MINIMUM CYCLE TIME OF SYNCHRONOUS CIRCUIT WITH BOUNDED DELAYS

by

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

Memorandum No. UCB/ERL M92/56

28 May 1992

MINIMUM CYCLE TIME OF SYNCHRONOUS CIRCUIT WITH BOUNDED DELAYS

by

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

Memorandum No. UCB/ERL M92/56

28 May 1992

TITLE PAGE

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

MINIMUM CYCLE TIME OF SYNCHRONOUS CIRCUIT WITH BOUNDED DELAYS

by

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

Memorandum No. UCB/ERL M92/56

28 May 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Minimum Cycle Time of Synchronous Circuit With Bounded Delays*

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

Traditionally, the minimum cycle time of a synchronous sequential circuit is obtained by computing the delay of the combinational logic of the circuit. This combinational delay can underestimate the true minimum cycle time, especially when the circuit has long false paths. In contrary, we define sequential delay of a synchronous sequential circuit to include the effect of the memory elements in the circuit, so that this definition gives the true minimum cycle time. We then give a condition under which combinational delay is equal to the sequential delay, or the minimum cycle time. Further, we formulate the problem of computing the exact minimum cycle time as a mixed Boolean linear programming problem, and provide efficient algorithms to compute the exact minimum cycle time for three delay models. In the first delay model, gate delays in a circuit are fixed constants; in the second delay model, gate delays vary within bounded intervals with tracking coefficient ϵ; in the third delay model, gate delays vary within bounded intervals independently. The complexities of the algorithms for the three delay models range from the simplest for the first case and to the most complex for the third case. In solving the mixed Boolean linear programming problem, a lower bound updating technique is used to decrease the complexity of the problem progressively. And the core computation of the problem is translated into the problems of tautology checking, test generation, redundancy check, and SAT.

1 Introduction

Computing the minimum cycle times of synchronous sequential circuits accurately are indispensable in high speed digital design. All previous approaches treat this problem combinationally: computing the delays of the combinational logic of the circuits as the minimum cycle times. In these approaches, various definitions of delay are used; the definitions vary in how the inputs are excited and what the states of the circuits are. Some examples of definitions of delay are: delay by static sensitization, delay by dynamic sensitization (or 2 vector) [MB89] [DKM91], and floating delay [CD90]. It is known that delay by 2-vector reflects more accurately the essence of delay in practical situations. In computing the delay by 2-vector, the input to the circuit is a pair of vectors switching at $t = 0$, the latest time of the last transition for all possible 2-vector inputs is the delay of the circuit. Hence, the delay by 2-vector of the combinational logic of a synchronous sequential circuit is a good estimate for the minimum cycle time. In this paper, we demonstrate that delays by 2-vector can underestimate the minimum cycle times, and give a condition under which delays by 2-vector are the minimum cycle times. Then, we present our approach to compute the minimum cycle times of synchronous sequential circuits. In our approach, we first give a definition of sequential delay (or minimum cycle time), which takes into account the effect of the memory elements as well as the combinational logic in the synchronous sequential circuits. Then, we consider the problem of computing the minimum cycle time with bounded gate delays.

This problem is first formulated as a mixed Boolean linear programming problem; then, we present an efficient algorithm to solve the mixed Boolean linear programming problem. Our solution is exact if all transitions of the synchronous sequential circuit (finite state machine) are possible.

The organization of this paper is as follows. First, we introduce Timed Boolean Functions, and apply them to modeling and circuit formulation. Second, we consider the relationship between delay by 2-vector and minimum cycle time. We demonstrate the inadequacy of defining minimum cycle time only in terms of the delays of combinational circuits; then we define sequential delay to include the effect of memory elements so that the sequential delay is equal to the minimum cycle time, and give a condition under which delay by 2-vector is equal to the sequential delay. Finally, we compute the *exact* minimum cycle times of synchronous sequential circuits with three delay models. In the first case, the gate delays are fixed; in the second case, gate delays the circuit track with coefficient ϵ; in the third case, gate delays of the circuit can vary independently within bounded intervals.

2 Timed Boolean Function

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:*

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

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

- *$F(v) = v$, $v \in W$, is a Timed Boolean Function.*
- *If $G : W^{n_1} \mapsto W, H : W^{n_2} \mapsto W$ are Timed Boolean Functions, then, $F = \overline{G}, F = G \cdot H, F = G + H$ are also Timed Boolean Functions.*

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

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

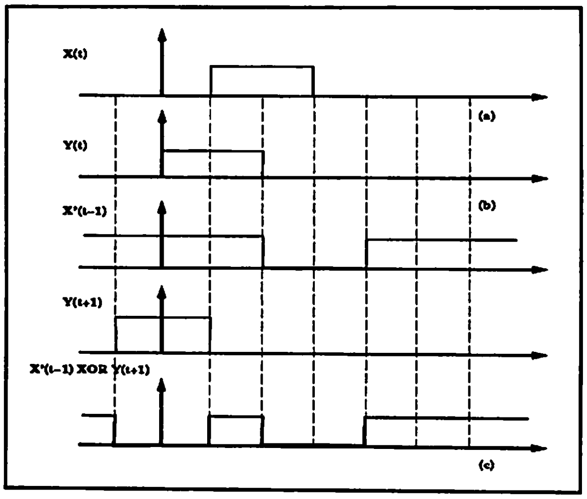

Figure 1: Representing Waveforms by TBF

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. 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.

4. Edge triggered D- flip flop with a common clock of period P. Let Q, D, d be the output, the data input, and the delay of the flip flop, respectively; then the flip flop is represented by

$$Q(t) = D\left(P \cdot \left\lfloor \frac{t-d}{P} \right\rfloor\right)$$

where $\lfloor x \rfloor$ = the greatest integer not exceeding x.

Example 3 *Assume the data input waveform D(t) is given, P=3, and d=2. We would like to compute the output value at t=31. Since*

$$Q(t) = D\left(3 \cdot \left\lfloor \frac{t-2}{3} \right\rfloor\right)$$

we get Q(31)=D(27).

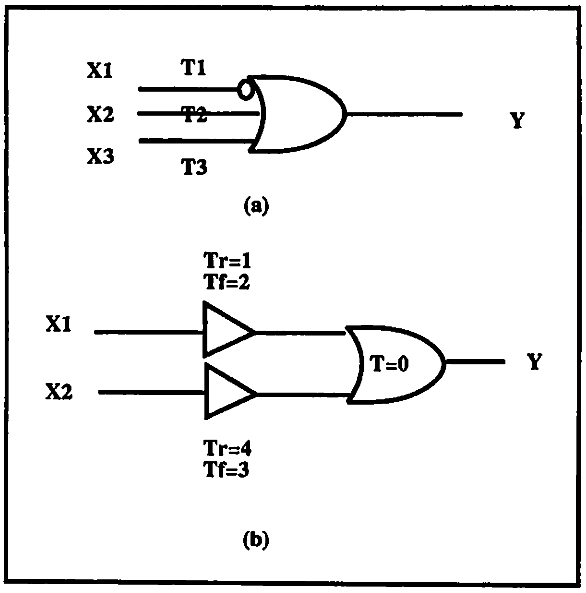

Figure 2: Modeling With TBF

5. Synchronous transparent latch. Let Q, D, d be the output, data input, and delay of the latch. Assume that the latch is transparent when the clock phase is positive, and that the first part of a clock period P is negatively phased and has the duration P_1, the second part is positively phased and has the duration P_2. Thus, $P_1 + P_2 = P$. The latch is represented by

$$Q(t) = D\left((1-\alpha)\left\lfloor\frac{t-d}{P}\right\rfloor P + \alpha\cdot(t-d)\right)$$

If $P_1 \geq P_2$, $\alpha = \left\lfloor\frac{(t-d)\bmod P}{P_1}\right\rfloor$; and if $P_1 \leq P_2$, $\alpha = \left\lfloor\frac{(t-P_1-d)\bmod P}{P_2}\right\rfloor + 1$. $\alpha = 0$ if $t-d$ is in the "latched" part of the clock cycle; $\alpha = 1$ if $t-d$ is in the "transparent" part of the clock cycle. (a mod b) is $a - \lfloor\frac{a}{b}\rfloor b, a > 0, b > 0$.

Example 4 *Let* $P_1 = 2, P_2 = 1, d = 0$. *At* $t = 16$, $\alpha = \lfloor\frac{16\bmod 3}{2}\rfloor = 0$, $Q(16) = D(\lfloor\frac{16}{3}\rfloor 3) = D(15)$. *It can be seen that at* $t = 16$ *the latch is in "latched" mode, as calculated. At* $t = 18$, $\alpha = \lfloor\frac{18\bmod 3}{2}\rfloor = 1$, $Q(18) = D(18)$. *It can be seen that at* $t = 16$ *the latch is in "transparent" mode, as calculated.*

Note that the flip flops (edge triggered D-FF and transparent latch) are represented without feedback; its memory effect is captured by the greatest integer function $\lfloor x\rfloor$. Being able to characterize memory element enables TBF's to represent sequential circuits with complete timing information.

2.2 Synchronous Circuit Formulation With Timed Boolean Function

Once all components of a circuit are represented by TBF's, the TBF for the circuit can be derived by identifying the timed variables corresponding to the ports connected to the same net. For synchronous sequential circuit, the combinational part of the circuit is first formulated with TBF's, then composed with the TBF's for the memory elements to obtain the TBF representation for the entire synchronous sequential circuit. We illustrate this with an example.

Example 5 *In Figure 3, the delay for each gate is shown inside the gate.*

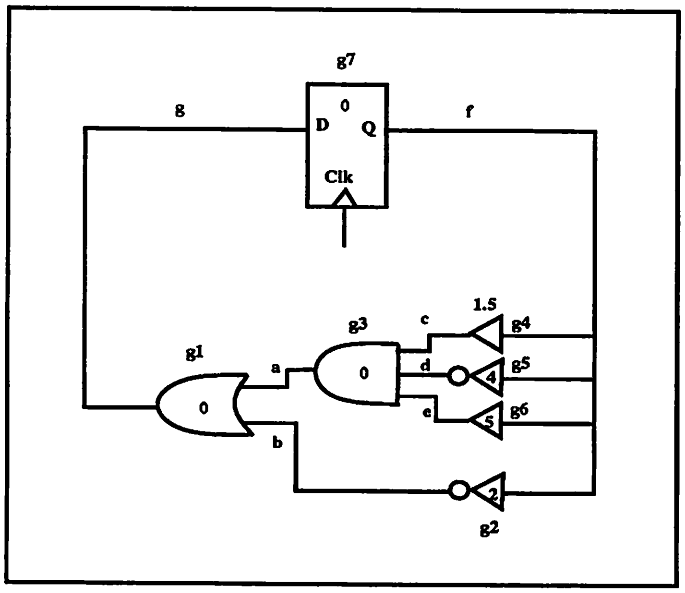

Figure 3: A Synchronous Sequential Circuit

First, we formulate the combinational part of the circuit with TBF's. Each gate is represented by a TBF, as follows.

$$\begin{aligned}g(t) &= a(t) + b(t)\\ b(t) &= f'(t-2)\\ a(t) &= c(t)d(t)e(t)\\ c(t) &= f(t-1.5)\\ d(t) &= f'(t-4)\\ e(t) &= f(t-5)\end{aligned}$$

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

$$\begin{aligned}a(t) &= f(t-1.5)f'(t-4)f(t-5)\\ b(t) &= f'(t-2)\end{aligned}$$

Therefore,

$$g(t) = f(t-1.5)f'(t-4)f(t-5) + f'(t-2)$$

The TBF for the D flip flop is

$$f(t) = g(\lfloor \frac{t}{\tau} \rfloor)$$

where τ is the cycle time of the synchronous sequential circuit.

Now, compose the two set of TBF's to obtain:

$$g(t) = g(\lfloor \frac{t-1.5}{\tau} \rfloor) g'(\lfloor \frac{t-4}{\tau} \rfloor) g(\lfloor \frac{t-5}{\tau} \rfloor) + g'(\lfloor \frac{t-2}{\tau} \rfloor) \quad (1)$$

This equation represents the complete functionality and timing information of the synchronous sequential circuit shown in Figure 3.

Comments:

1. For combinational circuits, when each circuit component is represented by a TBF having time argument of the form $t - k_i$, k_i is a constant, then the TBF for the circuit has only the time arguments of the form $t - k_i$.

2. The TBF's for synchronous sequential circuits with cycle time τ can be derived systematically as described below. Assume that all external inputs to the circuit are synchronized to the clock as shown in Figure 4. The TBF's for the combinational logic have the general form:

$$y_i(t) = f_i(x_1(t-k_{i1}), ..., x_n(t-k_{in}), v_1(t-k_{r_{i1}}), ..., v_l(t-k_{r_{il}})) \quad (2)$$

   We can treat v_i's as states of the circuit, hence,

$$y_i(t) = f_i(x_1(t-k_{il}), ..., x_s(t-k_{is}))$$

   where k_{ij} is the delay from ith flip flop's output to the jth flip flop's data input. Incorporating the flip flops' TBF's, we obtain:

$$y_i(t) = f_i(y_1(\lfloor \frac{t-k_{i1}-d_{f_1}}{\tau} \rfloor \tau), ..., y_s(\lfloor \frac{t-k_{is}-d_{f_s}}{\tau} \rfloor \tau)$$

$$y_i(t) = f_i(y_1(\lfloor \frac{t-h_{i1}}{\tau} \rfloor \tau), ..., y_s(\lfloor \frac{t-h_{is}}{\tau} \rfloor \tau) \quad (3)$$

   where d_{f_i} is the delay of the ith flip flop. $y_{n+1}, ..., y_s$ are the external inputs. Therefore, $h_{ij} = k_{ij} + d_{f_j}$ is the delay around the loop from the jth flip flop's input to the ith flip flop's input. Therefore,

$$y_i(n\tau) = f_i(y_1(n\tau + \lfloor \frac{-h_{i1}}{\tau} \rfloor \tau), ..., y_s(n\tau + \lfloor \frac{-h_{is}}{\tau} \rfloor \tau))$$

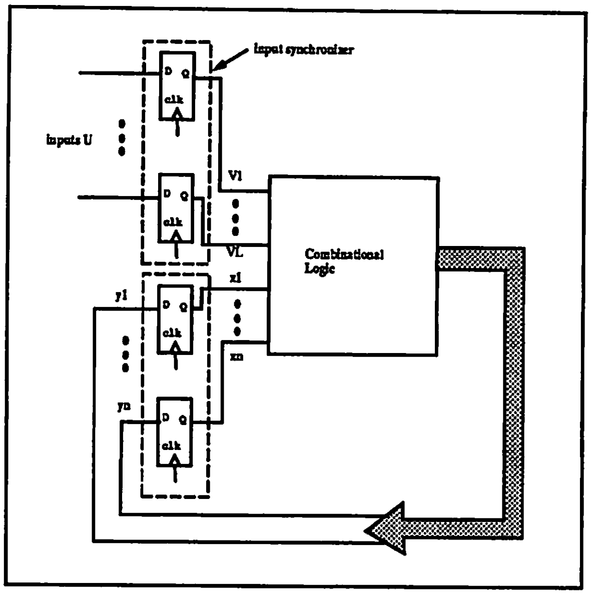

Figure 4: Block Diagram of a Synchronous Sequential Circuit

Normalizing to τ, the nth value of $y_i(t)$ is:

$$y_i(n) = f_i(y_1(n + \lfloor \frac{-h_{i1}}{\tau} \rfloor), ..., y_s(n + \lfloor \frac{-h_{is}}{\tau} \rfloor))$$

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. Let us illustrate with an example.

Example 6 *The TBF of the combinational logic of the circuit in Figure 3 is* $g(t) = f(t - 1.5)f'(t-4)f(t-5) + f'(t-2)$. *Because* $f(t)$ *is the output of the D-flip flop,* $f(t)$ *switches at* $t = ..., -\tau, 0, \tau, ...$ *Let* $f(n)$ *be the Boolean value of* $f(t)$ *for* $n\tau \leq t < (n+1)\tau$. *Assume*

$\tau = 3$. *We can evaluate* $g(t)$ *for* $0 \leq t < 3$, *as follows.*

$$\begin{array}{ll}
\textit{For } 0 \leq t < 1, & f(t-1.5) = f(-1) \\
& f(t-2) = f(-1) \\
& f(t-4) = f(-2) \\
& f(t-5) = f(-2) \\
\textit{Thus,} & g(t) = f'(-1) \\
\textit{For } 1 \leq t < 1.5, & f(t-1.5) = f(-1) \\
& f(t-2) = f(-1) \\
& f(t-4) = f(-1) \\
& f(t-5) = f(-2) \\
\textit{Thus,} & g(t) = f'(-1) \\
\textit{For } 1.5 \leq t < 2, & g(t) = f(0)f'(-1)f(-2) + f'(-1) \\
\textit{For } 2 \leq t < 3, & g(t) = f'(0)
\end{array}$$

This quantizing effect is taken into account automatically by the TBF's of the flip flops. With the flip flops' TBF's:

$$\begin{aligned}
g(t) = \; & g(\lfloor \tfrac{t-1.5}{3} \rfloor) g'(\lfloor \tfrac{t-4}{3} \rfloor) g(\lfloor \tfrac{t-5}{3} \rfloor) + g'(\lfloor \tfrac{t-2}{3} \rfloor) \\
= \; & g(\lfloor \tfrac{t-1.5}{3} \rfloor) g'(\lfloor \tfrac{t-1}{3} \rfloor - 1) g(\lfloor \tfrac{t-2}{3} \rfloor - 1) + g'(\lfloor \tfrac{t-2}{3} \rfloor)
\end{aligned}$$

It is easy to see that the "break points" for $0 \leq t < 3$ *are 1, 1.5, 2. Thus,*

$$\begin{array}{ll}
\textit{For } 0 \leq t < 1, & g(-1)g'(-2)g(-2) + g'(-1) \\
& = g'(-1) \\
\textit{For } 1 \leq t < 1.5, & g(-1)g'(-1)g(-2) + g'(-1) \\
& = g'(-1) \\
\textit{For } 1.5 \leq t < 2, & g(0)g'(-1)g(-2) + g'(-1) \\
\textit{For } 2 \leq t < 3, & g(0)g'(-1)g(-1) + g'(0) \\
& = g'(0)
\end{array}$$

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 waveform 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 into intervals $\{I_l\}$ by the end points of the intervals $\{g_{ij}^{-1}(\tau_{ijk})\}$. For each interval I_l, there exists some k such that $g_{ij}(I_l) \subseteq \lfloor \tau_{ijk}, \tau_{ij(k+1)} \rfloor$. 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, $\{\tau_{ijk}\} = \{3n : n \text{ is an integer}\}$, end points of the intervals $\{g_{ij}^{-1}(\tau_{ijk})\} = \{1 + 3n, 1.5 + 3n, 2 + 3n : n \text{ is an integer}\}$, $\{k(l)\} = \{n \text{ if } I_l \subseteq \lfloor 3n, 3(n+1) \rfloor\}$.

2.4 Decision Diagram For Timed Boolean Function

TBF's have BDD decision diagrams. In this section, we consider the decision diagrams for TBF's of the form as in equation (2) and of the form as in equation (3). The first type of TBF's derives from combinational circuits, and the second type derives from synchronous sequential circuits.

We first consider the TBF's of combinational circuits, For this type of TBF's, we consider the inputs that are a pair of vectors switching at t=0. For other inputs, the following results can be easily extended. In a TBF, we 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 earlier than those of normal Boolean variables.

To build a decision diagram for a TBF, we first insert all k_i's into a binary tree. Hence, the left branch of a k_i node represents the TBF for $t < k_i$, the right branch, the TBF for $t > k_i$. Thus, as we traverse from the root of the k_i tree down to a leave node, for each k node we encounter, some timed variable(s) $x(t - k)$ in the TBF become Boolean variables either $x(0^+)$ or $x(0^-)$, depending right or left branch of the k node is taken. Therefore, when we arrive a leave node, the TBF becomes an ordinary Boolean function, which will be called the resolved Boolean function at the leave node.

When all k_i's are inserted into a binary tree, to each leave node of this tree, we attach a binary decision diagram of the resolved Boolean function at the leave node.

Example 7 *Consider the combinational logic of the circuit in Figure 3,*

$$g(t) = f(t - 1.5)f'(t - 4)f(t - 5) + f'(t - 2) \tag{4}$$

For $t < 1.5, 1.5 < t < 2$, $g(t) = f'(0^-)$; *for* $2 < t < 4, 4 < t < 5, 5 < t$, $g(t) = f'(0^+)$. *Therefore, we get the decision diagram as shown in Figure 5(a). Reducing the decision diagram, we get Figure 5(c), which is the TBF* $g(t) = f'(t-2)$. *Hence, under 2-vector input, equation (4) is equivalent to* $g(t) = f'(t - 2)$. *In other words, the combinational logic in Figure 3 behaves like an inverter with delay of 2, under 2-vector inputs.*

For TBF's of synchronous sequential circuits, we work in terms of modulo τ. Referring to equation (3), let $t = r + n\tau$, $h_{ij} = h'_{ij} + m_{ij}\tau$, where $0 < r < \tau, 0 < h'_{ij} < \tau$. Without loss of generality, we consider the case $n = 0$; so,

$$y_i(r) = f_i(y_1(m_{i1}\tau + \lfloor \frac{r - h'_{i1}}{\tau} \rfloor \tau), ..., y_s(m_{is}\tau + \lfloor \frac{r - h'_{is}}{\tau} \rfloor \tau)$$

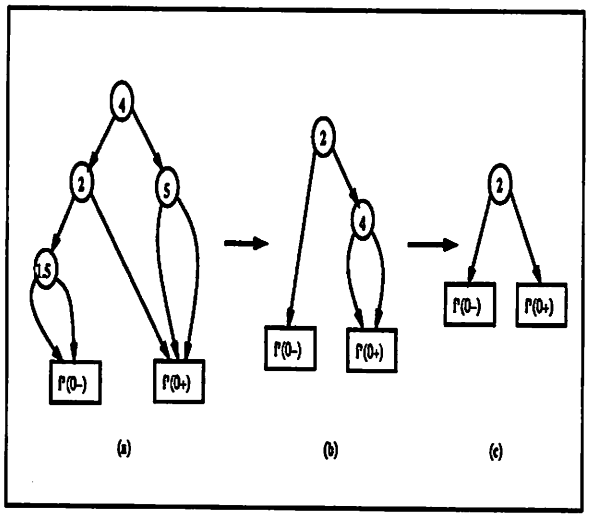

Figure 5: Decision Diagram For $f(t-1.5)f'(t-4)f(t-5)+f'(t-2)$

Normalizing to τ,

$$y_i(r) = f_i(y_1(m_{i1} + \lfloor\frac{r-h'_{i1}}{\tau}\rfloor), \ldots, y_s(m_{is} + \lfloor\frac{r-h'_{is}}{\tau}\rfloor)$$

Note that $\lfloor\frac{r-h'_{ij}}{\tau}\rfloor \in \{-1,0\}$. Now, treat $\{h'_{ij}\}$ as binary variables. h'_{ij} takes the value of 0, when $r < h'_{ij}$, otherwise, 1. A possible good variable ordering is to order h'_{ij}'s earlier than that of normal Boolean variables. To build a decision diagram, first insert h'_{ij}'s into a binary tree. When all h'_{ij}'s are all inserted into a binary tree, to each leave node of the tree, we attach the resolved binary decision diagram of that node.

Example 8 *Take the TBF(1) from example 5, with $\tau = 2$.*

$$g(t) = g(\lfloor\frac{t-1.5}{\tau}\rfloor)g'(\lfloor\frac{t-4}{\tau}\rfloor)g(\lfloor\frac{t-5}{\tau}\rfloor) + g'(\lfloor\frac{t-2}{\tau}\rfloor)$$

Let $t = r + n\tau$, and $n = 0$,

$$g(r) = g(\lfloor\frac{r-1.5}{2}\rfloor 2)g'(-2\cdot 2 + \lfloor\frac{r}{2}\rfloor 2)g(-1\cdot 2 + \lfloor\frac{r-1}{2}\rfloor 2) + g'(-1\cdot 2 + \lfloor\frac{r}{2}\rfloor 2)$$

normalizing to τ,

$$g(r) = g(\lfloor\frac{r-1.5}{2}\rfloor)g'(-2 + \lfloor\frac{r}{2}\rfloor)g(-1 + \lfloor\frac{r-1}{2}\rfloor) + g'(-1 + \lfloor\frac{r}{2}\rfloor)$$

equivalently,

$$g(r) = g(\lfloor\frac{r-1.5}{2}\rfloor)g'(-2)g(-1 + \lfloor\frac{r-1}{2}\rfloor) + g'(-1)$$

The decision diagram is therefore as shown in Figure 6. Along the the dashed path shown, we have imposed the conditions: $r < 1.5$ and $r < 1$. Under these conditions, the TBF becomes:

$$\begin{aligned} g(r) &= g(-1)g'(-2)g(-2) + g'(-1) \\ &= g'(-1) \end{aligned}$$

An interpretation of the diagram is that when we come to a h'_{ij} node, the left branch represents the TBF for $r < h'_{ij}$, and the right branch represents the function for $r > h'_{ij}$. When the encountered node is a Boolean variable, the usual BDD interpretation applies.

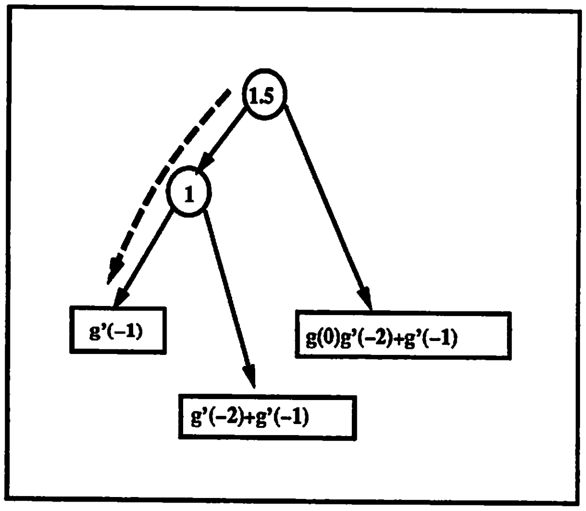

Figure 6: Decision Diagram of $g(\lfloor \frac{r-1.5}{2} \rfloor)g'(-2)g(\lfloor -1 + \frac{r-1}{2} \rfloor) + g'(-1)$

It can be seen that the decision diagrams described above are canonical. Two TBF's having the same canonical decision diagram have the same timing behaviors.

3 2-vector Delay and Minimum Cycle Time

All previous methods of computing the minimum cycle times of synchronous circuits compute the maximum delays of the combinational logic of the circuits, and treat these maximum delays as the minimum cycle times. Of all different delays for combinational circuits, delay by 2-vector is a more realistic model for this situation. In computing the delay by 2-vector, the input to the circuit is a pair of vectors switching at t=0, the time of the latest last transition for all possible input pairs is the delay of the circuit. In this section, by demonstrating that delay by 2-vector can underestimate the minimum cycle time, we illustrate the inadequacy of using the delay of the combinational logic of a synchronous sequential circuit as the minimum cycle time for the synchronous sequential circuit; and give a condition under which the 2-vector delay of the combinational logic of a synchronous circuit is the minimum cycle time for the synchronous sequential circuit.

Example 9 *Consider the synchronous sequential circuit shown in Figure 3. We first calculate the 2-vector delay of the combinational logic of the synchronous sequential circuit. From example 7, the combinational logic of the synchronous sequential circuit behaves like an inverter with delay of 2, under 2-vector inputs. This can also be seen by noting that all pathes from node f to node a are not sensitizable by any pair of vectors; specifically, node a stays at ZERO for any pair of vectors. Therefore, the effective circuit is an inverter of delay 2, i.e. gate g_2. Therefore, the 2-vector delay of the combinational logic is 2.*

The TBF of the synchronous sequential circuit is

$$g(t) = g(\lfloor\frac{t-1.5}{\tau}\rfloor\tau)g'(\lfloor\frac{t-4}{\tau}\rfloor\tau)g(\lfloor\frac{t-5}{\tau}\rfloor\tau) + g'(\lfloor\frac{t-2}{\tau}\rfloor\tau)$$

at $t = n\tau$,

$$g(n\tau) = g(n + \lfloor\frac{-1.5}{\tau}\rfloor\tau)g'(n + \lfloor\frac{-4}{\tau}\rfloor\tau)g(n + \lfloor\frac{-5}{\tau}\rfloor\tau) + g'(n + \lfloor\frac{-2}{\tau}\rfloor\tau)$$

normalizing to τ,

$$g(n) = g(n + \lfloor\frac{-1.5}{\tau}\rfloor)g'(n + \lfloor\frac{-4}{\tau}\rfloor)g(n + \lfloor\frac{-5}{\tau}\rfloor) + g'(n + \lfloor\frac{-2}{\tau}\rfloor)$$

If we let the minimum cycle time be 2, then any cycle time greater than 2 should ensure proper operations of the synchronous sequential circuit. For cycle time $\tau = 2$, n=1:

$$g(1) = g(1 + \lfloor\frac{-1.5}{2}\rfloor)g'(1 + \lfloor\frac{-4}{2}\rfloor)g(1 + \lfloor\frac{-5}{2}\rfloor) + g'(1 + \lfloor\frac{-2}{2}\rfloor)$$

$$g(1) = g(0)g'(-1)g(-2) + g'(0)$$

For cycle time $\tau = 5$, n=1:

$$g(1) = g(1 + \lfloor\frac{-1.5}{5}\rfloor)g'(1 + \lfloor\frac{-4}{5}\rfloor)g(1 + \lfloor\frac{-5}{5}\rfloor) + g'(1 + \lfloor\frac{-2}{5}\rfloor)$$

$$g(1) = g(0)g'(0)g(0) + g'(0) = g'(0)$$

which is not equal to $g(1) = g(0)g'(-1)g(-2) + g'(0)$. Therefore, the minimum cycle time is greater than 2. Hence, the 2-vector delay underestimates the minimum cycle time. A reason for this is that the pathes (from node f to node a) that are not sensitizable under 2 vectors now become sensitizable under a string of vectors spaced at 2 units of time apart.

In view of the above example, we want to define sequential delays, or the minimum cycle times, of synchronous sequential circuits, not just in terms of the combinational logic, but also in terms of the memory elements. If D_s is the minimum cycle time of a synchronous sequential circuit, we require that the synchronous sequential circuit operate properly at any cycle time $\tau \geq D_s$.

Definition 2 *Given a TBF $y(t,\tau)$ for a synchronous sequential circuit, where τ is the cycle time, the sequential delay, or the minimum cycle time, is the minimum D_s such that*

$$y(t,\tau) = y(t, D_s), \forall \tau > D_s$$

With the above definition, under what conditions will 2-vector delay be equal to the minimum cycle time? Use the notations in Figure 4. Let

$$y(n) = f(y_1(n + \lfloor\frac{-h_{i1}}{\tau}\rfloor), ..., y_s(n + \lfloor\frac{-h_{is}}{\tau}\rfloor))$$

be a TBF of a synchronous sequential circuit, then the TBF for the combinational logic of this circuit is

$$y(t) = f(y_1(t - h_{i1}), ..., y_s(t - h_{is}))$$

Assuming the delays of flip flops are included in the computation of 2-vector delay, the maximum 2-vector delay is the minimum D_2 such that

$$f(y_1(t - h_{i1}), ..., y_s(t - h_{is}) = f(y_1(D_2 - h_{i1}), ..., y_s(D_2 - h_{is})), \forall t > D_2$$

because D_2 is latest time of the last transitions. The following theorem asserts that if the 2-vector delay D_2 is at least $\frac{1}{2}max(h_{ij})$, where h_{ij} is the loop delay from jth flip flop's input to the ith flip's input, then D_2 is equal to the minimum cycle time, D_s.

Theorem 1 *Let D_2 be the 2-vector delay of the combinational logic of a synchronous sequential circuit, D_s, the minimum cycle time of the synchronous sequential circuit. If $D_2 \geq \frac{1}{2}max(h_{ij})$, then $D_2 = D_s$.*

Proof. Consider the variables in the TBF of the combinational logic.

$$\begin{aligned} y_i(t - h_{ij}) &= y_i(0^+) \overset{\text{def}}{=} y_i(n-1) \quad \text{if } t > h_{ij} \\ y_i(t - h_{ij}) &= y_i(0^-) \overset{\text{def}}{=} y_i(n-2) \quad \text{if } t < h_{ij} \end{aligned}$$

And,

$$\begin{aligned} n + \lfloor\tfrac{-h_{ij}}{t}\rfloor &= n-1 \quad \text{if } -1 \leq \tfrac{-h_{ij}}{t} < 0, \text{ equivalently, } t \geq h_{ij} > 0 \\ n + \lfloor\tfrac{-h_{ij}}{t}\rfloor &= n-2 \quad \text{if } -2 \leq \tfrac{-h_{ij}}{t} < -1, \text{ equivalently, } t \geq \tfrac{h_{ij}}{2} > \tfrac{t}{2} \end{aligned}$$

Therefore, for $t \geq \frac{1}{2}max(h_{ij})$,

$$y_i(n + \lfloor\frac{-h_{ij}}{t}\rfloor) = y_i(t - h_{ij})$$

Because D_2 is the minimum value such that

$$f(y_1(t - h_{i1}), ..., y_s(t - h_{is})) = f(y_1(D_2 - h_{1i}), ..., y_s(D_2 - h_{is}))\forall t > D_2$$

Hence, if $D_2 \geq \frac{1}{2}max(h_{ij})$, D_2 is also the minimum value such that:

$$f(y_1(n + \lfloor\frac{-h_{i1}}{\tau}\rfloor), ..., y_s(n + \lfloor\frac{-h_{is}}{\tau}\rfloor)) = f(y_1(n + \lfloor\frac{-h_{i1}}{D_2}\rfloor), ..., y_s(n + \lfloor\frac{-h_{is}}{D_2}\rfloor))\forall\tau > D_2$$

By definition, D_2 is also the minimum cycle time of the synchronous sequential circuit. □
Comments:

1. Above result can be generated to delay by n-vectors, then the condition becomes: if $D_n \geq \frac{1}{n}\max(h_{ij})$, then $D_n = D_s$.

2. In example 3, $\frac{1}{2}\max(h_{ij}) = 2.5$ and the 2-vector delay is 2, violating this condition. Note that half of the difference of the longest and shortest topological delays is $\frac{1}{2}(5 - 1.5) = 1.75$, less than the 2-vector delay, 2.

4 Computing the Minimum Cycle Time

4.1 Minimum Cycle Time for Fixed Delays

In this section, we compute the minimum cycle time for synchronous sequential circuits with each gate's delay fixed at a constant. We assume the general TBF for a synchronous sequential circuit is

$$y(n, \tau) = f(..., y_i(n + \lfloor\frac{-h_i}{\tau}\rfloor), ...) = f(..., y_i(n - z_i(\tau)), ...)$$

where $z_i(\tau) = -\lfloor\frac{-h_i}{\tau}\rfloor$, and that $h_i > h_{i+1}$. $z_i(\tau)$ changes values only at $\tau = \frac{h_i}{m}, m = 1, 2, ...$ Specifically, $z_i(\tau) = 1, 2, ...$, for $\tau \geq h_i, h_i > \tau \geq \frac{h_i}{2}, ...$, respectively. If the longest topological delay is taken to be the minimum cycle time, D_s, as in many practices, then, for any cycle time $\geq D_s$ the synchronous sequential circuit will operate properly. This fact is confirmed by noting that for any cycle time $\tau \geq D_s$, $n + \lfloor\frac{-h_i}{\tau}\rfloor = n - 1, \forall i$; thus, $f(..., y_i(n + \lfloor\frac{-h_i}{\tau}\rfloor), ...) = f(..., y_i(n - 1), ...)$.

To find the minimum cycle time, we start with $D_s = h_1$ and decrease D_s until $y(n, D_s) \neq y(n, h_1)$. If $y(n, \tau)$ is a constant, then, $y(n, D_s) = y(n, h_1)$ for all D_s. Therefore, need to find a lower bound on D_s. A lower bound of D_s is $min(h_i)$. This is because when $\tau < min(h_i)$, $z_i(\tau) > 1$; so all variables in $y(n, D_s)$ have arguments $< n-1$, while all variables in $y(n, k_1)$ have arguments of $n-1$; So, if $y(n, D_s) = y(n, h_1)$, $D_s < min(h_i)$, then $y(n, \tau)$ is a constant; hence the minimum cycle time is zero; if $y(n, \tau)$ is not a constant, then the minimum cycle time $\geq min(h_i)$.

As τ changes in the interval $\lfloor h_1, \min(h_i) \rfloor$, $z_i(\tau)$ changes values only a finite number of times. Therefore, there are only finite number of points in the interval $[h_1, \min(h_i)]$ at which the tautology $y(n, \tau) = y(n, h_1)$ needs to be checked.

In summary:

Algorithm for Minimum Cycle Time for Fixed Delays

1. Let $D = \{\frac{h_i}{m} : \frac{h_i}{m} \geq min(h_i), m = 1, 2, ...\}$ be the set of time constants and their harmonics above $min(h_i)$.

2. Sort D in decreasing order. Evaluate the inequality $y(n, \tau) \neq y(n, h_1)$ at the points of D, starting with the first value. If τ^* is the first value at which the inequality holds, then $D_s = \tau^* - \Delta$, where Δ is an arbitrarily small number.

Call this algorithm the **zero order computation.**

The following section computes the minimum cycle time when gate delays vary in intervals. First, we describe the delay model to be used.

4.2 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 track well, increasing or decreasing by about the same ratio; this is because of the miniature size of a chip that physical properties of the devices on the chip are subject to the similar fabrication conditions, and hence are closely matched. Although small, there is 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, ϵ, 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}.

4.3 Minimum Cycle Time with Delay Tracking ϵ

In practical situations, we assume the delay model discussed in the above section.

Definition 4 *If all the timed variables in a TBF are of the form*

$$x(t - \sum_{k=1}^{n_i} d_{l_k^i}) = x(t - h_i)$$

where d_i is the ith gate's delay variable, $\{h_i\}$ are called the time constants of the TBF. $\sum_i d_i$ is called a delay sum. The set of delay sums that add up to k_i is denoted by $[h_i]$. 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 1,2,3; and $[1] = \{d_1\}, [2] = \{d_2\}, [3] = \{d_2 + d_3\}$.

Because d_i varies randomly in the interval $[rd_i^0(1-\epsilon), rd_i^0(1+\epsilon)]$, a time constant h_i in a design may give rise $|[h_i]|$ time constants in a manufactured circuit. In other words, process variations modeled by ϵ may cause each delay sum in $[h_i]$ to add up to a slightly different time constant; and there are $|[h_i]|$ such delay sums. The maximum difference of these delay sums is $\epsilon \cdot h_i$. So a single time constant in a design becomes a band of time constants of width $\epsilon \cdot h_i$.

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

Definition 5 *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 $p - \epsilon$.*

An example of the relationship between the time constants of a design and the time constants of a manufactured circuit of the design is shown in Figure 7. In the figure, output waveforms of a design and a manufactured circuit are plotted. The output of the manufactured circuit is that of the design scaled by a factor $r = \frac{2}{3}$, and at the times of the time constants $\{r \cdot k_i\}$, appear groups of glitches due to manufacturing variations.

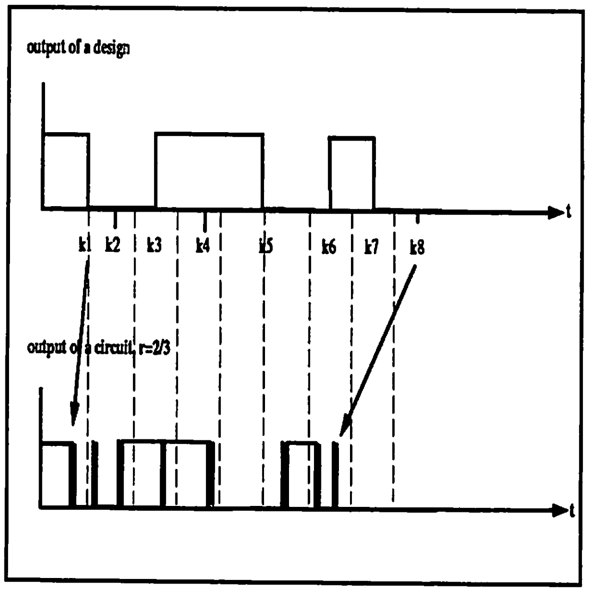

Figure 7: Splitting of Time Constants Into Bands

To compute the minimum cycle time for a synchronous sequential circuit with gate delays varying in bounded intervals $[d_i^{min}, d_i^{max}]$ and tracking with coefficient ϵ, we find a lower bound on the minimum cycle time by considering the circuit in which every gate takes on its maximum delay. Use the zero order computation to compute the minimum cycle time for this circuit. This cycle time is a lower bound, and set it equal to D_s.

Now, let the ith gate's delay vary in the interval $[(p - \epsilon)d_i^{max}, pd_i^{max}]$, with p=1. We want to enumerate all possible values of $z_{ij}(t)$. $z_{ij}(t) = -\lfloor\frac{-h_{ij}}{t}\rfloor$, h_{ij} can vary in the interval $[(p - \epsilon)h_i, ph_i]$. The possible values can be calculated as follows. Plot the band of h_i, i.e. the interval $[(p - \epsilon)h_i, ph_i]$, and draw vertical lines at $t, 2t, 3t, ...$, as in Figure 9. If interval $[m_1 t, m_2 t]$, where m_i is a nonnegative integer, is the smallest interval that contains the band of h_i, then, it is easy to see that the possible values of $z_{ij}(t)$ are $\{m_1 + 1, ..., m_2\}$. For example, referring to Figure 9, $[2t_2, 7t_2]$ is the smallest interval that contains S_3, thus, the possible values of $z_{3j}(t_2)$ are $\{3, ..., 7\}$.

Another way to enumerate the possible values of $z_{ij}(t)$ is to plot the band of h_i and its harmonics above D_s, i.e. $[(p-\epsilon)h_i, ph_i]$, $[(p-\epsilon)\frac{h_i}{2}, p\frac{h_i}{2}]$,... If $t \in [(p-\epsilon)\frac{h_i}{m}, p\frac{h_i}{m}]$ for some m, then, the possible values of $z_{ij}(t) = -[\frac{-h_{ij}}{t}] = \{m, m+1\}$. Let $C = \{m_j : t \in [(p-\epsilon)\frac{h_i}{m_j}, p\frac{h_i}{m_j}]\}$, If $C \neq \phi$, the possible values of $z_{ij}(t)$ is :

$$\bigcup_{m_i \in C} \{m_i, m_i + 1\}$$

If $C = \phi$, there exists a m such that $t \in [p\frac{h_i}{m+1}, (p-\epsilon)\frac{h_i}{m}]$; so $z_{ij}(t) = m + 1$. This method of enumeration will be used in the following discussion.

Partition the t-axis by points $\{(p-\epsilon)\frac{h_i}{m}, p\frac{h_i}{m} > D_s, \forall i, m = 1, 2, ...\}$ In this case, $p = 1$. Label the partitioned intervals from right to left by $I_1, I_2, ...$

Definition 6 *1. Let $z(I)$ be the set of all possible values of $z(t) = (z_{11}(t), ..., z_{ij}(t), ...)$, for $t \in I$. A value $\sigma = (\sigma_{i1}, ..., \sigma_{ij}, ...) \in z(I)$ is feasible if the following inequalities are satisfiable:*

$$(p-\epsilon)d_i^{max} \leq d_i \leq pd_i^{max}$$
$$z_{ij}(I) = -\lfloor -\frac{h_{ij}}{t} \rfloor = \sigma_{ij}$$

equivalently,

$$(p-\epsilon)d_i^{max} \leq d_i \leq pd_i^{max}$$
$$(\sigma_{ij} - 1)t < \sum_{k}^{n_{ij}} d_{l_k^{ij}} \leq (\sigma_{ij})t$$

2. Let $f(n-\sigma)$ denote $f(..., y_{ij}(n - \sigma_{ij}), ...)$. An interval I is feasible if there exists a feasible $\sigma \in z(I)$ such that $f(\sigma) \neq f(n-1)$.

Search the minimum cycle time in the intervals, starting from I_1. A lower bound of the minimum cycle time is calculated in the first feasible interval. Symbolically, let I_f be the first feasible interval,

$$\text{A lower bound of } D_s = \max_{\sigma} \tau(\sigma), \sigma \in z(I_f)$$
$$f(n-\sigma) \neq f(n-1)$$
$$\tau(\sigma) = \max t$$
$$(p-\epsilon)d_i^{max} \leq d_i \leq pd_i^{max}$$
$$(\sigma_{ij} - 1)t < \sum_{k}^{n_{ij}} d_{l_k^{ij}} \leq (\sigma_{ij})t$$

Comments:

1. For this linear programming problem to be meaningful, the strict inequality $a < b$ needs to be changed to $a + \Delta \leq b$, where Δ is an arbitrarily small number. Hence, if $I_f = [r, s]$ is the first feasible interval, and if $|\tau(\sigma_1) - s| \leq \Delta$, then a lower bound is s; so other $\sigma \in z(I_f)$ need not be considered.

2. In enumerating values of $z(I)$, some values of $z(I)$ may have been considered in the previous enumerations; therefore, only those not yet considered values in $z_{ij}(I)$ are chosed.

Now continue decreasing the scaling factor p until some bands above the lower bound D_s start to merge, and the position at which they merge is above D_s. Let p_1 be the value of p at which some bands $> D_s$ start to merge at above D_s, and $p_2 < p_1$, be the value of p at which some other bands $> D_s$ start to merge at above D_s. For $p \in [p_1, p_2]$, the overlapping of the merging bands forms new intervals, $\{I_m\}$. If I_m is feasible, a lower bound of minimum cycle time is computed as follows:

$$D_s = \max_{\sigma} \tau(\sigma), \sigma \in z(I_m)$$

$$f(n-\sigma) \neq f(n-1)$$

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

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

$$p_1 \leq p < p_2$$

$$(\sigma_{ij} - 1)t < \sum_{k}^{n_{ij}} d_{l_k^{ij}} \leq (\sigma_{ij})t$$

where $\sigma \in z(I_m)$. Above procedure is repeated with decreasing p and increasing lower bound D_s, until all bands of time constant above D_s either do not merge or merge below D_s when p equals ρ. Then the minimum cycle time of the circuit is D_s.

In summary,

Algorithm for Computing Minimum Cycle Time with Tracking ϵ:

1. Perform zero order computation on the circuit in which all gate delays take on their respective maximum values to obtain a lower bound for the minimum cycle time, D_s.

2. Partition t-axis into intervals by points $(1-\epsilon)\frac{h_i}{m}, \frac{h_i}{m} > D_s$. Search for the first feasible interval I_f in the order of decreasing magnitude. The lower bound is updated as follows:

$$D_s = \max_{\sigma} \tau(\sigma), \sigma \in z(I_f)$$

$$f(n-\sigma) \neq f(n-1)$$

$$\tau(\sigma) = \max\ t$$
$$(p-\epsilon)d_i^{max} \le d_i \le pd_i^{max}$$
$$(\sigma_{ij}-1)t < \sum_k^{n_{ij}} d_{l_k^{ij}} \le (\sigma_{ij})t$$

Steps (1) and (2) are called the **first order computation.**

3. Calculate p_i at which some bands $> D_s$ start to merge above D_s and $p_2 < p_1$ at which some other bands $> D_s$ start to merge above D_s. For $p \in [p_1, p_2]$, the overlapping of the merging bands forms new intervals. Starting from the greatest new band above D_s, determine the first feasible I_m, and update D_s as follows.

$$D_s = \max_\sigma \tau(\sigma), \sigma \in z(I_m)$$
$$f(n-\sigma) \ne f(n-1)$$
$$\tau(\sigma) = \max\ t$$
$$(p-\epsilon)d_i^{max} \le d_i \le pd_i^{max}$$
$$p_1 \le p < p_2$$
$$(\sigma_{ij}-1)t < \sum_k^{n_{ij}} d_{l_k^{ij}} \le (\sigma_{ij})t$$

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

4.4 Validating Condition For First Order Computation

In this section, we derive the conditions under which the first order computation gives the minimum cycle time of the circuit. Suppose that we have performed the first order computation to get a lower bound D_s, and found that the bands of $b_1, ..., b_n > D_s$ are infeasible, where $b_i = \frac{h_i}{m}$, for some m. Since D_s is a lower bound, we will ignore the bands $< D_s$ and consider only the bands $> D_s$.

Definition 7 *The disjoint succeeding band(s) of band $B(b_i) = [(p-\epsilon)b_i, pb_i]$ is the greatest band(s) $B(b_j) = (p-\epsilon)b_j, pb_j]$, i.e. the greatest b_j, satisfying:*

1. $b_i > b_j$

2. $B(b_i) \bigcap B(b_j) = \phi$

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

Condition 1) gives:

$$(b_j - b_i) \cdot \rho \geq b_i \cdot \epsilon$$

equivalently,

$$\frac{b_i}{b_j} \leq 1 - \frac{\epsilon}{\rho}$$

Conditions 2) gives:

$$p^* \cdot b_i \leq D_s$$

at

$$p^* = \frac{\epsilon}{1 - \frac{b_i}{b_j}}$$

at which bands of b_i and b_j start to merge. Equivalently,

$$\frac{1}{b_i} - \frac{1}{b_j} \geq \frac{\epsilon}{D_s}$$

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

1.

$$\frac{b_i}{b_j} \leq 1 - \frac{\epsilon}{\rho}$$

2.

$$\frac{1}{b_i} - \frac{1}{b_j} \geq \frac{\epsilon}{D_s}$$

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

1. $b_i \leq (\frac{\rho}{\epsilon} - 1) \cdot l_i$

2. $b_i \leq \sqrt{\frac{l_i D_s}{\epsilon} + \frac{l_i^2}{4}} - \frac{l_i}{2}$

For a given b_i, the validating values of l_i are the shaded area in Figure 8.

Comments:

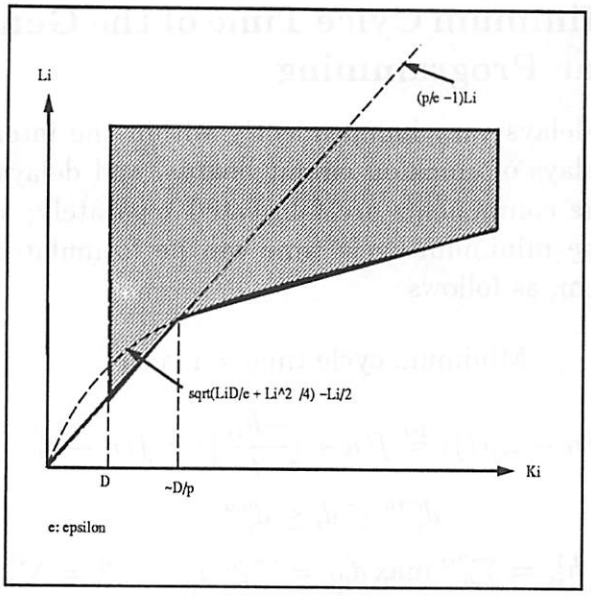

Figure 8: Valid Region For First Order Analysis

1. Figure 8 shows that the intervals between time constants and their harmonics 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 10 *Let the tracking coefficient* $\epsilon = 1\%$, $\rho = \frac{d_i^{min}}{d_i^{max}} = 50\%$, $D_s = 100$, *and the ineffective bands of time constants above* D_s *be* $h_4 = 102, h_3 = 104, h_2 = 106, h_1 = 216$. *So, the bands and harmonics above the lower bound are:* $h_4 = 102, h_3 = 104, h_2 = 106, h_1 = 216, \frac{h_1}{2} = 108$. *Then, minimum of* $1/b_i - /b_j$ = *1/106 - 1/108* $= 1.75 \times 10^{-3} > \epsilon/D_s = 10^{-3}$. *Therefore, the first order computation gives the true delay of circuit. The minimum cycle time of the circuit is therefore* $D_s = 100$. *Note the intervals between the time constants and their harmonics 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 $> D_s$ merge above D_s at $p^* \geq \rho$. If willing to trade accuracy for speed, we assume all merging bands at $p^* \geq \rho$ produce realizable transitions; hence, an upper bound for the delay is $p^* \cdot b_m > D_s$, where b_m is the greatest band merged at $p^* \geq \rho$.

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

4.5 Computing Minimum Cylce Time 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 minimum cycle time can be formulated as a mixed Boolean linear programming problem, as follows.

$$\text{Minimum cycle time} = \max \text{t}$$

$$f(n - z_{ij}(t)) \overset{\text{def}}{=} f(n + \lfloor \frac{-h_{ij}}{t} \rfloor) \neq f(n-1)$$

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

where $h_{ij} = \sum_k^{n_{ij}} d_{l_k^{ij}}$. Let $h_{ij}^1 = \sum_k^{n_{ij}} \max d_{l_k^{ij}} = \sum_k^{n_{ij}} d_{l_k^{ij}}^{max}$, $h_{ij}^0 = \sum_k^{n_{ij}} \min d_{l_k^{ij}} = \sum_k^{n_{ij}} d_{l_k^{ij}}^{min}$, $h_{min} = \min h_{ij}^0$ When $t < h_{min}, z_{ij}(t) > 1$. So if $f(n - z_{ij}(t) = f(n-1)$ at $t < k_{min}$, then $f(n-1) = 1$ or 0; then the minimum cycle time is 0. Therefore, if $f(n-1) \neq$ a constant, the minimum cycle time $\geq h_{min}$. Hence, the possible values of $z_{ij}(t)$ are $-\lfloor -\frac{h_{ij}^1}{h_{min}} \rfloor, \ldots, 1$. Denote a value of $z(t) \overset{\text{def}}{=} (z_1 1(t), \ldots, z_{ij}(t), \ldots)$ by $\sigma = (\sigma_{11}, \ldots, \sigma_{ij}, \ldots)$. The problem of computing the minimum cycle time becomes:

$$MCT = \max_{\sigma} \tau(\sigma)$$

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

$$f(n - \sigma) \neq f(n-1)$$

$$(\sigma_{ij} - 1)t < \sum_k^{n_{ij}} d_{l_k^{ij}} \leq \sigma_{ij} t$$

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

Example 11 *Consider the TBF of a synchronous sequential circuit,*

$$f(t) = g(\lfloor \frac{t - d_1}{\tau} \rfloor \tau) g'(\lfloor \frac{t - d_2 - d_3}{\tau} \rfloor \tau) g(\lfloor \frac{t - d_3 - d_4}{\tau} \rfloor \tau) + g(\lfloor \frac{t - d_2}{\tau} \rfloor \tau)$$

$$1 \leq d_1 \leq 1.5$$

$$1 \leq d_2 \leq 2$$

$$1 \leq d_3 \leq 2$$

$$1 \leq d_4 \leq 3$$

Normalizing to τ,

$$f(z) = g(n - z_1)g'(n - z_2)g(n - z_3) + g(n - z_4)$$

where $z_1 = -\lfloor\frac{-d_1}{\tau}\rfloor$, $z_2 = -\lfloor\frac{-d_2-d_3}{\tau}\rfloor$, $z_3 = -\lfloor\frac{-d_3-d_4}{\tau}\rfloor$, $z_4 = -\lfloor\frac{-d_2}{\tau}\rfloor$. $f(n-1) = g(n-1)g'(n-1)g(n-1) + g'(n-1) = g'(n-1)$. $h_{min} = 1, h_1^1 = 1.5, h_2^1 = 4, h_3^1 = 5, h_4^1 = 2$. *Therefore, the possible values of* $z_1, ..., z_4$ *are* $\{-\lfloor\frac{-1.5}{1}\rfloor = 2, 1\}$,$\{4,3,2,1\}$, $\{5,4,3,2,1\}$, $\{2,1\}$, *respectively. Let* $\sigma_1 = (1,4,3,1)$. *Determine whether* $f(n - \sigma_1) \neq f(n-1)$, *or,* $g(n-1)g'(n-4)g(n-3) + g'(n-1) \neq g'(n-1)$. *If they are not equal, a lower bound for the minimum cycle time is calculated from the following linear programming.*

$$\max\ t$$

$$0 < d_1 \leq t$$

$$3t < d_2 + d_3 \leq 4t$$

$$2t < d_3 + d_4 \leq 3t$$

$$0 < d_2 \leq t$$

$$1 \leq d_1 \leq 1.5$$

$$1 \leq d_2 \leq 2$$

$$1 \leq d_3 \leq 2$$

$$1 \leq d_4 \leq 3$$

If the two Boolean functions are equal, pick another σ and repeat above.

The result from above mixed Boolean linear programming is a lower bound of the minimum cycle time of the circuit. The minimum cycle time is the greatest lower bound of σ's. The number of σ's 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.

4.6 Solving Mixed Boolean Linear Programming

Observe that $f_{\tau>D_s}(t,\tau), f(t,\tau)$ restricted to $\tau > D_s$, is no more complicated than $f(t,\tau)$. Because $f_{\tau>D_s}(t,\tau)$ is derived from $f(t,\tau)$ by replacing $x(n + \lfloor-\frac{h_i}{\tau}\rfloor)$ by $x(n-1)$ if $h_i < D_s$. For example, $f(t,\tau) = a(n + \lfloor-\frac{3}{\tau}\rfloor)b'(n + \lfloor-\frac{2}{\tau}\rfloor)c(n + \lfloor-\frac{1.6}{\tau}\rfloor)$; then $f_{\tau>2}(t,\tau) = a(n + \lfloor-\frac{3}{\tau}\rfloor)b'(n-1)c(n-1)$; Hence, once a lower bound D_s is computed, $f(t,\tau)$ is replaced by $f_{\tau>D_s}(t,\tau)$.

Therefore, start with the first order computation to compute a lower bound D_s. The ϵ is set to be a small number $< \min \mid b_i - b_j \mid$, where $b_i = \frac{h_i}{m}$, for some m. Replace $f(t,\tau)$ by

$f_{\tau>D_s}(t,\tau)$. Then, compute a lower bound for a σ, update D_s, recompute $f_{\tau>D_s}(t,\tau)$, and repeat for another σ. Each time D_s is updated and $f_{\tau>D_s}(t,\tau)$ is recomputed, timed variable $x(n+\lfloor -\frac{h_i}{\tau}\rfloor)$ with $h_i < D_s$ become ordinary Boolean variables $x(n-1)$; Hence, in order to reduce most timed variables, early choices of σ should have as many large components as possible.

In summary,

Algorithm for Computing the Minimum Cycle Time of the General Case

1. Use first order computation to compute a lower bound D_s. The ϵ is set to be a small number $< \min \mid b_i - b_j \mid$, where $b_i = \frac{h_i}{m}$, for some m.

2.

$$MCT = \max_{\sigma} \tau(\sigma)$$

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

$$f(n-\sigma) \neq f(n-1)$$

$$(\sigma_{ij}-1)t < \sum_{k}^{n_{ij}} d_{l_k^{ij}} \leq \sigma_{ij} t$$

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

$$t > D_s$$

Specifically,

(a) For a σ, if $f(n-\sigma) \neq f(n-1)$, do:

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

$$(\sigma_{ij}-1)t < \sum_{k}^{n_{ij}} d_{l_k^{ij}} \leq \sigma_{ij} t$$

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

$$t > D_s$$

If σ is feasible, then $D_s = \tau^*$ and compute $f_{\tau>D_s}(t,\tau)$. If $f(n-\sigma) = f(n-1)$, pick another σ and repeat above linear programming.

(b) Determine the set of σ's for $f_{\tau>D_s}(t,\tau)$; if there are not yet considered σ's, go to step (a); else the minimum cycle time is D_s.

4.7 Delay Specific Enumeration

In a TBF with n timed variables, there are many σ's. However, some values are impossible due to constraints imposed by the minimum and maximum values of $\sum_i d_{n_i}$. Therefore, taking into account of the specific values of delays reduces the number of σ. For a given τ, the possible values of σ_i are the possible values $-\lfloor -\frac{\sum_i d_{n_i}}{\tau} \rfloor$ can take with $d_i^{min} \leq d_i \leq d_i^{max}$. For instance, if $\tau = 2$, $2.2 \leq \sum_i d_{n_i} \leq 6.8$, then the possible values of σ_i are $\{2,3,4\}$. The possible values of σ is then the Cartisian product of σ_i.

The procedure for delay specific enumeration is as follows. Plot the ranges of the values of delay sum of each timed variable; denote the range interval of the delay sum of timed variable x_i by S_i. Draw veritical lines at $t, 2t, 3t, ...$ At t, if $[m_1 t, m_2 t]$ is the smallest interval containing S_i, then, the possible values of $z_i(t)$ are $\{m_1 + 1, ..., m_2\}$, where m_1, m_2 are non-negative integers. To find the minimum cycle time, decrease t from the maximum of all delay sums. As t decreases, the verical lines at $t, 2t, 3t, ...$ move toward the origin accordingly. The set of possible values of $z_i(t)$ changes only when the smallest S_i-containing interval $[m_1 t, m_2 t]$ changes to $[m_1' t', m_2' t']$, where $m_1' \neq m_1$ or $m_2' \neq m_2$ or both. At each t, an $\sigma_i \in z_i(t)$ is chosen.

Example 12 *In Figure 9, there are 3 timed variables with distinct delay sums, S_i. The shaded areas are the ranges of $S_i = \sum_i d_{n_i}$. At $t = t_1$, $S_1 \subseteq [2t_1, 4t_1]$; thus, $\{\sigma_1\} = \{3,4\}$. $S_2 \subseteq [0t_1, 2t_1]$; thus, $\{\sigma_1\} = \{1,2\}$. $S_3 \subseteq [t_1, 3t_1]$; thus, $\{\sigma_1\} = \{2,3\}$. Similarly, at $t = t_2$, $\{\sigma_1\} = \{6, .., 9\}, \{\sigma_2\} = \{2, ..., 5\}, \{\sigma_3\} = \{3, .., 7\}$.*

An advantage of delay specific enumeration is that the σ's leading to a greater lower bound can be clearly chosen in this scheme. This is beneficial because greatest upper bound reduces the greatest number of timed variables.

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

5 Conclusion

In this paper, we defined sequential delay of a synchronous sequential circuit to include the effect of the memory elements in the circuit, so that this definition gives the true minimum cycle time. We then gave a condition under which combinational delay is equal to the sequential delay, or the minimum cycle time. Further, we formulated the problem of computing the exact minimum cycle time as a mixed Boolean linear programming problem, and provided efficient algorithms to compute the exact minimum cycle time for three delay models. In the first delay model, gate delays in a circuit are fixed constants; in the second delay model, gate delays vary within bounded intervals with tracking coefficient ϵ; in the third delay model, gate delays vary within bounded intervals independently. The complexities of the algorithms

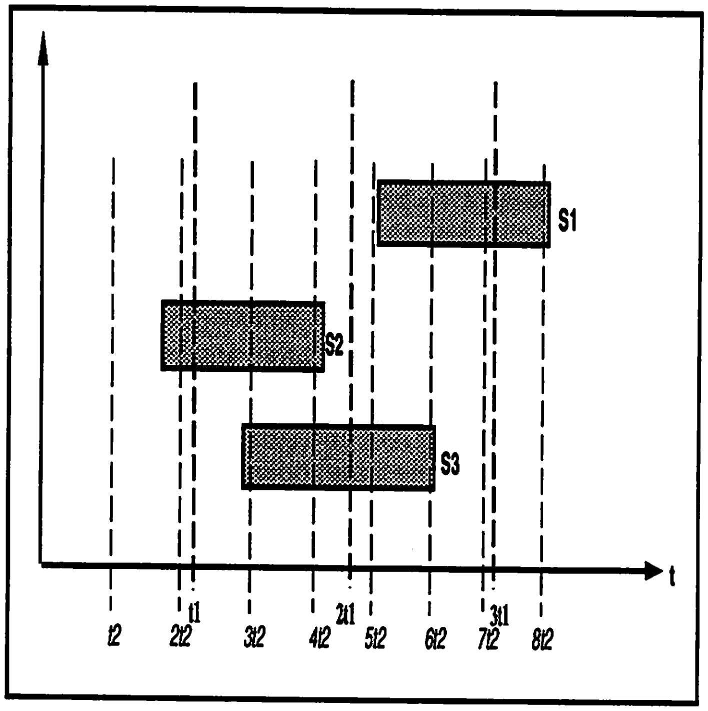

Figure 9: Delay Specific Enumeration

for the three delay models are the simplest for the first case and the most complex for the third case. In solving the mixed Boolean linear programming problem, a lower bound for the minimum cycle time is updated during each iteration so that the complexity of the problem is decreasing progressively. And the core computation of the problem is translated into the problems of tautology checking, test generation, redundancy check, and SAT.

References

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

[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. *Proc. of the ICCAD,* Nov. 1991.

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

[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.

[LBSV] W. Lam, R. Brayton, and A. Sangiovanni-Vincentelli. Exact delay computation with timed boolean function. *In preparation.*

[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.