its side inputs take on non-controlling values.

Without loss of generality, we assume the last vectors of all sequences of vectors are applied at $t = 0$ in the following discussion.

Definition 3 *1. Let g be a gate on a given path π, π_g the partial path of π that ends at g, and π_s a side path of g. The side path relative arrival time at g for π_s is $|\pi_g| - |\pi_s|$, where $|\pi|$ is the length of path π.*

2. A gate in a circuit is said to propagate a transition from a fanin if a transition at the fanin causes a transition at the gate's output and there is no transition at other fanins at the time of the transition at the fanin.

3. A path is said to propagate a transition if a transition can propagate through all the gates on the path.

Theorem 1 *Let π be a sensitizable path in the floating delay model. If all relative arrival times for side paths of π from the same primary input are different, i.e. the side paths originating from the same primary input and reconverging to a gate on π have different relative arrival times, then there exists an input sequence such that π propagates the last output transition.*

Proof. See Appendix.

The condition on side path relative arrival times in the above Theorem can be guaranteed by the following topological condition on a circuit: every gate in the circuit has variable delay, i.e. $d^{min} \neq d^{max}$, and no two distinct paths have the same set of gates, (this condition can always be alleviated by adding a buffer of variable delay at a lead). If this condition is met, gate delays can be chosen to meet the requirement of side path relative arrival times. This fact is stated in the following Lemma.

Lemma 1 *If every gate in a circuit has variable delay and no two distinct paths have the same set of gates, then for any path π there exists a gate delay assignment d_i^*, where $d_i^* \in [d_i^{max} - \epsilon, d_i^{max}]$ for infinitesimal ϵ, such that all relative arrival times for side paths of π from the same primary input are different.*

Proof. See Appendix.

Therefore from Theorem 1 and Lemma 1, we know that for most practical circuits, floating delay is equal to delay by sequences of vectors:

Theorem 2 *If every gate in a circuit has variable delay and no two distinct paths having the same set of gates, then, floating delay $= D([d^{min}, d^{max}], \omega^-)$.*

Proof. See Appendix.

If gates with fixed delays are allowed, then $D([d^{min}, d^{max}], \omega^-) \leq$ floating delay.

9 Exact Delay by Sequences of Vectors

We have shown that delays by sequences of vectors, and floating (or viability) delays give the exact delays if a circuit has variable gate delays and no two distinct paths consist of the same set of gates. There are efficient algorithms computing the viability delays [MSS+91] and floating delays [DKM92]. Even though, we present an algorithm to compute the delay by sequences of vectors to illustrate the use of TBF's and that the computations of the 2-vector delay and delay by sequences of vectors can be done under the *same* framework with TBF's.

To compute the exact delay by sequences of vectors, we examine how the algorithm for 2-vector computation can be modified. Since there is no restriction on the sequence of vectors before $t = 0$, there can be a vector at any time before time 0. Thus, the only difference in computing the delay by sequences of vectors and the delay by a pair of vectors is when $t < \sum d_i^{max}$; then $x(t - \sum d_i)$ evaluates to the input vector at time $t - \sum d_i$ instead of delay dependent or negative.

The following discussion on computing the exact delay by sequences of vectors is organized as follows. Recall that there are two steps in delay computation: finding a feasible delay assignment (topological constraint) and checking whether this delay assignment sensitizes the longest sensitizable path (functional constraint). We first give a feasible delay assignment, then a search algorithm under this delay assignment, and show that once the search algorithm detects an output transition, i.e. the TBF is not equal to its static function, the delay assignment produces the delay of the circuit.

9.1 Satisfying Topological Constraints

The exact delay by sequences of vectors of a circuit is the worst delay over all possible delay assignments. We will show that the following delay assignment is the worst assignment. Consider the delay assignment such that $d_i^{max} - d_i < \delta$, where δ is an infinitesimally small number and that for each primary input x all k_i's in $\{k_i : \exists x(t-k_i)\}$ are different. In other words, for this assignment, all paths starting at primary input x have different lengths but within ϵ of their maximum lengths, where ϵ is some fixed multiple of δ. The existence of such a delay assignment is warranted by Lemma 3 in the Appendix.

9.2 Searching Algorithm for Delay

Again we divide the search domain $[0, L]$ into intervals separated by the points $\{k_i^{max}\}$. Under the delay assignment of section 9.1 , all k_i's are within ϵ of k_i^{max}'s which are upper bounds of some intervals. Let $f(t)$ be the TBF of the circuit and $f(\infty)$ its static function.

For such an interval $[a, b]$, we want to determine whether $f(t) \neq f(\infty)$ for $t \in [a, b]$. As t varies from b to a, TBF variables in $f(t)$ evaluate accordingly. For instance, TBF variable $x(t - k_i) = x(0^+)$, if $t > k_i$, and $x(t - k_i)$ becomes a new Boolean variable representing the input vector at time t if $t < k_i$. In order to have $f(t) \neq f(\infty)$ for $t \in [a, b]$, the $f(t)$ with the greatest flexibility should be chosen. This is the one where t creates the most new Boolean variables. Since all $k_i \in [a, b]$ are within ϵ of b by this particular delay assignment, i.e. $b \geq k_i \geq b - \epsilon$ for all $k_i \in [a, b]$, then at $t = b - \epsilon$ all TBF variables with $k_i \in [a, b]$ become new Boolean variables; thus, all these

TBF variables with the same primary input become distinct new Boolean variables, because all their k_i's are numerically different under the delay assignment. In summary, at $t = b - \epsilon$, all TBF variables with $k_i \in [a, b]$ become distinct new Boolean variables; hence, this delay assignment is a worst case delay assignment. Therefore, for interval $[a, b]$ only the point $t = b - \epsilon$ needs to be examined.

At each t, we check $f(t) \neq f(\infty)$. Once this holds in an interval, $[a, b]$, t is maximized to obtain an upper bound of delay. However, the value of t is only ϵ less than the maximum possible value and ϵ can be arbitrarily small. Therefore, the maximum t is b.

9.3 Satisfying Functional Constraints

To show that b is the exact delay, we demonstrate an input sequence that indeed invokes an output transition at $b - \epsilon$. Compute $f(b - \epsilon) \oplus f(\infty)$ and let v be a Boolean vertex of $f(b - \epsilon) \oplus f(\infty)$. Construct the input sequence to the primary input x as follows. For each TBF variable $x(t - k_i)$, create a pulse of arbitrarily narrow width centered at $b - \epsilon - k_i$ and of value equal to the literal of the TBF variable in v. Apply this input sequence to the circuit. It is easy to see that the output of the circuit at $t = b - \epsilon$ is not equal to its static value, because the output of the circuit at $t = b - \epsilon$ is $f(b - \epsilon)$, which is not equal to $f(\infty)$. Therefore, there is an output transition at $t = b - \epsilon$. Since $b - \epsilon$ is the first time it invokes an output transition, it is the delay.

It is interesting to see that the above discussion holds for any values of lower bounds on gate delays; therefore, the delay by sequences of vectors is the same for all lower bounds not equal to upper bounds. Formally,

Theorem 3 *If in circuit C there are no two distinct paths consisting of the same set of gates, then*

$$D(C,[d^{min}, d^{max}], \omega^-) = D(C,[0, d^{max}], \omega^-), \forall\, d_i^{min} < d_i^{max}.$$

Unlike delays by sequences of vectors which may be smaller for circuits with gates of fixed delays (see Example 5), floating delays are the same whether the circuits have fixed or variable gate delays.

Theorem 4 *The floating delay (which is the same as the viability delay [MB89]) is invariant under fixed (worst case), unbounded, and bounded gate delay models*

Proof. See Appendix.

9.4 Algorithm for Computing Exact Delays by Sequences of Vectors

Assume every gate in a circuit has variable delay and no two distinct paths having the same set of gates. Let $f(t)$ be a TBF of a circuit, K_i's, time constants. The delay by sequences of vectors for this circuit can be found as follows.

Sort K_i^{max} in descending order.
do{
t=next K_i^{max};

Evaluate TBF $f(t)$;
for each TBF variable $x(t-k)$,{
$x(t-k) \rightarrow x(0^+)$, if $k^{max} < t$;
$x(t-k) \rightarrow$ new Boolean variable Y_x, if $k^{max} > t$;
}
}while($f(t) = f(\infty)$)
Delay=t.

10 Effects of Gate Delay Lower Bounds

As seen in the previous section, lower bounds on gate delays have no effect on floating delays and delay by sequences of vectors. When will lower bounds of gate delays have an effect on 2-vector delays? As stated below, whether lower bounds of gate delays impact 2-vector delays depends on the relative magnitudes of the 2-vector delays and lower bounds.

Theorem 5 *In a circuit, if all lower bounds of all paths' lengths are less than the circuit's 2-vector delay, then further decreasing the lower bounds of the gate delays of the circuit will not speed up the circuit. Symbolically, for a circuit C with gate delay model $[d_i^{min}, d_i^{max}]$, if $max(k_i^{min}) < D(C,[d_i^{min}, d_i^{max}],2)$, then $D(C,[d_i^{min}, d_i^{max}],2) = D(C,[x_i^{min}, d_i^{max}],2), \forall\, x_i^{min} < d_i^{min}$.*

Proof. See Appendix.

If $max(k_i^{min}) > D(C,[d_i^{min}, d_i^{max}],2)$, then,

$$D(C,[d_i^{min}, d_i^{max}],2) \leq D(C,[x^{min}, d_i^{max}],2), \forall\, x_i^{min} < d_i^{min}$$

It is common that gates' lower bounds are some percentages of their upper bounds. In this case, it is easy see whether precision of a manufacturing process has an impact on the 2-vector delays of circuits. Let $d^{min} = f \cdot d^{max}$. Then, the range of f in which lower bounds have no effect on 2-vector delay is:

$$f < \frac{D(C,[0,d_i^{max}],2)}{L}$$

where $D(C,[0,d_i^{max}],2)$ is the 2-vector delay on unbounded gate delay model, and L is the longest topological path length. The above inequality has the interpretation that if a manufacturing process does not have the precision to achieve the threshold value, $f < \frac{D(C,[0,d_i^{max}],2)}{L}$, then a less precise manufacturing process will fabricate circuits with the same 2-vector delays. In addition, since $D(C,[0,d_i^{max}],2)$ is only valid for estimating cycle time [LBSV92b] when $> \frac{L}{2}$, $f < 0.5$ is not worthwhile.

11 An Example: 4-bit Ripple Bypass Adder

In this example, we find the carry output delay (2-vector) of a 4-bit ripple bypass adder, and illustrate how TBF's can be expressed implicitly with circuits. A 4-bit ripple bypass adder is

shown in Figure 7. The ranges of gate delays are shown in parenthesis next to the gates. Gate g_0 models the delay from the previous stage. The sum bits are ignored.

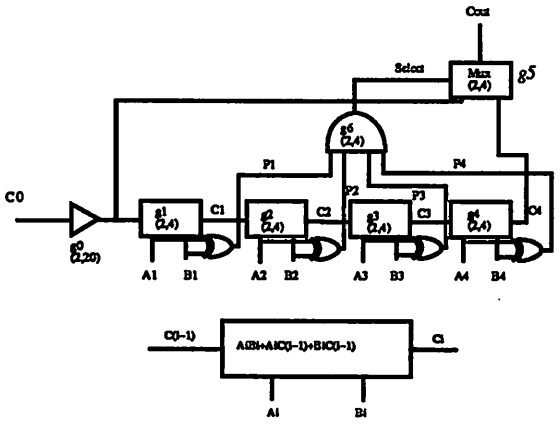

Figure 7: 4-bit Ripple Bypass Adder

1. The longest topological path length is 40, contributed by the gates $g_0, g_1, g_2, g_3, g_4, g_5$. We divide the search domain $[0, 40]$ into intervals by the points $\{k_i^{max}\}$. The k_i^{max}'s are 40, 24, 20, ... Thus, the first two intervals are $[24, 40]$ and $[20, 24]$.

2. In the first interval $[24, 40]$, we evaluate the TBF at $t = 40^-$. From the circuit, only the path $g_0 - g_1 - g_2 - g_3 - g_4 - g_5$ is evaluated to be delay dependent; all other paths have positive valuations. Thus, the input of the path $g_0 - g_1 - g_2 - g_3 - g_4 - g_5$ that was connected to c_0 is now connected to c_0^*, which denotes delay dependent valuation. The TBF network at $t = 40^-$ is shown in Figure8.

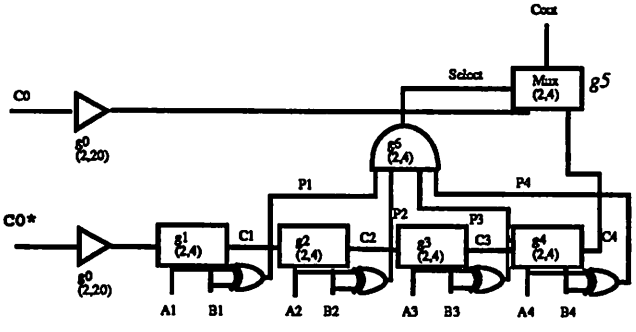

Figure 8: TBF Networks for 4-bit Carry Byass Adder, $t = 40^-$

BDD's are built for the TBF network and the original circuit. The result is that both circuits have the same functionality. Thus, the search moves to the next interval $[20, 24]$.

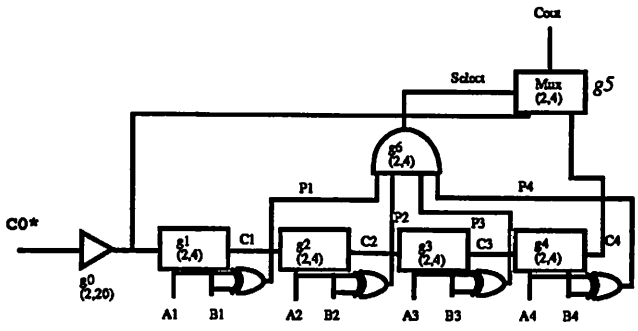

Figure 9: TBF Networks for 4-bit Carry Byass Adder, $t = 24^-$

3. For $t = 24^-$, both paths $g_0 - g_1 - g_2 - g_3 - g_4 - g_5$ and $g_0 - g_5$ evaluate to be delay dependent, while all others evaluate positive. The TBF network for $t = 24^-$ is shown in Figure9. By comparing the BDD's for this TBF network and the original network, this TBF network is not equal to the original circuit. Then the XOR BDD is computed; a cube of this XOR BDD is $c_0 = 1, c_0(0^-) = 0$, and $A_i \oplus B_i = 1$. The induced linear programming problem is:

$$\begin{array}{rcl} & \max t & \\ t & < & g_0 + g_5 \\ t & < & g_0 + g_1 + g_2 + g_3 + g_4 + g_5 \\ 2 \leq & g_0 & \leq 20 \\ 2 \leq & g_i & \leq 4, i = 1, ..., 5 \end{array}$$

The maximum value is 24, which is also the upper bound of the search interval. Therefore the 2-vector delay of this 4-bit carry bypass adder is 24.

12 Experimental Results

We implemented the algorithms for computing the exact 2-vector delays and delays by a sequence of vectors. The implemented codes can use as upper bounds results from other heuristic delay computation programs, and start computing exact delays from there. The experimental data shown below did not use any heuristic upper bound program as a front-end. In the final version of this paper, we will use the efficient program in [MSS+91] to find delay bounds and then proceed from the bounds to exact delays. With this addition, we expect the CPU times to improve by a factor of about 10 to 100.

The following ISCAS benchmarks were run on a DECstation 5000 (38 mips) with a standard sis script and then mapped to the mcnc library. The actual delay values used are the ones given in the library. The minimum gate delay is assumed to be 90% of the maximum gate delay.

Circuit	Inputs	Outputs	Approx. Gates	Top. length	Exact 2-vector Delay	CPU time (sec)
C432	36	7	160	39	37.3	2868.28
C499	41	32	202	28.9	28.9	231.85
C880	60	26	383	32.8	32.8	4.45
C1355	41	32	546	28.9	28.9	423.76
C1908	33	25	880	41.1	27.2	12140†
C2670	233	140	1193	36.5	36.5	2.69
C3540	50	22	1669	53.1	51.2	1247.31
C5315	178	123	2307	51.6	45.9	717.43
C6288	32	32	2406	140	fail to build bdd.	
C7552	207	108	3512	59.8	59.8	4.04

†Upper bound, out of memory in building XOR BDD; exact delay will be given in the final version.

Circuit	Inputs	Outputs	Approx. Gates	Top. length	Exact Delay by seq. vec.	CPU time (sec)
C432	36	7	160	39.0	37.3	1815.6
C499	41	32	202	28.9	28.9	191.6
C880	60	26	383	32.8	32.8	3.73
C1355	41	32	546	28.9	28.9	376
C1908	33	25	880	41.1	27.2	12140
C2670	233	140	1193	36.5	36.5	1.84
C3540	50	22	1669	53.1	51.2	592.8
C5315	178	123	2307	51.6	45.9	455.6
C6288	32	32	2406	140	memory out	
C7552	207	108	3512	59.8	59.8	2.92

As can be seen the delays by sequences of vectors or floating delays and delays by a pair of vectors are the same for all the ISCAS benchmarks.

13 Conclusion

In this paper, we introduced a new circuit delay model, delay by sequences of vectors. which captures the essence of viability and floating delays. Then, we classified delays of circuits according to both the delay models of the gates making up the circuits *and* the family of inputs to the circuits. In this classification, we gave sufficient conditions under which floating delay is the same as delay by sequences of vectors; these sufficient conditions are true for most practical circuits. This implies that the assumption of arbitrary node values used in the floating delay model is not too conservative. Thus, delay by sequences of vectors (hence, viability and floating delays), transition delay, and cycle time delay have coherent definitions under the *same* framework.

Next, we studied the problem of computing the exact circuit delays under both bounded and unbounded gate delay models for which previous research gives only upper bounds.

By using a new formulation technique, called Timed Boolean Function, we formulated the problem of computing the exact delays as a mixed Boolean linear programming problem for which we gave efficient algorithms to compute the *exact* delays of combinational circuits for transition delay and delay by sequences of vectors.

The algorithms consider a subset of paths at one time and only the paths potentially responsible for the delay of a circuit are considered. Moreover, the core computation of the algorithms are composed of two computationally efficient algorithms: linear programming and BDD manipulations.

We next showed that delays by sequences of vectors, as well as floating delays and viability delays are invariant under both bounded and unbounded gate delay models. Finally, we addressed the effect of gate lower bounds on delays of circuits. We demonstrated the effectiveness of the method by giving exact delay results for all ISCAS benchmark circuits (except C6188).

References

[CD90] H. C. Chen and D. H. Du. Path sensitization in critical path problem. *1990 ACM Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 1990.

[DKM91] S. Devadas, K. Keutzer, and S. Malik. Delay computation in combinational logic circuits: Theory and algorithms. *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 1991.

[DKM92] S. Devadas, K. Keutzer, and S. Malik. Certified timing verification and transition delay of a logic circuit. *Proc. of the 29th Design Automation Conference*, June, 1992.

[HPS91] S. Huang, T. Parng, and J. Shyu. A new approach to solving false path problem in timing analysis. *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 1991.

[LBSV92a] W. Lam, R. Brayton, and A. Sangiovanni-Vincentelli. Exact delay computation with timed boolean function. *UC Berkeley ERL memorandum: UCB/ERL M92/57*, May 1992.

[LBSV92b] W. Lam, R. Brayton, and A. Sangiovanni-Vincentelli. Minimum cycle time of synchronous circuit with bounded delays. *UC Berkeley ERL memorandum: UCB/ERL M92/56*, May 1992.

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

[MSS+91] P. McGeer, A. Saldanha, P. Stephan, R. Brayton, and A. Sangiovanni-Vince ntelli. Timing analysis and delay-fault test generation using path recursive functions. *IEEE International Conference on Computer-Aided Design*, pages 180–183, Nov. 1991.

14 Appendix

Proof for theorem 1. Because π is sensitizable in the floating delay model, there exists an input vector $v = (..., v_\pi, ...)$, where v_π is the primary input of π, such that for each gate on π the followings are true. 1) If all fanins have non-controlling values under v, π is the last to be of non-controlling value. 2) If there is a controlling value at a fanin of the gate under v, then π is the first to be of controlling value, [CD90].

Now construct an input sequence of vectors for π to propagate the last output transition as follows. Let the last vector applied at $t = 0$ be v, and the vector applied before $t = 0$ be $v' = (..., \overline{v_\pi}, ...)$. The remaining vectors are constructed so that as the input transition at π, $\overline{v_\pi} \rightarrow v_\pi$ arrives at a gate on π, all the side inputs take on non-controlling values. Consider gate g on π. If the value π at g is non-controlling, then all side inputs to g must have settled to non-controlling values, because in this case π is the last to be non-controlling under v. So, the transition propagates through g. If the value π at g is controlling, then the side inputs to g that have settled must settle

to be non-controlling, because in this case π is the first to be controlling. For those unsettled side inputs, construct the input sequence so that these side inputs take on non-controlling values at the time of transition at π_g, which is $t = |\pi_g|$. The functions of these unsettled side inputs are the TBF's at $t = |\pi_g|$. A TBF variable in these TBF's has the form $x(t - |\pi_s|)$, where π_s is a side path of g ending at the side input. At $t = |\pi_g|$, the value of $x(t - |\pi_s|)$ is $x(|\pi_g| - |\pi_s|)$, which is the value of input at $t = |\pi_g| - |\pi_s|$, which is the side path relative arrival time. Thus, if $|\pi_g| - |\pi_s| > 0$ the value of $x(t - |\pi_s|)$ is the value of v at primary input x, i.e. the TBF variable has settled; otherwise, it is the input value at $t = |\pi_g| - |\pi_s|$. Since these side inputs have not settled, the supports of their TBF's contain TBF variables that have not settled. Since all side paths from the same primary input have different relative arrival times, all unsettled TBF variables can be regarded as different Boolean variables. So, no two unsettled TBF variables are the same. Thus, we can find a Boolean assignment to the unsettled TBF variables such that the TBF's of the unsettled side inputs take on non-controlling values. If $x(t - |\pi_s|)$ has Boolean assignment 1, add a pulse of infinitesimal width centering at $t = |\pi_g| - |\pi_s|$ with the value of the pulse equal to 1. We add such a pulse for each TBF variable $x(t - |\pi_s|)$ and repeat for other primary inputs. Similarly for assignment of 0.

Now, suppose the constructed waveforms are input to the primary inputs; at $t = |\pi_g|$, $x(t - |\pi_s|)$ takes the value of its Boolean assignment; hence, the value of the side input is non-controlling at $t = |\pi_g|$. Therefore, when a transition arrives at g, all unsettled side inputs are non-controlling, permitting the transition to propagate through.

For other gates on π, we just add more pulses to the existing waveforms. This process can be done without creating conflicting pulses on the same primary input at the same point on the time axis, because relative arrival times for all side paths from the same primary input are different along π, namely all $|\pi_g| - |\pi_s|$ are different for all g on π and all π_s's from the same primary input.

Therefore, the input transition $\overline{v}_\pi \rightarrow v_\pi$ propagates along π. Since this path is the last to be non-controlling if all settled fanins are non-controlling and the first to be controlling if some fanins are controlling, this transition is the last transition at the output. ■

Proof for lemma 1. Consider $|\pi_g| - |\pi_s|$'s for the side paths from the same primary input, where $|\pi_g| = \sum d_i$ is the partial path length ending at gate g on π, and $|\pi_s| = \sum d_i$ is that of a side path at g. Finding the required delay assignment is equivalent to finding an assignment on d_i's such that all $|\pi_g| - |\pi_s|$'s are different.

We first show that no two $|\pi_g| - |\pi_s|$'s have the same set of d_i's, and then construct the required delay assignment. Consider two $|\pi_g| - |\pi_s|$'s at the same gate, namely, $|\pi_g| - |\pi_{s_1}|$ and $|\pi_g| - |\pi_{s_2}|$. They do not have the same set of d_i's because the two distinct paths π_{s_1} and π_{s_2} do not have the same set of gates, hence, d's. Consider two $|\pi_g| - |\pi_s|$'s at different gates on π, namely, $|\pi_{g_1}| - |\pi_{s_1}|$ and $|\pi_{g_2}| - |\pi_{s_2}|$. Let gate g_2 be after gate g_1. Then, any d_i between g_1 and g_2 is in π_{g_2} but not in π_{g_1} and π_{s_1}, because the side path π_{s_1} never extends beyond g_1. Since π_{s_2} is a side path of π_{g_2}, there is a d_i between g_1 and g_2 that is in π_{g_2} but not in π_{s_2}. Then, this d_i is in $|\pi_{g_2}| - |\pi_{s_2}|$ but not in $|\pi_{g_1}| - |\pi_{s_1}|$. Hence, they do not have the same set of d_i.

Denote $|\pi_g| - |\pi_s|$ by $\sum a \cdot d_i$, $a \in \{1, -1\}$. Then, by lemma 2, the desired assignment can be constructed. ■

Lemma 2 *If all $\sum a \cdot d_i$, where $a \in \{1, -1\}$ and $d_i \leq d_i^{max}$, have different sets of d_i's, then there exists an assignment d_i^* on d_i's with $d_i^* \in [d_i^{max} - \epsilon, d_i^{max}]$, ϵ is infinitesimal such that all $\sum a \cdot d_i$*

evaluate differently.

Proof. Construct an assignment on d_i's so that all $\sum a \cdot d_i$'s evaluate differently. This construction modifies an initial assignment iteratively; after each iteration, more $\sum a \cdot d_i$'s evaluate differently. This procedure stops when all $\sum a \cdot d_i$'s evaluate differently. Start with the initial assignment $d_i = d_i^{max}$ and assume there are at least two $\sum a \cdot d_i$'s evaluating to the same value; let them be s_1 and s_2. Since s_1 is not the same as s_2, there is a d_x that is in s_1 but not in s_2 (or vice versa). Let δ be the minimum of $|s_i - s_j|, \forall s_i \neq s_j$. Modify the delay assignment by decreasing d_x by an amount of $\frac{\min(\delta,\epsilon)}{2}$. It is easy to see that now s_1 and s_2 do not evaluate the same, and $\sum a \cdot d_i$'s that evaluated differently evaluate differently under this modified assignment, because each $\sum a \cdot d_i$ can decrease by no more than the minimum separation between neighboring $\sum a \cdot d_i$'s. Continue modifying the delay assignment until all $\sum a \cdot d_i$'s evaluate differently. ■

Lemma 3 *If every gate in a circuit has variable delay and no two distinct paths have the same set of gates, then there exists a gate delay assignment d^* with $d^* \in [d^{max} - \epsilon, d^{max}]$ ϵ is infinitesimal such that all paths from any primary input to all nodes have different lengths.*

Proof. Since no two distinct paths have the same set of gates, hence, d_i's, the assignment can be obtained by lemma 2. ■

Proof for theorem 2. Let π be a sensitizable path in the floating delay model in a circuit. Then the floating delay of this circuit is the maximum of $|\pi|$, i.e. when $d_i = d_i^{max}$. If the circuit meets the condition in theorem 2, lemma 1 guarantees a delay assignment such that the side path arrival time (with respect to π) requirement in theorem 1 is satisfied. Then, theorem 1 assures the existence of an input sequence that propagates the last output transition. Therefore the delay by sequences of vectors is equal to the length of π under this assignment. Since this delay assignment d^* can be made arbitrarily close to d^{max}, the length of π under this delay assignment is arbitrarily close to the maximum of $|\pi|$. ■

Proof for theorem 5. In the search for $D(C,[x_i^{min}, d_i^{max}], 2)$, the search region is $[D(C,[d_i^{min}, d_i^{max}], 2), L]$; hence, none of k_i^{min} is involved because all of k_i^{min} are less than $D(C,[d_i^{min}, d_i^{max}], 2)$. Therefore, further decreasing k_i^{min}, has no effect on the search process. ■

Proof for theorem 4. Let $D_f([d^{max}, d^{max}]), D_f([0, d^{max}]), D_f([d^{min}, d^{max}])$ denote the floating delay under fixed worst case, unbounded, and bounded delay models, respectively. It is easy to see that $D_f([d^{max}, d^{max}]) \leq D_f([d^{min}, d^{max}]) \leq D_f([0, d^{max}])$ because the set of circuits under a delay model is a subset of the set of circuits under the succeeding model. In [CD90], it is shown that $D_f([d^{max}, d^{max}]) = D_f([0, d^{max}])$. ■