Valid Clock Frequencies and Their Computation in Wavepipelined Circuits *

William K.C. Lam
Hewlett-Packard Co. Palo Alto, California
Robert K. Brayton Alberto L. Sangiovanni-Vincentelli
Department of EECS, University of California, Berkeley

Abstract

It is known that wavepipelined circuits offer high performance, because their maximum clock frequencies are limited only by the path delay differences of the circuits, as opposed to the longest path delays, and that precision in clock frequency is essential for proper operation. Using a new representation, Timed Boolean Function, we take into account circuit functionality in analyzing wavepipelined circuits and derive analytical expressions for valid clocking intervals in terms of topological, 2-vector, and single vector delays, both the longest and the shortest. These expressions subsume Cotten's lower bound. Further, these intervals eliminate the pessimism of long and short false paths and include effects of circuit parameters such as delay variations, clock skews, and setup and hold times of flip flops. Then, we study the problem of computing all exact valid clocking intervals and its computational complexity by demonstrating discontinuity and non-monotonicity of the harmonic number $H(\tau)$ (the number of valid simultaneous data waves allowed) as a function of the clock period τ. Finally, we propose algorithms to compute the exact valid intervals for a given set of harmonic numbers and demonstrate performance enhancement of balanced circuits from ISCAS benchmarks with gate delay variations.

*Supported by Fannie and John Hertz Foundation and SRC under contract 93-DC-008.

Auther Correspondence

1. William K.C. Lam (corresponding author)
   4968 Creekwood Dr.
   Fremont, CA 94555
   (510) 642-3433
   lamwill@ic.berkeley.edu

   William K.C. Lam

2. Robert K. Brayton
   Room 519c Cory Hall
   University of California
   Berkeley, CA 94720

3. Alberto L. Sangiovanni-Vincentelli
   Room 515 Cory Hall
   University of California
   Berkeley, CA 94720

Parts of this manuscript appeared in ICCAD '92.

Footnotes, affiliations

William K.C. Lam is with Hewlett-Packard Co. Palo Alto, California.

Robert K. Brayton, Alberto L. Sangiovanni-Vincentelli are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley.

Footnotes:

(1) This definition is a refined version of that in [14].

William K.C. Lam is supported by Fannie and John Hertz Foundation and SRC under contract 93-DC-008.

1 Introduction

The delay of a circuit is commonly used as a measure for the circuit's computational performance, i.e. how fast a circuit can compute its outputs. Thus, conventionally, input data are supplied to a circuit in intervals greater than or equal to the delay of the circuit. For example, in the pipelined combinational circuit shown in Figure 1 where blocks of combinational circuits are sandwiched between banks of flip flops, the clock period is chosen to be greater than or equal to the maximum path delay of the combinational logic. At this frequency, data in any logic block have enough time to propagate through and settle before new data enter. However, in wavepipelining mode, the circuit in Figure 1 will be clocked at a period less than the maximum path delay of a stage; thus a data wave is pumped into a stage before the preceding wave reaches the registers at the end of the stage. So wavepipelining circuits operate at higher speeds than conventional circuits, sometimes orders of magnitude higher. Since the clock period is shorter than the delay of a circuit, data from neighboring clock cycles co-exist in the circuit simultaneously, and they can interact to cause the circuit to compute incorrectly. For instance, if a long and a short path converge at a gate and the clock frequency is high enough to cause the present data on the short path to arrive at the gate earlier than the preceding data along the long path, invalid computation results. Hence wavepipelining circuits involve complex signal interactions in the temporal domain and their proper operations require precise timing analysis.

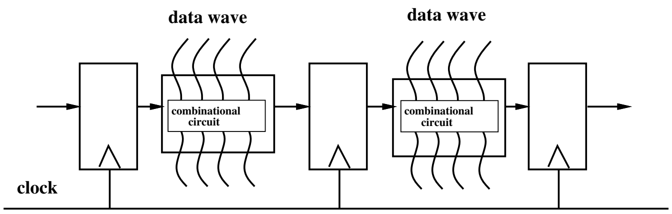

Figure 1: A Pipelined Circuit

Traditional studies on pipelining have been focusing on system and functional levels [9, 6], in which a functional block may perform a single dedicated or a selectable collection of computations, which are called static and dynamic pipelines, respectively. Pipelinings on the circuit level, usually restricted to static pipelines, are referred to as maximum rate pipelining [1] or wavepipelining. The main result from previous studies on combinational wavepipelining circuits is that the maximum clock frequency is proportional to the difference between the maximum and minimum path delays. Thus, much research on wavepipelining has been directed to balancing circuits' path lengths. Maximizing the clock frequency by inserting delays into paths and satisfying various timing constraints for proper operation have been formulated as a linear programming problem.

Previous work on the theoretical side involves formulating timing constraints for proper wavepipelining operation. Cotten [1] first published the concept, although wavepipelining circuits were built [16] prior to that. Ekroot [3] studied in depth the various timing constraints of wavepipelining. Sakallah et al [17] proposed a scheme to formulate various timing problems. Fishburn [4] studied minimization of clock periods while Joy and Ciesielski [8] extended Fishburn's work to wavepipelining. Gray, Liu, and Cavin [5] studied timing constraints on clocking periods. Lam, Brayton, and Sangiovanni-Vincentelli [14] derived exact valid clocking intervals using timed Boolean functions. On the implementation side, various wavepipelining circuits such as multipliers, counters, domino circuits, and Cache RAM's using ECL and CMOS technology have been fabricated. The first published wavepipelining circuit was built by Loomis [16] in 1966. Wong, DeMicheli, and Flynn [19, 20] used delay insertion and device fine-tuning techniques to balance path delays in ECL circuits. Joy and Ciesielski [7] studied layout and placement of wavapipelining circuits. Lien and Burleson [15] designed multipliers using wave-domino logic.

In this paper, we present a systematic algebraic method for analysis and design of wavepipelining circuits using Timed Boolean Function's (TBF's). With TBF's, *exact* results combining both logical and timing information can be derived rigorously, and effects such as clock skews, uncertainties in physical parameters, and timing constraints of flip flops can be incorporated readily.

2 Timed Boolean Functions

In timing research, the entire temporal behavior of circuits is studied; thus, a representation technique capturing temporal properties should be devised. In this section, we use TBF's as a representation formalism for timing analysis and study its properties in formulating circuits with both logical and timing functionalities.

DEFINITION 1 *Timed Boolean Functions*[1]

1. *A* **binary signal space** $B(t)$ *is a collection of mappings* $f : R \mapsto B$*, where* R *is the set of real numbers and* $B = \{0, 1\}$*.*

2. *A* **Timed Boolean Function (TBF)** *is any function with domain* $B^n(t)$ *and range* $B(t)$*. In this paper, we will focus only on the TBF's satisfying following properties.*

   TBF $F : B^n(t) \mapsto B(t)$ *satisfies the following properties.*

   - Identity. *The identity function* F *(i.e.* $F(v)(t) = v(t)$, $v(t) \in B(t)$*) is a TBF.*
   - Closure under Boolean operations. *If* $G : B^{n_1}(t) \mapsto B(t)$ *and* $H : B^{n_2}(t) \mapsto B(t)$ *are TBF's, then,* $\overline{G}, G \cdot H, G + H$ *are also TBF's.*
   - Closure under argument transformation. *If* $F(t)$ *is a TBF, then, for any function* $\phi : R^n \mapsto R$, $F(\phi)$ *is also a TBF.*
   - Closure under composition. *If*

     $$G = G(\ldots, x_i(g_{i1}), \ldots, x_i(g_{im}), \ldots), i = 1, \ldots, n_1, \tag{1}$$

     and $H = H(t)$ *are TBF's, where* $G : B^{n_1}(t) \mapsto B(t)$*,* $H : B^{n_2}(t) \mapsto B^{n_1}(t)$*, and* $g_{ij} : R^n \mapsto R$*, then*

     $$G \circ H = G(\ldots, H_i(g_{i1}), \ldots, H_i(g_{im}), \ldots) \tag{2}$$

     is also a TBF, where H_i *is the ith component of* H*.*

[1]This definition is a refined version of that in [12]

An element in a binary signal space can be regarded as a binary signal whose value changes with time, with domain being time and the range being the value of the signal at the time. Note that a TBF is a natural generalization of a Boolean function whose domain and range are restricted to constant mappings in $B(t)$, i.e. $B(t)$ is a set of binary variables whose values do not change with time. Multiple output TBF's are just vectors consisting of single output TBF's. Sometimes it is convenient to think of a TBF as a function from R^n to B^m analogous to looking at the outputs of a circuit with m outputs and $n-1$ parameters such as gate delays and the nth parameter being time. But a TBF is not defined as such because functions from R^n to B^m do not allow composition: compositions model circuits composed of sub-circuits.

DEFINITION 2 *A variable $x(\phi)$ in a TBF is called a* **TBF variable***, and $\phi : R^n \mapsto R$, the* **argument** *of the TBF variable.*

Example 1 *Function $f(a,b,c)(t) = a(t-k)b(t^2 - \log t - k) + \bar{c}(\sqrt{t+2+k^2})$ is a TBF and has TBF variables $a(t-k)$, $b(t^2 - \log t - k)$, $c(\sqrt{t+2+k^2})$; and the TBF variables have arguments of $t-k$, $t^2 - \log t - k$, and $\sqrt{t+2+k^2}$, respectively.*

3 Modeling Timing Behaviors

The functionalities of digital circuits can be regarded as transformations on input binary signals. As inputs propagate through a gate in a circuit, the gate delays the inputs according to the delay of the gate and produces a resulting signal via Boolean operations on the delayed inputs. To model a circuit's timing behavior, we first consider the modeling of several commonly used simple gates with TBF's. Modeling of complex gates can be done by decomposing the complex gates into these simple gates and deriving their TBF representations from those of the simple gates, as will be shown later.

1. *Gates characterized by a single delay for each input-output pair.* Here a gate's delay is modeled by delays of input-output pairs; and the delay of an input-output pair is assumed to be independent of logical values on other inputs. Gates with multiple outputs are decomposed into connections of gates with single outputs; hence, we will focus on gates with single output here.

   Example 2 *The complex gate in Figure 2(a) has three inputs; input x_i has a delay τ_i to the output, both rising and falling. This gate is modeled by the TBF:*

   $$y(t) = \bar{x}_1(t-\tau_1) + x_2(t-\tau_2) + x_3(t-\tau_3) \tag{3}$$

2. *Buffer with different rising and falling delays.* Here the delay of a buffer depends on whether the transition is rising or falling. We examine whether such phenomena can be regarded as the interaction of buffers with equal rising and falling delays. Let τ_r and τ_f be the rising and falling delays of a buffer, respectively. To model a buffer with different rising and falling

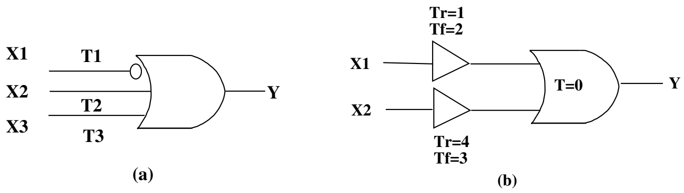

Figure 2: Modeling with TBF's

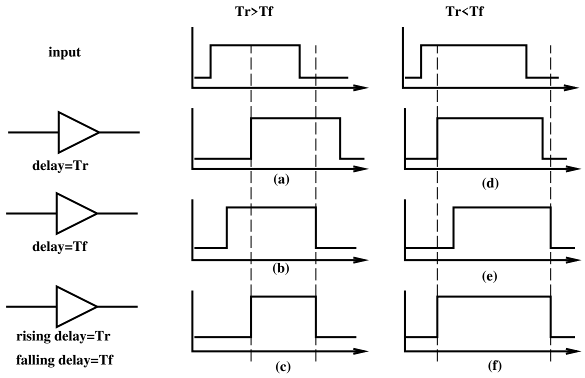

Figure 3: Modeling Rising and Falling Delays

delays, we consider two cases: $\tau_r > \tau_f$ and $\tau_r < \tau_f$. For $\tau_r > \tau_f$, we show the timing diagrams for both rising and falling transitions for a buffer with delay τ_r in Figure 3(a), a buffer with delay τ_f in Figure 3(b), and a buffer with rising delay τ_r and falling delay τ_f in Figure 3(c). It is seen that the signal in Figure 3(c) is the conjunction of those in (a) and (b); thus, the timing behavior of a buffer with different τ_r and τ_f is the conjunction of the timing behaviors of a buffer with delay τ_r and another with delay τ_f. Thus, the TBF for a buffer with rising delay τ_r and falling delay τ_f is (for $\tau_r > \tau_f$):

$$y(t) = x(t - \tau_r) \cdot x(t - \tau_f). \tag{4}$$

For the case $\tau_r < \tau_f$, the corresponding timing diagrams are shown in Figure 3(d), (e), (f). It is evident that the TBF for a such buffer is:

$$y(t) = x(t - \tau_r) + x(t - \tau_f). \tag{5}$$

3. *Gates with different rising and falling delays for each input-output pair.*

Example 3 *Consider an OR gate with inputs x_1 and x_2. Assume that input x_1 has a rising delay of 1 and a falling delay of 2, while input x_2 has a rising delay of 4 and a falling delay of 3. This OR gate can be modeled by the small circuit in Figure 2(b), The buffer modeling input x_1 has TBF $x_1(t-1) + x_1(t-2)$, because the falling delay is greater than the rising* *delay. The buffer modeling input 2 has TBF $x_2(t-4) \cdot x_2(t-3)$. Therefore, the OR gate has TBF*

$$y(t) = x_1(t-1) + x_1(t-2) + x_2(t-4) \cdot x_2(t-3). \tag{6}$$

A common problem in digital circuit design is pulse shrinkage or dilation. Pulse shrinkage (dilation) 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. Pulse shrinkage and dilation can cause problems, for example, violating setup and hold times of flip flops. Because TBF's can model gates with different rising and falling delays, pulse shrinkage and dilation can be detected.

4. *Edge triggered D flip flop with a clock of period P.* Let Q, D, d be the output, the data input, and the delay of the flip flop, respectively. Since the clock is periodic with period P, the output Q is the sample and hold version of the data input D. Lumping the delay of a flip flop to the data input, the TBF representing the flip flop is

$$Q(t) = D\left(\left\lfloor \frac{t-d}{P} \right\rfloor \cdot P\right) \tag{7}$$

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

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 negative and has duration P_1, the second part is positive and has duration P_2. Thus, $P_1 + P_2 = P$. The latch is represented by

$$Q(t) = D\left((1-\alpha)\left\lfloor \frac{t-d}{P} \right\rfloor P + \alpha \cdot (t-d)\right) \tag{8}$$

If $P_1 \geq P_2$, $\alpha = \left\lfloor \frac{(t-d) \mathrm{mod} P}{P_1} \right\rfloor$; and if $P_1 \leq P_2$, $\alpha = \left\lfloor \frac{(t-d) \mathrm{mod} P - P_1}{P_2} \right\rfloor + 1$. If $\alpha = 0$, $t-d$ falls in the negative phase of the clock; and if $\alpha = 1$, $t-d$ falls in the positive phase of the clock. Now let us see whether the above TBF indeed represent the desired behavior. If $\alpha = 0$, the latch's TBF becomes $Q(t) = D\left(\left\lfloor \frac{t-d}{P} \right\rfloor P\right)$, which gives the value latched in the previous cycle, as expected. If $\alpha = 1$, the latch's TBF become $Q(t) = D(t-d)$, which says the latch is in transparent model, as expected. (a mod b) is the remainder when a is divided by b, i.e. $a - \lfloor \frac{a}{b} \rfloor b, a > 0, b > 0$.

Example 4 *Let $P_1 = 2, P_2 = 1, d = 0$. So, the clock period is 3 and the times at which data are latched are $3n$. At $t = 16$, $\alpha = \lfloor \frac{16 \mathrm{mod} 3}{2} \rfloor = 0$; $Q(16) = D(\left\lfloor \frac{16}{3} \right\rfloor 3) = D(15)$. Because $\alpha = 0$, the latch is in "latched" mode; so the output is equal to the previously latched value, as calculated. At $t = 17$, $\alpha = \lfloor \frac{17 \mathrm{mod} 3}{2} \rfloor = 1$; $Q(17) = D(17)$. Because $\alpha = 1$, the latch is in "transparent" mode; so the output is equal to the current input value.*

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

4 Circuit Formulation

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 a 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 examples.

Example 5 *Assume the gate delays in the circuit of Figure 4 are the constants shown. The TBF at node f is obtained by composing the TBF's of the invertor and the buffers with that of the OR gate. Thus the TBF at f is*

$$f(t) = a(t-3) + \bar{a}(t-4) + b(t-2) \tag{9}$$

The TBF at node y is obtained by composing the TBF at f with that of the AND gate, giving:

$$\begin{aligned} y(t) = &\ a(t-1)b(t-1)f(t-1) \\ = &\ a(t-1)b(t-1)(a(t-4) + \bar{a}(t-5) + b(t-3)) \end{aligned} \tag{10}$$

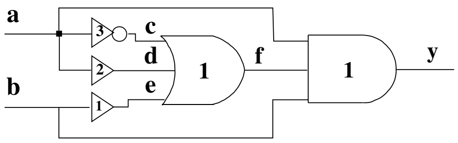

Figure 4: An Example Circuit for TBF

TBF's for synchronous circuits do not involve recursive equations because the feedbacks are accounted for by the feedbackless TBF's of flip flops and latches, as illustrated in the following example.

Example 6 *In Figure 5, the delay for each gate is shown inside the gate. First, we formulate the combinational part of the circuit with TBF's. Each gate is represented by a TBF, as follows.*

$$\begin{aligned} g(t) &= a(t) + b(t) \\ b(t) &= \bar{f}(t-2) \\ a(t) &= c(t)d(t)e(t) \\ c(t) &= f(t-1.5) \\ d(t) &= \bar{f}(t-4) \\ e(t) &= f(t-5) \end{aligned} \tag{11}$$

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

$$\begin{aligned} a(t) &= f(t-1.5)\bar{f}(t-4)f(t-5) \\ b(t) &= \bar{f}(t-2) \end{aligned} \tag{12}$$

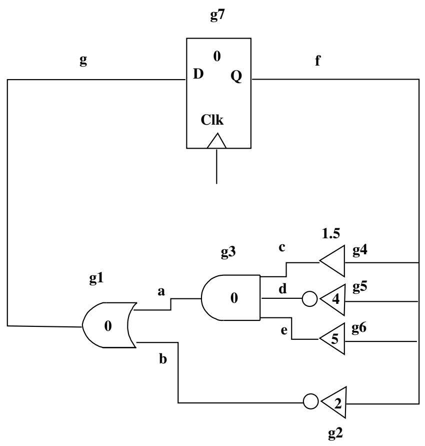

Figure 5: A Synchronous Sequential Circuit

Therefore,

$$g(t) = f(t - 1.5)\bar{f}(t - 4)f(t - 5) + \bar{f}(t - 2) \tag{13}$$

The TBF for the D flip flop with zero delay is

$$f(t) = g(\left\lfloor \frac{t}{\tau} \right\rfloor \tau) \tag{14}$$

where τ is the clock period of the synchronous sequential circuit.

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

$$g(t) = g(\left\lfloor \frac{t-1.5}{\tau} \right\rfloor \tau)\bar{g}(\left\lfloor \frac{t-4}{\tau} \right\rfloor \tau)g(\left\lfloor \frac{t-5}{\tau} \right\rfloor \tau) + \bar{g}(\left\lfloor \frac{t-2}{\tau} \right\rfloor \tau) \tag{15}$$

This equation represents the complete functionality and timing information of the synchronous sequential circuit in Figure 5.

4.1 General Formulations for Combinational and Synchronous Sequential Circuits

For combinational circuits where each gate is represented by a TBF having time argument of the form $t - d_i$, where d_i is a constant, the TBF for the whole circuit, obtained by composing the TBF's for the gates in the circuit, has only time arguments of the form $t - \sum d_i$. In particular, if d_i is the delay of the ith gate in a circuit, then $\sum d_i$ is the delay of a path from an input to an output. The general form of a TBF for a combinational circuit is

$$f(t, x_1, \ldots, x_n) = f(\tilde{x}_1(t - k_1), \ldots, \tilde{x}_p(t - k_p)) \tag{16}$$

$$k_i = \sum_j d_j \tag{17}$$

where k_i is the delay of path π_i and d_i is the delay of ith gate, p is the number of paths, $\tilde{x}_k = x_i$ if π_k starts at input x_i.

The TBF's for synchronous sequential circuits with clock period τ can be derived systematically as described below. Referring to Figure 6, v_i is a synchronized primary input, y_i and x_i are the input and output of the ith flip flop, and z_i is a primary output. Assume that all external inputs to the circuit are synchronized to the clock as shown in Figure 6. The TBF's for the combinational next state logic have the general form:

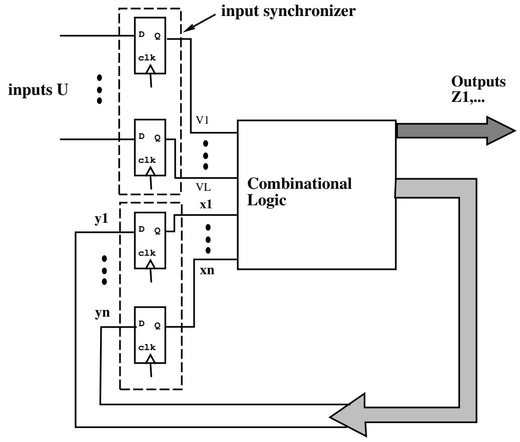

Figure 6: Block Diagram of a Synchronous Sequential Circuit

$$y_i(t) = f_i(x_1(t - d_{i1}), ..., x_p(t - d_{ip}), v_1(t - d_{iv_1}), ..., v_q(t - d_{iv_q})),\ i = 1, \ldots, n; \tag{18}$$

where p is the number of paths from input x_i to output y_j, q, the number of paths from input v_i to output y_j. We treat v_i's as states of the circuit, hence,

$$y_i(t) = f_i(x_1(t - d_{i1}), ..., x_s(t - d_{is})),\ i = 1, \ldots, n; \tag{19}$$

where d_{ij} is the delay of a path from the jth flip flop's output to the ith flip flop's data input. Incorporating the flip flops' TBF's, $y(t) = \left\lfloor \frac{t-d}{\tau} \right\rfloor \tau$, we obtain:

$$y_i(t) = f_i(y_1(\left\lfloor \frac{t - d_{i1} - d_{f_1}}{\tau} \right\rfloor \tau), ..., y_s(\left\lfloor \frac{t - d_{is} - d_{f_s}}{\tau} \right\rfloor \tau)),\ i = 1, \ldots, n; \tag{20}$$

$$y_i(t) = f_i(y_1(\left\lfloor \frac{t - k_{i1}}{\tau} \right\rfloor \tau), ..., y_s(\left\lfloor \frac{t - k_{is}}{\tau} \right\rfloor \tau)),\ i = 1, \ldots, n; \tag{21}$$

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

$$y_i(n\tau) = f_i(y_1(n\tau + \left\lfloor \frac{-k_{i1}}{\tau} \right\rfloor \tau), ..., y_s(n\tau + \left\lfloor \frac{-k_{is}}{\tau} \right\rfloor \tau)),\ i = 1, \ldots, n. \tag{22}$$

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

$$y_i(n) = f_i(y_1(n + \left\lfloor \frac{-k_{i1}}{\tau} \right\rfloor), ..., y_s(n + \left\lfloor \frac{-k_{is}}{\tau} \right\rfloor)), i = 1, \ldots, n. \tag{23}$$

The TBF's for the primary outputs z_i's are similarly derived.

5 Wavepipelining of Combinational Circuits

Figure 1 shows a block diagram of a pipelined combinational circuit. If the circuit is clocked at a period less than the maximum path delay of a stage, a data wave is pumped into the stage before the previous wave reaches the registers at the end of the stage; hence, multiple data waves co-exist within the combinational logic. If a long path and a short path converge at a gate and the clock frequency is high enough to cause the present data along the short path to arrive at the gate earlier than the preceding data along the long path, invalid computation results. This is the intuition that the difference of path delays determines the maximum clock frequency. Cotten [1] derived a lower bound on the minimum clock period, which is stated in a variant form as follows.

$$\tau \geq t_{setup} + t_{hold} + d^{max} - d^{min} \tag{24}$$

where t_{setup} and t_{hold} are the setup and hold times of registers, and d^{max} and d^{min} are the maximum and minimum delays of a pipeline stage.

Therefore, to maximize the clock frequency of a wavepipeline, the maximum path delay differences in the circuit should be minimized. But, after a circuit's delays are balanced, what are the frequencies that yield correct computation of the circuit, noting that Cotten gave only a lower bound? Some fundamental questions are:

1. Will the circuit operate properly at any clock period greater than the minimum, i.e., is the set of valid clock periods connected? If not, how can the valid intervals be found?

2. How do parametric uncertainties, e.g. delay variations, clock skews, and flip flop setup and hold times, affect the valid clocking intervals?

3. For a given clock frequency, how do we know what function the circuit is computing?

Question 1 and 2 have been studied by various authors [3, 4, 8, 19, 5, 15], in which valid clocking periods and their relations to various circuit parameters, such as delay variations, clock skews, and setup and hold times, have been formulated. However, the valid clocking intervals derived in the previous methods did not consider circuit functionality; thus, these clocking intervals may be pessimistic. In this paper, we use timed Boolean functions to formulate valid clocking intervals. In this approach, we derive the *exact* valid clocking intervals by incorporating the circuit's functionality, i.e. false paths are taken into account. This approach provides simple algebraic formulas relating a circuit's *true* delays and various parameters such as clock skew, setup and hold times, which allow the designer to examine the effects of these parameters on the clock intervals. Finally, with TBF's, question 3 is answered rather easily.

5.1 Valid Clocking Intervals

In this section, we derive the range of clock frequencies for a pipelined circuit at which the circuit computes the designed functionality. We first consider a single stage pipelined circuit shown in Figure 7; results for single stage pipelined circuits extend readily to circuits with multiple stages, as we will see in a later section.

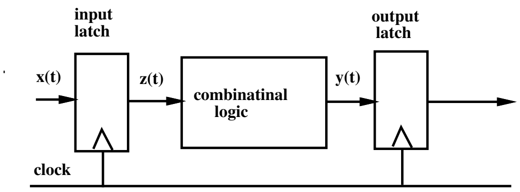

Figure 7: A Stage of a Pipelined Circuit

In a circuit, we call the longest and the shortest path lengths **maximum topological delay**, denoted by L^{max}, and **minimum topological delay**, denoted by L^{min}, respectively. For simplicity, we assume the flip flops have zero delays, setup, and hold times; all gate delays are fixed; and there is no clock skew. We will see shortly that these effects can be incorporated quite easily.

Let $y_j(t)$ be the input signal of the jth output flip flop, and $x_i(t)$ and $z_i(t)$ be the input and output signals of the ith input flip flop, as shown in Figure 7. The TBF for the combinational logic has the form

$$y_j(t) = f(..., z_i(t - k_{ij}), ...) \tag{25}$$

where k_{ij} is a delay from the output of the ith input flip flop to the input of the jth output flip flop. Thus, $L^{min} \leq k_{ij} \leq L^{max}$.

An input flip flop with zero delay has the TBF,

$$z_i = x_i(\left\lfloor \frac{t}{\tau} \right\rfloor \tau) \tag{26}$$

Eliminating z_i yields

$$y_j(t) = f(..., x_i(\left\lfloor \frac{t - k_{ij}}{\tau} \right\rfloor \tau), ...) \tag{27}$$

Since edge triggered flip flops are assumed, only data at the times of clock transitions are of interest; thus, sampling at period τ gives

$$y_j(n\tau) = f(..., x_i(n\tau + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor \tau), ...) \tag{28}$$

Normalizing to τ yields

$$y_j(n) = f(..., x_i(n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor), ...) \tag{29}$$

where, $y_j(n)$ is the value of y_j at the nth clock period.

We want to choose the clock period τ such that the function at the inputs of the output flip flops are as designed. In conventional pipelines, $\tau \geq L^{max}$; so, all the arguments of x_i, i.e. $n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor$, are equal to $n-1$. Furthermore, if τ is chosen so that all arguments of x_i's are equal to $n - H$, for some positive integer H, the circuit still computes the same function, except delayed by $H\tau$. Conversely, even if τ is so chosen that the arguments of some variables $x_i(n)$ are different, $y_i(n)$ may still be a delayed version of the original function because the variables with different arguments may be "redundant". For example, $y_1(n) = x_1(n-3)\bar{x}_1(n-2) + x_1(n-3)$ is a delayed version of the original function $y_1(n) = x_1(n-1)$ even though not all arguments of the variables are the same, because $\bar{x}_1(n-2)$ is redundant. Therefore, the circuit of $y_i(n)$ computes correctly if and only if all irredundant variables, or the minimum support, of $y_i(n)$ have the same argument. The case where some irredundant variables have different arguments corresponds to interference or collision of data from different clock cycles along true paths. If data collisions occur only on false paths, then all irredundant variables of $y_i(n)$ have the same argument while some redundant variables have different arguments. Therefore, we define a valid clock period τ as one at which the circuit computes a delayed version of the original function ($H = 1$).

DEFINITION 3

1. *Let $f(..., x_i, ...)$ be a TBF and x_i be a TBF variable. The* **cofactor** *of $f(..., x_i, ...)$ with respect to x_i, denoted by f_{x_i}, is $f(..., x_i, ...)$ evaluated at $x_i = 1$. Similarly, the cofactor with respect to $\bar{x}_i$ is the function evaluated at $x_i = 0$.*
2. *The* **minimum support** *of TBF $f(..., x_i, ...)$ is $\{x_i : f_{x_i} \neq f_{\bar{x}_i}\}$, i.e. the set of variables whose changes of value affect the value of the function.*
3. *Let $f[n-H]$ be derived from $f(\ldots, x_i(n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor), \ldots)$ by replacing the time argument of each variable with $n - H$, i.e. replace $n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor$ by $n - H$. τ is a* **valid clock period** *if $f(\ldots, x_i(n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor), \ldots) = f[n-H]$ for some H. H is called the* **harmonic number** *at τ.*[2]

Note that in $f(\ldots, x_i(n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor), \ldots)$ if clock period τ is such that $n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor = n - H$, for some fixed constant H and for all x_i in the minimum support of f, then τ is a valid clock period. Valid clocking intervals consist of all valid clock periods. A sufficient condition for valid clocking intervals is that all time arguments of $f(\ldots, x_i(n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor), \ldots)$ are equal.

THEOREM 1 *The clock period τ for a pipelined circuit with TBF $f(.)$ is valid if for some integer H and for all stages in the pipeline,*

$$\frac{L^{max}}{H} \leq \tau < \frac{L^{min}}{H-1}. \tag{30}$$

The computed function at τ in the above interval is $f[n-H]$.

[2]The reason we call H the harmonic number rather than latency, the conventional terminology, is that the term harmonic number provides a natural generalization to the concept of fundamental mode in the Hoffman model of asynchronous systems. When an asynchronous circuit operates at the fundamental mode in the Hoffman model, all transient must have settled before a new input is allowed. This condition corresponds to harmonic number of 1. And the natural name for operation at harmonic number equal to 1 is fundamental mode. Hence, when the fundamental mode in Hoffman model is violated, the circuit operates at a higher harmonic mode.

Proof. Because

$$\frac{L^{max}}{H} \leq \tau < \frac{L^{min}}{H-1} \tag{31}$$

then

$$\frac{k_{ij}}{H} \leq \tau < \frac{k_{ij}}{H-1}, \ \forall i, j \tag{32}$$

equivalently,

$$-H \leq \frac{-k_{ij}}{\tau} < -H+1, \ \forall i, j \tag{33}$$

Hence,

$$n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor = n - H, \ \forall i, j \tag{34}$$

Therefore,

$$y_j(n) = f(..., x_i(n + \left\lfloor \frac{-k_{ij}}{\tau} \right\rfloor), ...) = f[n-H] \tag{35}$$

■

This theorem illustrates how the longest path L^{max} and the shortest path L^{min} affect the valid clock frequency. The theorem implies that the closer the L^{min} and L^{max} the wider the valid intervals. The largest interval is achieved when $L^{max} = L^{min}$, consistent with intuition.

5.2 Multiple-stage Pipelines

For a pipeline with multiple stages, the valid clock interval is the intersection of the valid clock intervals of all stages; that is, if L_i^{max} and L_i^{min} are the maximum and minimum path delays of the ith stage, respectively, then the valid interval for the multiple stage pipeline is:

$$\bigcap_i [\frac{L_i^{max}}{H}, \frac{L_i^{min}}{H-1}); \tag{36}$$

equivalently,

$$[\frac{\max_i L_i^{max}}{H}, \frac{\min_i L_i^{min}}{H-1}); \tag{37}$$

or

$$[\frac{L_*^{max}}{H}, \frac{L_*^{min}}{H-1}), \tag{38}$$

where L_*^{max} and L_*^{min} are the maximum and minimum path delay among all the stages, respectively. Therefore, for pipelines with multiple stages, L^{max} and L^{min} are simply the maximum and minimum topological delays of all stages, respectively.

Since the intervals given by this theorem do not take into account logical functionalities and false paths of circuits, they are conservative and may be narrower than the exact valid intervals.

5.3 Physical Interpretation of H

In conventional pipelining, the clock period is greater than the longest topological delay, $H = 1$ and the number of data waves within a combinational circuit at clock transitions is 1. For $H > 1$, H corresponds to the number of data waves along true paths at clock transitions. For example, in the circuit of section 6.1.2, at $\tau = 1.5$ $H = 1$: there is only one data wave on the true paths while all false paths have two data waves. Therefore, H is a measure of the degree of wavepipelining; the larger the H, the more wavepipelining there is.

For a given clock period τ, we can decide whether τ is valid for a circuit by substituting τ into the circuit's TBF. If the resulting TBF is $f[n-H]$, τ is valid, otherwise invalid. The resulting TBF, whether at a valid τ or not, is the function of the circuit computing at τ. Conversely, for a given H, or degree of wavepipelining, we can determine whether there is a corresponding clocking interval by applying Theorem 1; if the interval $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$ is not empty, there is a valid clocking interval; otherwise, there still may be one but a definite answer requires taking into account logic functionality of the circuit, e.g. false paths, which we will do in Section 5.7.

5.4 Disjointedness of Valid Clock Intervals

Consider two nonempty adjacent intervals given by Theorem 1:

$$\frac{L^{max}}{H} \leq \tau < \frac{L^{min}}{H-1} \tag{39}$$

and

$$\frac{L^{max}}{H-1} \leq \tau < \frac{L^{min}}{H-2} \tag{40}$$

with harmonic number H and $H-1$ respectively. Since $L^{min} \leq L^{max}$, $\frac{L^{min}}{H-1} \leq \frac{L^{max}}{H-1}$; hence, the intervals are disjoint if $L^{min} < L^{max}$. Although the exact valid intervals for some circuits may still be connected, it is easy to construct circuits (e.g. with no false paths) all of whose exact valid intervals are given by Theorem 1. Because of the disjointedness of the valid clocking intervals, knowledge of only the minimum valid clock period is insufficient to guarantee proper pipeline operation: a clock period greater than the minimum achievable period may fall outside any of the valid intervals.

Consider the valid intervals

$$[L^{max}, \infty), [\frac{L^{max}}{2}, L^{min}), [\frac{L^{max}}{3}, \frac{L_{min}}{2}), \ldots \tag{41}$$

on the τ-axis obtained for $H = 1, 2, 3, \ldots$ The widths of the intervals, $\frac{L^{min}}{H-1} - \frac{L^{max}}{H}$, are narrower at larger H and higher clock frequencies, and become zero when the lower bound exceeds the upper bound, i.e.

$$\frac{L^{max}}{H} \geq \frac{L^{min}}{H-1} \tag{42}$$

The rightmost interval $[L^{max}, \infty)$ $(H = 1)$ corresponds the conventional clocking scheme. The intervals closer to zero correspond to faster clock rates, higher computation speeds, higher degree of wavepipelining, more latency; and the intervals are narrower, meaning more precise delay control is required for higher speed computation.

5.5 Effect of Physical Uncertainties

Here we discuss how delay variations, clock skew, flip flop setup and hold times can be incorporated in the above theory.

1. Delay variation. Delay variations of gates among circuits of the same design are caused by manufacturing fluctuations, and are usually modeled by bounded intervals, e.g. the ith gate delay varies within a bounded interval $[d_i^{min}, d_i^{max}]$. This effect is included by replacing each d_i in L^{max} with d_i^{max}. Similarly, each d_i in L^{min} is replaced by d_i^{min}. That is,

$$L^{max} = \max_{\pi} \sum_{i \in \pi} d_i^{max}, \; L^{min} = \min_{\pi} \sum_{i \in \pi} d_i^{min}. \tag{43}$$

   where π is a path consisting of gates with delays d_i's.

2. Clock skew. Clock skew can be modeled by buffers at the data inputs or outputs of flip flops. If the clock of an input flip flop has a skew Δ, then the data from the output of this input flip flop will be delayed by Δ; thus, the clock skew can be modeled by a buffer with delay Δ at the data output of the flip flop. Similarly, if the clock of an output flip flop has a skew Δ, then this clock skew can be modeled with a buffer with delay $-\Delta$ at the data input of the flip flop.

3. Setup and hold times. Setup time, t_{setup}, means that the data must arrive at the flip flops earlier than the clock by at least t_{setup}, which is equivalent to increasing the longest path by an amount t_{setup}. Thus,

$$L^{max} \rightarrow L^{max} + t_{setup} \tag{44}$$

   Hold time, t_{hold}, means that the data must persist for at least t_{hold}, which is equivalent to having the shortest path shorter by t_{hold}. Thus,

$$L^{min} \rightarrow L^{min} - t_{hold} \tag{45}$$

All the effects of these parameters can be formally included in the proof of Theorem 1. Note that all these physical uncertainties, except for the clock skews on the input flip flop, shrink the valid clocking intervals.

Example 7 *Suppose we wavepipeline the circuit in Figure 8. Assume the gate delays shown are the maximum delays and the minimum delays are 90% of the maxima. Further the setup and hold times of the flip flops are* 0.1 *units of time and the clock skew of the output flip flop is* 0.3 *units of time. From the circuit, the maximum topological delay is* 5 *and the minimum topological delay is* $90\% \times 4 = 3.6$. *Thus,* $L^{max} = 5 + 0.1 - 0.3 = 4.8$ *and* $L^{min} = 3.6 - 0.1 - 0.3 = 3.2$. *According to*

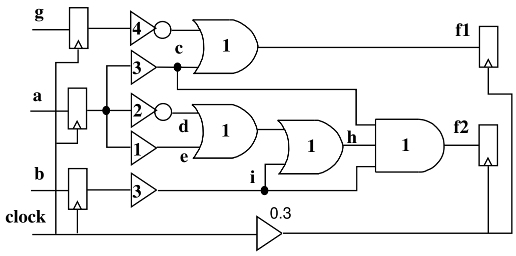

Figure 8: A Combinational Circuit for Wavepipelining

Theorem 1, the valid clocking intervals are: $[4.8, \infty)$, $[2.4, 3.2)$. *The first interval corresponds to the conventional clocking scheme, no wavepipelining. The second interval has a latency of* 1, $H = 2$.

Now consider the multiple stage pipeline in Figure 9, assuming the same flip flop setup and hold times. Consider the first stage. $L_1^{max} = 2.6$ *and* $L_1^{min} = 2.0$. *All the input flip flops in this stage have no clock skew, while one output flip flop has a clock skew of* 0.3 *which reduces the effective length of the longest paths from* 2.6 *to* 2.3. *Since the effective length of the longest paths (2.3) is less than the second longest path whose effective length is* 2.5, $L_1^{max} = 2.5$. $L_1^{min} = 2.0$. *Incorporating the setup and hold time,* $L_1^{max} = 2.6$ *and* $L_1^{min} = 1.9$.

For the second stage, one input flip flop and both output flip flops have skews of 0.3. *Incorporating the skews and setup and hold times,* $L_2^{max} = 2.4$ *and* $L_2^{min} = 1.6$. *Therefore,* $L_*^{max} = 2.6$ *and* $L_*^{min} = 1.6$, *and the valid intervals are* $[2.6, \infty)$ *and* $[1.3, 1.6)$.

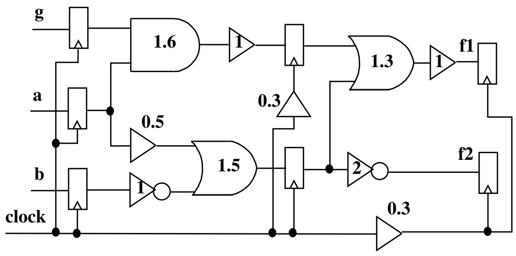

Figure 9: A Multiple Stage Wavepipelining Circuit

5.6 Relation to Cotten's Lower Bound

We show that Theorem 1 is more general than Cotten's lower bound by deriving the latter from the former. In Theorem 1, the minimum clock period in a valid interval is equal to its lower bound, $\frac{L^{max}}{H}$. Therefore, the overall minimum clock period is $\frac{L^{max}}{\max H}$. An upper bound on H is the maximum H such that $\frac{L^{max}}{H} \leq \frac{L^{min}}{H-1}$, equivalently, $\frac{L^{max}}{H} \geq L^{max} - L^{min}$. Thus, a lower bound on the clocking period is $L^{max} - L^{min}$. Incorporating the setup and hold times as in Section 5.5, we get the Cotten lower bound:

$$\tau \geq t_{setup} + t_{hold} + d^{max} - d^{min} \tag{46}$$

5.7 Effect of False Paths

Clocking in the intervals given by Theorem 1 will guarantee proper operations of circuits. Theorem 1 uses only path length (i.e. topological delays) information and does not consider logical functionalities of circuits. Considering the logical functionality will further widen the valid intervals. In timing analysis, the effect of logical functionalities are usually expressed with the notion of false paths and true delays. Here we relate the notion of true delays to our theory of valid intervals. Theorem 1 is tight if both the shortest and longest path delays are the true minimum and maximum delays, and is conservative otherwise. There are several definitions for the maximum and the minimum true delays, for example, floating delay, viability delay, transition delay, and delays by sequences of vectors. According to the general circuit delay model introduced in [10], these delays can be classified into two main types: 2-vector delay and delay by sequences of vectors. In this section, we relate these two types of delays to clocking intervals. The delays are those of the combinational logic in the wavepipelining circuits. If there are false paths, using 2-vector delays or delays by sequences of vectors will usually give wider valid clock intervals than using just topological delays; of course, topological delays are much easier to compute.

5.7.1 2-vector Delays

In 2-vector delay, a pair of vectors are applied to a circuit with the first vector applied at $t = -\infty$ and the second vector applied at $t = 0$. The delays of the gates in the circuit are modeled by bounded intervals. The maximum 2-vector delay of the circuit, denoted by D_2^{max} is the time of the latest output transition. Because inputs travel along paths to outputs, output transitions cannot be delayed by more than the length of the longest path; thus, $D_2^{max} \leq L^{max}$. If the first vector applied to input x_i is $x_i(0^-)$ and the second vector is $x_i(0^+)$, then TBF variable $x_i(t) = x_i(0^-)$, if $t < 0$ and $x_i(t) = x_i(0^+)$, if $t > 0$. Under this input condition, D_2^{max} is the minimum value such that:

$$f(..., x_i(t - k_i), ...) = f(..., x_i(0^+), ...), \forall t \geq D_2^{max}. \tag{47}$$

The setting for minimum 2-vector delay, denoted by D_2^{min} is the same as that for maximum 2-vector delay, and D_2^{min} is the time of the earliest output transition. Again, the length of the shortest path is the earliest time inputs can arrive at outputs; thus, $D_2^{min} \geq L^{min}$. Although wavepipelined inputs arrive in periodic sequences instead of in pairs as in the 2-vector delay scenario, 2-vector delays can be used as upper bounds on the valid clocking intervals if the wavepipelined inputs arrive at large enough separations so that the circuit sees at most two inputs at any time.

If the longest true paths are shorter than the longest topological paths and/or the shortest true paths are longer than the shortest topological paths, the valid clocking intervals can be widened. Since techniques for computing the 2-vector longest and shortest delays (or their upper and lower bounds) are available, e.g. [2, 12], it is advantageous to express the valid clocking intervals in terms of these delays. Assume the interval based on topological delays

$$\frac{L^{max}}{H} \leq \tau < \frac{L^{min}}{H-1} \tag{48}$$

is not null. Then the following theorem states the above valid clock interval can be widened according to the maximum and minimum 2-vector delays.

THEOREM 2 *If the interval* $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$ *is non-empty, the widened valid clocking interval is*

$$\max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}) \leq \tau < \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2}),\ for\ H > 1 \tag{49}$$

and

$$\max(D_2^{max}, \frac{L^{max}}{2}) \leq \tau,\ for\ H = 1 \tag{50}$$

The computed function at τ *in the above interval is* $f[n-H]$.

Proof. Let $f(..., x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor), ...)$ be a TBF of a stage of the pipeline, where k_i is a delay from an input to an output. We first consider $H > 1$. Because $D_2^{max} \leq L^{max}$,

$$\max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}) \leq \frac{L^{max}}{H}. \tag{51}$$

Similarly,

$$\frac{L^{min}}{H-1} \leq \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2}). \tag{52}$$

Therefore,

$$[\frac{L^{max}}{H}, \frac{L^{min}}{H-2}) \subseteq [\max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}), \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2})). \tag{53}$$

Hence, we can split the interval in the theorem into two at point $\frac{L^{max}}{H}$ and consider τ in the first sub-interval,

$$\frac{L^{max}}{H} \leq \tau < \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2}),\ H > 1 \tag{54}$$

This interval is not null, because $\frac{L^{max}}{H} < \frac{L^{min}}{H-1} < \frac{L^{min}}{H-2}$, and $\frac{L^{min}}{H-1} \leq \frac{D_2^{min}}{H-1}$. This interval implies that

$$(H-2)\tau < L^{min} \leq k_i \leq L^{max} \leq H\tau \tag{55}$$

Therefore,

$$\left\lfloor \frac{-k_i}{\tau} \right\rfloor = -H + 1 \text{ or } -H \tag{56}$$

So the arguments of the variables in $f(..., x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor), ...)$ are either $n-H+1$ or $n-H$. Specifically,

$$\left\lfloor \frac{-k_i}{\tau} \right\rfloor = \begin{cases} -H & \text{if } k_i > (H-1)\cdot\tau \\ -H+1 & \text{if } k_i \leq (H-1)\cdot\tau \end{cases} \tag{57}$$

Thus, the variables with $k_i \leq (H-1)\cdot\tau$ have time arguments $n-H+1$, while others have $n-H$.

Now compare the function $f(..., x_i(n+\left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...)$ with $f(..., x_i((H-1)\cdot\tau - k_i), ...)$. By defining $x_i(0^-) = x_i(n-H)$ and $x_i(0^+) = x_i(n-H+1)$, so

$$x_i((H-1)\cdot\tau - k_i) = x_i(0^-) \Leftrightarrow x_i(n+\left\lfloor\frac{-k_i}{\tau}\right\rfloor) = x_i(n-H) \tag{58}$$

and

$$x((H-1)\cdot\tau - k_i) = x(0^+) \Leftrightarrow x_i(n+\left\lfloor\frac{-k_i}{\tau}\right\rfloor) = x(n-H+1) \tag{59}$$

Because

$$\tau < \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2}) \leq \frac{D_2^{min}}{H-1} \tag{60}$$

then

$$(H-1)\cdot\tau < D_2^{min} \tag{61}$$

By the definition of D_2^{min}, for all $(H-1)\tau \leq D_2^{min}$,

$$f(..., x_i((H-1)\cdot\tau - k_i), ...) = f(..., x_i(0^-), ...) \tag{62}$$

i.e., the variables $x_i(0^+)$ are not in the minimum support of

$$f(..., x_i((H-1)\cdot\tau - k_i), ...); \tag{63}$$

thus, $x(n-H+1)$'s are not in the minimum support of

$$f(..., x_i(n+\left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...). \tag{64}$$

Therefore,

$$f(..., x_i(n+\left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...) = f(..., x_i(n-H), ...) \tag{65}$$

So, for τ in the above interval, the function is $f[n-H]$. Similarly for the second interval,

$$\max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}) \leq \tau \leq \frac{L^{max}}{H} \tag{66}$$

Since

$$\frac{L^{max}}{H+1} \leq \max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}) \leq \tau \tag{67}$$

then,

$$k_i \leq L^{max} \leq (H+1)\tau \tag{68}$$

From the condition $\frac{L^{max}}{H} < \frac{L^{min}}{H-1}$, and the upper bound of this sub-interval becomes,

$$\tau \leq \frac{L^{max}}{H} < \frac{L^{min}}{H-1} \tag{69}$$

Thus,

$$(H-1)\tau < L^{min}; \tag{70}$$

hence

$$(H-1)\tau < k_i \leq (H+1)\tau \tag{71}$$

or $\left\lfloor \frac{-k_i}{\tau} \right\rfloor = -H$ or $-H-1$. Specifically,

$$\left\lfloor \frac{-k_i}{\tau} \right\rfloor = \begin{cases} -H-1 & \text{if } k_i > H\tau \\ -H & \text{if } k_i \leq H\tau \end{cases} \tag{72}$$

Now compare the function $f(..., x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor), ...)$ with $f(..., x_i(H\tau - k_i), ...)$. By defining $x(0^-) = x(n-H-1))$ and $x(0^+) = x(n-H))$, so

$$x(H\tau - k_i) = x(0^-) \Leftrightarrow x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor) = x(n-H-1) \tag{73}$$

and

$$x(H\tau - k_i) = x(0^+) \Leftrightarrow x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor) = x(n-H). \tag{74}$$

Since

$$\frac{D_2^{max}}{H} \leq \max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}) \leq \tau \tag{75}$$

then,

$$H\tau \geq D_2^{max} \tag{76}$$

And by the definition of D_2^{max}, for all $H\tau \geq D_2^{max}$,

$$f(..., x_i(H\tau - k_i), ...) = f(..., x_i(0^+), ...). \tag{77}$$

Therefore,

$$f(..., x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor), ...) = f(..., x_i(n-H), ...) \tag{78}$$

Putting together the two intervals, we have shown that if

$$\max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}) \leq \tau < \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2}) \tag{79}$$

then

$$f(..., x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor), ...) = f(..., x_i(n-H), ...) \tag{80}$$

The interval for $H = 1$ is just a special case of the inteval just proved, which can also be found in [11]. ■

The terms $\frac{L^{max}}{H+1}$, $\frac{L^{min}}{H-2}$ and the functions $max()$, $min()$ in the theorem restrict the valid intervals so that circuits see only two input vectors at any time, keeping the 2-vector assumption in the computation of 2-vector delays valid. Note that the intervals in the theorem are indeed at least as wide as those original intervals, i.e. the use of 2-vector delays widen valid clock intervals:

$$[\frac{L^{max}}{H}, \frac{L^{min}}{H-1}) \subseteq [\max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}), \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2})), \ for\ H > 1 \tag{81}$$

and

$$[L^{max}, \infty) \subseteq [\max(D_2^{max}, \frac{L^{max}}{2}), \infty), \ for\ H = 1. \tag{82}$$

5.7.2 Delays by Sequences of Vectors

Since floating or viability delays, and delays by sequences of vectors are equivalent in practice [10], we will focus on the delays by sequences of vectors in this section. The setting for the delay by sequences of vectors for a circuit is as follows. Assume the circuit is initially settled. A sequence of input vectors is applied with the last vector applied at $t = 0$. The arrival time of the last output transition is the maximum delay by sequences of vectors, denoted by D_∞^{max}. Because L^{max} is the longest path length, input signals cannot be delayed by more than L^{max}; hence, $D_\infty^{max} \leq L^{max}$. Let $x_i(0^+)$ denote the last vector of a sequence of input vectors applied to input x_i. TBF variable $x_i(t) = x_i(0^+)$ for $t > 0$; $x_i(t)$ can be any value for $t < 0$ because the vectors and their temporal separations of an input sequence are arbitrary for $t < 0$. Then, under this input condition, D_∞^{max} is the minimum value such that:

$$f(..., x_i(t), ...) = f(..., x_i(0^+), ...), \forall t \geq D_\infty^{max}. \tag{83}$$

Similarly, the setting of the minimum delay by sequences of vector, D_∞^{min}, is: a sequence of vectors is applied with the first vector applied at $t = 0$, the arrival time of the first output transition is D_∞^{min}. Thus, $D_\infty^{min} \geq L^{min}$. Both D_∞^{max} and D_∞^{min} can be computed efficiently.

The following theorem relates these delays to clocking intervals, whose relations resemble interestingly those related to path lengths in Theorem 1, rather than the form of Theorem 2.

THEOREM 3 *If the interval $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$ is non-empty, the widened valid clocking interval is*

$$\frac{D_\infty^{max}}{H} \leq \tau < \frac{D_\infty^{min}}{H-1} \tag{84}$$

The computed function at τ in the above interval is $f[n-H]$.

Proof. Observe that $D_\infty^{max} \leq L^{max}$ and $D_\infty^{min} \geq L^{min}$. If the interval $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$ is non-empty, the interval $[\frac{D_\infty^{max}}{H}, \frac{D_\infty^{min}}{H-1})$, can be split into three sub-intervals: $[\frac{D_\infty^{max}}{H}, \frac{L^{max}}{H}]$, $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$, and $[\frac{L^{min}}{H-1}, \frac{D_\infty^{min}}{H-1})$, which are all non-empty if $D_\infty^{max} < L^{max}$ and $D_\infty^{min} > L^{min}$.

Obviously, the second interval is valid. We need to show the first and the third intervals are also valid. Let $f(\ldots, x(n + \left\lfloor \frac{-k}{\tau} \right\rfloor), \ldots)$ be a TBF of the circuit. Assuming $D_\infty^{min} > L^{min}$, we want to show that for $\tau \in [\frac{L^{min}}{H-1}, \frac{D_\infty^{min}}{H-1})$,

$$f(\ldots, x(n + \left\lfloor \frac{-k}{\tau} \right\rfloor), \ldots) = f[n-H] \tag{85}$$

Consider the TBF variables such that $x(n+\left\lfloor\frac{-k}{\tau}\right\rfloor) \neq x(n-H)$. Since $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$ is non-empty, $\frac{L^{min}}{H-1} > \frac{L^{max}}{H}$, equivalently, $\frac{L^{max}}{L^{min}} < \frac{H}{H-1}$. Further, $\tau \in [\frac{L^{min}}{H-1}, \frac{D_\infty^{min}}{H-1})$ implies $\frac{1}{\tau} \leq \frac{H-1}{L^{min}}$. Together with $k \leq L^{max}$, we have:

$$-\frac{k}{\tau} \geq (\frac{L^{max}}{L^{min}})(1-H) > -H. \tag{86}$$

Thus,

$$\lfloor -\frac{k}{\tau} \rfloor \geq -H. \tag{87}$$

Now if $x(n+\left\lfloor\frac{-k}{\tau}\right\rfloor) \neq x(n-H)$, i.e. $n+\left\lfloor\frac{-k}{\tau}\right\rfloor \neq n-H$, then $n+\left\lfloor\frac{-k}{\tau}\right\rfloor \geq n-H+1$; equivalently, $(H-1)\tau \geq k$. Therefore, these variables correspond to the variables with $t \geq k$ in $f(\ldots, x(t-k), \ldots)$, where $t = (H-1)\tau$. The interval $[\frac{L^{min}}{H-1}, \frac{D_\infty^{min}}{H-1})$ implies $L^{min} \leq (H-1)\tau < D_\infty^{min}$; or $L^{min} \leq t < D_\infty^{min}$. By the definition of D_∞^{min}, the variables with $t \geq k$ are not in the minimum support of $f(\ldots, x(t-k), \ldots)$, if $t < D_\infty^{min}$. Therefore, the variables such that $x(n+\left\lfloor\frac{-k}{\tau}\right\rfloor) \neq x(n-H)$ are not in the minimum support of $f(\ldots, x(n+\left\lfloor\frac{-k}{\tau}\right\rfloor), \ldots)$. Hence, $f(\ldots, x(n+\left\lfloor\frac{-k}{\tau}\right\rfloor), \ldots) = f[n-H]$. The interval $[\frac{L^{min}}{H-1}, \frac{D_\infty^{min}}{H-1})$ is valid.

The proof for the interval $[\frac{D_\infty^{max}}{H}, \frac{L^{max}}{H}]$ is analogous. ■

Example 8 *Consider the circuit in Figure 10. Assume that the gates' delays are fixed as shown, setup and hold times of flip flops are zero, and there is no clock skew. Then, $L^{max} = 5$ and $L^{min} = 3$. If functionality is not taken into account, by Theorem 1, the valid clocking intervals are $[5, \infty)$ and $[2.5, 3)$; With functionality considered, long false paths are discovered and wider clocking intervals are found. In this example, under the delay by sequences of vectors criterion, path {a,d,h,f2} is a long false path and there is no short false path; thus, $D_\infty^{max} = 4$ and $D_\infty^{min} = 3$. Under the 2-vector delay criterion, path {a,d,h,f2} and {a,e,h,f2} are long false paths and there is no short false path; thus, D_2^{max} and D_2^{min} are 3. By Theorem 3, the valid clocking intervals using delays by sequences of vectors are $[4, \infty)$, $[2, 3)$. Using 2-vector delays, we get still wider valid intervals. The slackness of Theorem 3 comes from the conservative use of delay by sequences of vector as delays for wavepipelining where input waveforms are periodic; that is, delay by sequences of vector assumes arbitrary input sequences but in wavepipelining circuits inputs are actually periodic. By Theorem 2, the first interval is $[3, \infty)$. The second interval is:*

$$\max(\frac{D_2^{max}}{H}, \frac{L^{max}}{H+1}) \leq \tau < \min(\frac{D_2^{min}}{H-1}, \frac{L^{min}}{H-2}) \tag{88}$$

or

$$\max(\frac{3}{2}, \frac{5}{2+1}) \leq \tau < \min(\frac{3}{2-1}, \frac{3}{2-2}) \tag{89}$$

which is $[1.67, 3)$. These results are tabulated in Table 1.

In the table, L denotes the use of topological delays in valid clocking interval computation, D_∞, delay by sequences of vector, D_2, 2-vector delay. All the intervals in the table are valid. L is most conservative assuming all paths are true, while D_2 is least conservative. As shown in the table, more conservative delays give narrower valid intervals. For instance, the lower portion, i.e. $(1.67, 2)$, of the valid interval $[1.67, 3)$ found by D_2 for $H = 2$ is regarded as invalid by using D_∞.

H	D_2	D_∞	L
1	$[3, \infty)$	$[4, \infty)$	$[5, \infty)$
2	$[1.67, 3)$	$[2, 3)$	$[2.5, 3)$

Table 1: Valid Clocking Intervals Computed Using different Delays

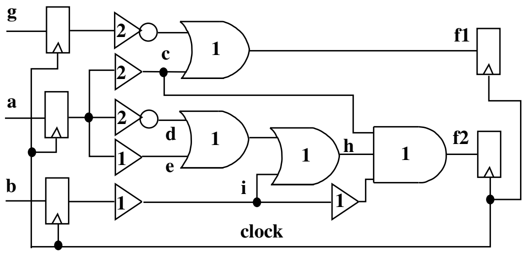

Figure 10: Circuit for Computing Valid Intervals

6 Computation of All Exact Valid Clock Intervals

At this point we know that in the interval $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$ the circuit computes $f[n-H]$. But there may be other values of τ outside of $[\frac{L^{max}}{H}, \frac{L^{min}}{H-1})$ where the circuit also computes $f[n-H]$. In particular, we saw by analyzing the false paths with more accurate delay analysis that this interval can be widened. However, there may be even more other values of τ where the circuit computes the correct value $f[n-H]$ for some H. In this section, we give algorithms to find *all* values of τ where $f(\ldots, x(n + \left\lfloor\frac{-k}{\tau}\right\rfloor), \ldots) = f[n-H]$ for different values of H. In general, for each τ, the circuit either computes $f[n-H]$ for some H or something not of the form $f[n-H]$ for any H. Thus we can think of $H(\tau)$ as being a function of τ, where for some τ it may not be defined.

Let $f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...)$ be the TBF of a stage of a pipeline. For a given value of H, all the exact valid clock periods are:

$$\{\tau : \; f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...) = f[n-H]\} \tag{90}$$

To find these intervals, we sweep τ in the interval $[\frac{L^{min}}{H}, \frac{L^{max}}{H-1})$, for $H > 1$. The reason of using this interval is that for $\tau \notin [\frac{L^{min}}{H}, \frac{L^{max}}{H-1})$ all arguments $n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor$ are unequal to $n-H$, because $n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor = n - H$ implies $\frac{k_i}{H} \leq \tau < \frac{k_i}{H-1}$. thus,

$$f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...) \neq f[n-H], \; \tau \notin [\frac{L^{min}}{H}, \frac{L^{max}}{H-1}). \tag{91}$$

Thus, the valid intervals can only exist within $[\frac{L^{min}}{H}, \frac{L^{max}}{H-1}]$; we do not have to examine any τ outside this interval. For $H = 1$, the sweeping interval is $[L^{min}, L^{max}]$. To sweep through $[\frac{L^{min}}{H}, \frac{L^{max}}{H-1})$, it can be shown that only a finite number of points, called time constants [12, 13], need to be examined to find all exact valid clocking intervals in $[\frac{L^{min}}{H}, \frac{L^{max}}{H-1})$. At each of these points, BDD's for $f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...)$ and $f[n-H]$ are built and compared. As τ is stepped through the time constants in decreasing order, we mark time constant τ_r the *beginning* of a valid interval, if

$$f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau_r}\right\rfloor), ...) = f[n-H] \tag{92}$$

and

$$f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau_{r-1}}\right\rfloor), ...) \neq f[n-H] \tag{93}$$

at the preceding time constant $\tau_{r-1} > \tau_r$. Similarly, a valid interval *ends* at time constant τ_s (intervals are open on the right), when

$$f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau_s}\right\rfloor), ...) \neq f[n-H] \tag{94}$$

and

$$f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau_{s-1}}\right\rfloor), ...) = f[n-H] \tag{95}$$

at the preceding time constant $\tau_{s-1} < \tau_s$. Note that this gives us maximal intervals of validity. Therefore, for a given H, we can find all exact valid clocking intervals. As can be seen from the above discussion, TBF's translate all temporal behaviors in wavepipelined circuits into ordinary logic manipulations which can be done efficiently using BDD's, provided the BDD's can be built.

Thus, to find all valid clocking intervals, we repeat the above algorithm for $H = 1, 2, 3, \ldots$ Note that for any H, the interval that must be examined $[\frac{L^{min}}{H}, \frac{L^{max}}{H-1})$ is never empty. So, in general, we would have to examine all $H \geq 1$. But of course we can not do this for all integers. We need a valid stopping criterion for H so that for $H > H_{stop}$, we are assured that no valid interval exists. For example, if for a given H there is no valid interval, does it mean that there is no valid interval for larger H's? To answer these questions, we study properties of $H(\tau)$.

6.1 Properties of the Harmonic Number $H(\tau)$

DEFINITION 4 *1. Let $H(\tau)$ of $f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...)$ be defined as follows:*

$$H(\tau) = \begin{cases} H, & \text{if } f(..., x_i(n + \left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...) = f[n-H] \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{96}$$

A valid clocking interval is one in which $H(\tau)$ is defined and constant over the entire interval.

2. Let I_1 and I_2 be two neighboring valid clocking intervals. $H(\tau)$ is **continuous** *if*

$$|H(\tau_1) - H(\tau_2)| \leq 1, \text{ for } \tau_1 \in I_1, \tau_2 \in I_2. \tag{97}$$

3. Let I_1 and I_2 be two disjoint valid clocking intervals, and $\tau_1 \in I_1$, $\tau_2 \in I_2$, $\tau_2 < \tau_1$. $H(\tau)$ is **monotonic** *if $H(\tau_1) \le H(\tau_2)$, and* **strictly monotonic** *if $H(\tau_1) < H(\tau_2)$.*

If $H(\tau)$ is continuous and strictly monotonic, the algorithm to find all valid intervals is simple. We start with $H(\tau) = 1$ and decrease τ (discretely, at the time constants) from L^{max} until $f(..., x_i(n+\left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...) \neq f[n-1]$. Since $H(\tau)$ is continuous and strictly monotonic, the next $H(\tau)$ is 2. So, the valid interval for $H(\tau) = 2$ starts at the first τ at which $f(..., x_i(n+\left\lfloor\frac{-k_i}{\tau}\right\rfloor), ...) = f[n-2]$. Valid intervals for other $H(\tau)$'s can be similarly found. Once we find the interval for an $H(\tau)$ to be empty, we know all intervals have been found because a non-empty interval at the next $H(\tau)$ would mean $H(\tau)$ is non-monotonic or discontinuous or both.

However, we show by examples that $H(\tau)$ can be discontinuous and non-monotonic.

6.1.1 Discontinuity of $H(\tau)$

Example 9 *Consider the circuit in Figure 11, which has the following TBF.*

$$f(\tau) = a(n + \left\lfloor\frac{-1.1}{\tau}\right\rfloor)\bar{a}(n + \left\lfloor\frac{-1.9}{\tau}\right\rfloor) + b(n + \left\lfloor\frac{-2.1}{\tau}\right\rfloor)\bar{b}(n + \left\lfloor\frac{-2.9}{\tau}\right\rfloor) + c(n + \left\lfloor\frac{-3.1}{\tau}\right\rfloor) + d(n + \left\lfloor\frac{-3.9}{\tau}\right\rfloor) \quad (98)$$

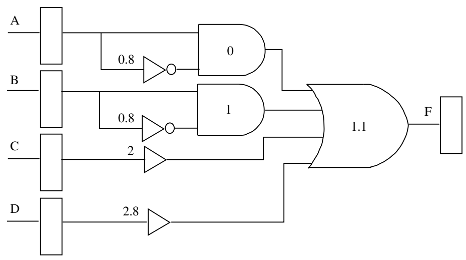

Figure 11: A Circuit with Discontinuous $H(\tau)$

Hence, $f[n-H] = c(n-H) + d(n-H)$. We sweep τ starting from $\tau = 3.9$ to obtain the first few valid clock intervals.

$$\begin{array}{lll l}
3.9 \le \tau, & f = & c(n-1) + d(n-1) = f[n-1] & \textbf{Valid} \\
3.1 \le \tau < 3.9, & f = & c(n-1) + d(n-2) & \\
2.9 \le \tau < 3.1, & f = & c(n-2) + d(n-2) = f[n-2] & \textbf{Valid} \\
2.1 \le \tau < 2.9, & f = & b(n-1)\overline{b}(n-2) + c(n-2) + d(n-2) & \\
1.95 \le \tau < 2.1, & f = & c(n-2) + d(n-2) = f[n-2] & \textbf{Valid} \\
1.9 \le \tau < 1.95, & f = & c(n-2) + d(n-3) & \\
1.55 \le \tau < 1.9, & f = & a(n-1)\overline{a}(n-2) + c(n-2) + d(n-3) & \\
1.45 \le \tau < 1.55, & f = & a(n-1)\overline{a}(n-2) + c(n-3) + d(n-3) & \\
1.3 \le \tau < 1.45, & f = & a(n-1)\overline{a}(n-2) + b(n-2)\overline{b}(n-3) & \\
 & & +c(n-3) + d(n-3) & \\
1.1 \le \tau < 1.3, & f = & a(n-1)\overline{a}(n-2) + b(n-2)\overline{b}(n-3)) & \\
 & & +c(n-3) + d(n-4) & \\
1.05 \le \tau < 1.1, & f = & b(n-2)\overline{b}(n-3) + c(n-3) + d(n-4) & \\
1.03 \le \tau < 1.05, & f = & c(n-3) + d(n-4) & \\
0.96 \le \tau < 1.03, & f = & c(n-4) + d(n-4) = f[n-4] & \textbf{Valid}
\end{array} \quad (99)$$

From above table, the first few valid intervals and the corresponding functions are:

$$
\begin{array}{ccc}
\textbf{Interval} & \textbf{f[n-H]} & \textbf{H}(\tau) \\
3.9 \leq \tau & f[n-1] & 1 \\
2.9 \leq \tau < 3.1 & f[n-2] & 2 \\
1.95 \leq \tau < 2.1 & f[n-2] & 2 \\
0.96 \leq \tau < 1.03 & f[n-4] & 4
\end{array}
\qquad (100)
$$

$H(\tau)$ versus τ is plotted in Figure 12.

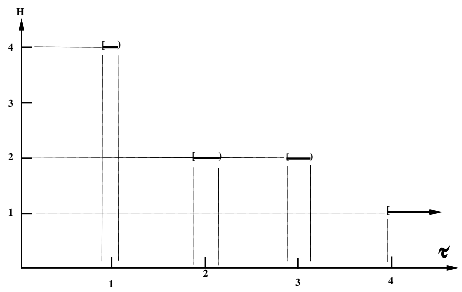

Figure 12: Valid Clocking Intervals for Example Circuit

It can be seen that the harmonic number $H(\tau)$ repeats 2 twice and skips 3. That is, if the clock frequencies in this pipeline is increased continuously, at some point the output is computing the correct function delayed by 2 units of time, then, computes incorrectly, computes again the correct function delayed by 2 units, miscomputes, and finally, computes, interestingly, the correct function delayed by 4, skipping 3. Therefore, $H(\tau)$ is neither continuous nor strictly monotonic and the region of validity for harmonic number $H = 2$ is two disconnected intervals. Note that the valid clocking interval for $H = 3$ is empty, but is not empty for $H = 4$; therefore, the stopping rule of H based on the non-existence of a valid interval is conservative, and would miss the interval for $H = 4$.

6.1.2 Non-Monotonicity of $H(\tau)$

Example 10 *In this example, we show that the harmonic number $H(\tau)$ can be non-monotonic, jumping back to its previous value. Consider the circuit in Figure 13, whose TBF is:*

$$f(x) = x(n + \left\lfloor \frac{-3}{\tau} \right\rfloor)(x(n + \left\lfloor \frac{-1}{\tau} \right\rfloor) + \bar{x}(n + \left\lfloor \frac{-2}{\tau} \right\rfloor)) + \bar{x}(n + \left\lfloor \frac{-3}{\tau} \right\rfloor)x(n + \left\lfloor \frac{-1}{\tau} \right\rfloor)\bar{x}(n + \left\lfloor \frac{-2}{\tau} \right\rfloor) \qquad (101)$$

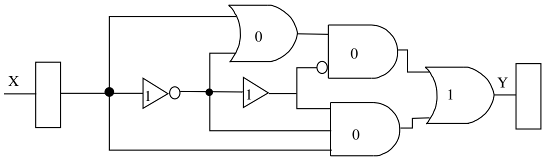

Figure 13: A Circuit with Non-monotonic $H(\tau)$

Thus, $f[n-H] = x(n-H)$. *Analyze the intervals as follows:*

$$\begin{array}{llcl}
3 \le \tau, & f(x) & = & x(n-1)(x(n-1)+\overline{x}(n-1)) \\
 & & & +\overline{x}(n-1)x(n-1)\overline{x}(n-1) \\
 & & = & x(n-1) \\
 & & = & f[n-1] \\
2 \le \tau < 3, & f(x) & = & x(n-2)(x(n-1)+\overline{x}(n-1)) \\
 & & & +\overline{x}(n-2)x(n-1)\overline{x}(n-1) \\
 & & = & x(n-2) \\
 & & = & f[n-2] \\
1.5 \le \tau < 2, & f(x) & = & x(n-2)(x(n-1)+\overline{x}(n-2)) \\
 & & & +\overline{x}(n-2)x(n-1)\overline{x}(n-2) \\
 & & = & x(n-1) \\
 & & = & f[n-1]
\end{array} \tag{102}$$

The valid intervals are:

Interval	f[n-H]	H(τ)
$3.0 \le \tau$	$f[n-1]$	1
$2.0 \le \tau < 3.0$	$f[n-2]$	2
$1.5 \le \tau < 2.0$	$f[n-1]$	1

(103)

$H(\tau)$ versus τ is plotted in Figure 14.

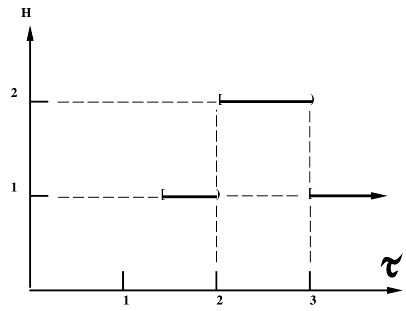

Figure 14: Valid Clocking Intervals for Example Circuit

$H(\tau)$ starts out with 1, goes up to 2, and comes down to 1. Physically, if the clock frequency is increased continuously, the circuit computes the correct function delayed by 1, computes the function delayed by 2, and finally re-computes, interestingly, the correct function delayed by 1. Hence, $H(\tau)$ is non-monotonic. This phenomenon is a bit counter-intuitive, because higher frequency should intuitively mean more data waves existing in the combinational logic between flip flops, thus, more latency, or higher harmonic number. The reason is that at $1.5 \leq \tau < 2.0$, paths having more than two data waves become false paths.

Therefore, in general it is difficult to find all valid clocking intervals. A compromise is to assume either continuity or monotonicity or both. By doing so, some valid intervals may be missed.

6.2 Algorithms for Continuous $H(\tau)$

For practical designs, $H(\tau)$ may be rarely discontinuous and non-monotonic. Assuming continuity and monotonicity reduces the complexity of computing valid clock intervals. The following algorithm, assuming continuous $H(\tau)$, is more efficient than the one in Section 6. This algorithm will find all exact intervals if $H(\tau)$ is indeed continuous and may miss some intervals otherwise. Even if the assumption on $H(\tau)$ for the algorithm is not true, the intervals found by these two algorithms are still exact, i.e. clocking at period outside of these intervals alters the functionality of the circuit.

Like the algorithm in Section 6 which visited the time constants in the interval $[\frac{L_{min}}{H}, \frac{L_{max}}{H-1})$ and repeated this for each H until H meets some stopping criterion, this algorithm adopts the stopping criterion that if for some H, there is no valid interval, then does not compute for any $\tilde{H} > H$. So all intervals with $\tilde{H} > H$ will be missed. In this case we would find all the intervals shown in Figure 14 but would miss the interval for $H = 4$ in Figure 12, since the region for $H = 3$ is empty.

For each H, starting with $H = 1$, visit the time constants in $[\frac{L_{min}}{H}, \frac{L_{max}}{H-1})$, compare $f(..., x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor), ...)$ and $f[n - H]$, and derive the intervals, the same procedure as that in the algorithm in Section 6. If the region is empty for some H, stop. The algorithm is in Figure 15.

Symbolically,

$$I = \bigcup_{i=1}^{N} H^{-1}(i) \tag{104}$$

$$\text{where } N = \min\{n : \ H^{-1}(n) = \phi\} \tag{105}$$

6.3 An Example: A 4-bit Pipelined Parallel Multiplier

In this section, we apply the above theory to a 4-bit pipelined parallel multiplier in Figure 16. The parallel multiplier is like a standard parallel multiplier except delays are added to minimize path delay differences. For example, buffers g_x, g_y, are added to the input word lines, x_i, y_i, in each cell, Figure 16(a) (there are more buffers than actually needed). C_i and P_i are the carry bit and the sum bit of the ith cell, respectively. The multiplier inputs and outputs are assumed to be latched at a period τ.

Assume gate delays as given in Table 2 (approximate values from MCNC lib2). Let D_{sum} be the delay of the sum bit of the adder in a cell, and D_{carry}, the delay of the carry bit. From the

Let $f(..., x_i(n + \left\lfloor \frac{-k_i}{\tau} \right\rfloor), ...)$ be a TBF of a wavepipelined circuit.
$I = \phi$; /* valid intervals, I */
$K = 1$;
do{
    /* Sweep τ in $[\frac{L^{min}}{K}, \frac{L^{max}}{K-1})$; replace $\frac{L^{max}}{K-1}$ by L^{max} if K=1 */
    for each time constant h in $[\frac{L^{min}}{K}, \frac{L^{max}}{K-1})$
        if($f(..., x_i(n + \left\lfloor \frac{-k_i}{h} \right\rfloor), ...) = f[n-K]$)
            mark h valid.
        else
            mark h invalid.
    $H^{-1}(K)$ = intervals bound by the valid h's;
    $I = I \bigcup H^{-1}(K)$;
    K++;
}while($H^{-1}(K) \neq \phi$)

Figure 15: Algorithm Computing Valid Clock Intervals with Continuous $H(\tau)$

above delay table, $D_{sum} = 10, D_{carry} = 12$. Assume the delays of the buffers (added intentionally to minimize delay differences) are adjusted so that the maximum delay path is the path along all carries and the minimum delay path is the path along all the sum bits. Then the maximum delay is along a path starting at a word line and propagating through four adders' carry bits, which is equal to $4 \times D_{carry} + D_{AND}$. Thus, $L^{max} = 53$, assuming the delay of an AND gate is equal to that of a NAND. The minimum delay is along a path starting at a word line and propagating through four adders' sum bits, which is equal to $4 \times D_{sum} + D_{AND}$. Thus, $L^{min} = 45$.

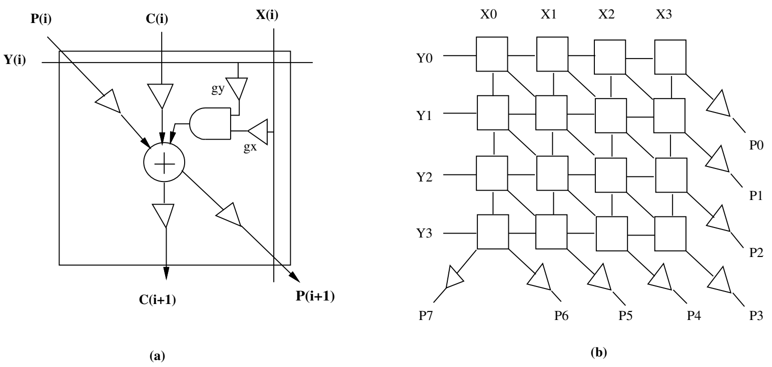

Figure 16: Pipelined Parallel Multiplier

Gate	Delay
Inverter	2
Buffer	2
NAND2	5
NAND3	7
XOR	10
Carry	12
Sum	10

Table 2: Delays of Library Gates

By Theorem 1, the valid clocking intervals are:

Valid Interval	**Harmonics H**
$53 \leq \tau$	1
$26.5 \leq \tau < 45$	2
$17.7 \leq \tau < 22.5$	3
$13.25 \leq \tau < 15$	4
$10.6 \leq \tau < 11.25$	5
$8.83 \leq \tau < 9$	6

(106)

The intervals and harmonic numbers are plotted in Figure 17. The vertical axis is the harmonic number corresponding to the intervals. It is interesting that the intervals are discrete and narrower at high harmonics, resembling electron energy bands, the uppermost interval unbounded above, successive intervals decreasing in width.

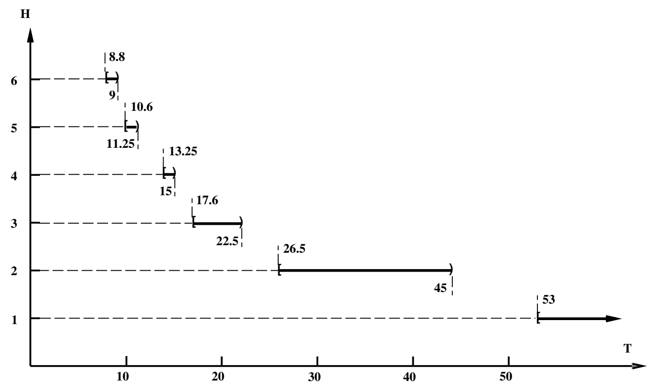

Figure 17: Valid Clock Intervals vs. Harmonics

If a bounded delay model is used, then the maximum gate delays are used in calculating L^{max}, and the minimum gate delays in L^{min}. For example, if the gate delays track with error of 10%, then,

$L^{max} = 53 \times 110\% = 58.3$, and $L^{min} = 45 \times 90\% = 40.5$. The valid intervals can be recalculated accordingly, as shown below. Note that the additional delay variations reduce the number of valid intervals drastically.

Valid Interval	**Harmonics H**
$58.3 \leq \tau$	1
$29.2 \leq \tau < 40.5$	2
$19.4 \leq \tau < 20.3$	3

(107)

7 Experimental Results

The following MCNC benchmarks were run on a DECstation 5000 (38 mips) with a standard SIS script and then mapped to a MCNC library. The actual delay values used are the ones given in the library. Without delay balancing, these circuits' maximum and minimum delays differ so much that they can only operate in non-wavepipelining mode, i.e. $H = 1$.

To make these circuits operational in wavepipelining mode, we balance the paths in the circuits by transistor sizing and buffer insertion, using the following balancing algorithm. The complexity of this algorithm is linear in the number of connections of the circuit.

Circuit Balancing Algorithm

1. Sort nodes in a topological order (a node appears later than all of its fanins.)
2. For each node, compute the lengths of the longest and the shortest partial paths from the node to the primary inputs.
3. For a node n, consider all its fanins that do not have the longest partial paths from n to the primary inputs. Resize the gates whose outputs are connected at the fanins to increase their lengths. If resizing of some fanins affect some longest paths in the circuit, add delays to the inputs of n at these fanins.

Figure 18: Circuit Balancing Algorithm

In a perfectly balanced circuit, the longest and the shortest path lengths are equal. Even a design can be perfectly balanced, the manufactured instances of the design may not be still perfectly balanced. For the circuits of the following table, we perfectly balanced them in *design* and assume that manufacturing causes a variation of 20% in path lengths; that is, $L^{min} = 80\% L^{max}$. Because the circuits are perfectly balanced in design, only manufacturing variations can cause the maximum delays (e.g. 2-vector, delay by sequences of vectors) to be less than that of L^{max}. Thus, it is not too conservative to assume all maximum delays are equal to L^{max}. We use D^{max} to denote this maximum delay. $D^{max} = L^{max}$. Similarly for D^{min}.

Circuit	D^{max}	D^{min}	Valid Intervals	H
C432	39	31.2	[39, ∞),[19.5,31.2),[13,15.6),[9.7,10,4)	1,2,3,4
C499	28.9	23.1	[28.9, ∞),[14.5,23.2),[9.6,11.6),[7.2,7.7)	1,2,3,4
C880	36.4	29.1	[36.4, ∞),[18.2,29.1),[12.1,14.6),[9,9.7)	1,2,3,4
C1355	28.9	23.1	[28.9, ∞),[14.5,23.1),[9.6,11.6),[7.2,7.7)	1,2,3,4
C1908	41.1	32.8	[41.1, ∞),[20.6,32.8),[14,16.4),[10,11)	1,2,3,4
C2670	36.5	29.2	[36.5, ∞),[18.3,29.2),[12.2,14.6),[9.1,9.7)	1,2,3,4
C3540	53.1	42.4	[53.1, ∞),[26.6,42.4),[17.7,21.2),[13.2,14.1)	1,2,3,4
C5315	51.6	41.2	[51.6, ∞),[25.8,41.2),[17.2,20.6),[12.9,13.7)	1,2,3,4
C6288	140	112	[140, ∞),[70,112),[46.6,56),[35,37.3)	1,2,3,4
C7552	59.8	47.8	[59.8, ∞),[29.9,47.8),[20,23.9),[15,15.9)	1,2,3,4
alu2	54.1	43.2	[54.1,∞),[27.0,43.2),[18.0,21.6),[13.7,14.4)	1,2,3,4
alu4	51.3	41.0	[51.3,∞),[25.7,41.0),[17.1,20.5),[12,8,13,7)	1,2,3,4
comp	25.6	20.5	[25.6,∞),[12.8,20.5),[8.5,10.3),[6.4,6.8)	1,2,3,4
cu	10.7	8.6	[10.7,∞),[5.4,8.6),[3.6,4.3),[2.6,2.8)	1,2,3,4
dalu	73.2	58.6	[73.2,∞),[36.6,58.6),[24.4,29.3),[18.8,19.8)	1,2,3,4
decod	3.4	2.7	[3.4,∞),[1.7, 2.7),[1.1,1.4),[0.8,0.9)	1,2,3,4
t481	26.3	21.0	[26.3,∞),[13.2,21.0);[8.8,10.3),[6.6,7)	1,2,3,4
des	37.2	29.8	[37.2∞),[18.6,29.8),[12.4,14.9),[9.3,9.9)	1,2,3,4
rot	31	24.8	[31.0,∞),[15.5,24.8),[10.3,12.4),[7.7,8.1)	1,2,3,4

As can be seen, after balancing, the circuits can operate at speeds as much as four times faster, even with manufacturing variations accounted for.

8 Conclusion

In this paper, we presented a systematic algebraic analysis of combinational wavepipelining circuits using Timed Boolean Function. With TBF's, we took into account circuit functionality in analyzing wavepipelined circuits and derived analytical expressions for valid clocking intervals in terms of topological, 2-vector, and single vector delays, both the longest and the shortest. These expressions subsume Cotten's lower bound. Further, these intervals eliminate the pessimism of long and short false paths and include effects of circuit parameters such as delay variations, clock skews, and setup and hold times of flip flops. Then, we studied the problem of computing all exact valid clocking intervals and its computational complexity by demonstrating discontinuity and non-monotonicity of the harmonic number $H(\tau)$ (the number of valid simultaneous data waves allowed) as a function of the clock period τ. Finally, we proposed algorithms to compute the exact valid intervals for a given set of harmonic numbers and demonstrate performance enhancement of balanced circuits from ISCAS benchmarks with gate delay variations.

References

[1] L. Cotten. Maximum rate pipelined systems. *1969 AFIPS Proceedings of Spring Joint Computer Conference*, pages 581–586, 1969.

[2] 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.

[3] B. Ekroot. Optimization of pipelined processors by insertion of combinational logic delay. *PhD thesis, Stanford University*, 1987.

[4] J. Fishburn. Clock skew optimization. *IEEE Transactions on Computers, Vol. 39, No. 7*, pages 945–951, July 1990.

[5] C. T. Gray, W. Liu, and R. K. Cavin III. Timing constraints for wave-pipelined systems. *IEEE Transactions on Computer-Aided Design*, pages 987–1004, August 1994.

[6] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers, 1989.

[7] D. Joy and M. Ciesielski. Placement for clock period minimization with multiple wave propagation. *Proceedings of 28th Design Automation Conference*, pages 640–643, 1991.

[8] D. Joy and M. Ciesielski. Clock period minimization with wave pipelining. *IEEE transactions on Computer-Aided Design*, pages 461–472, April 1993.

[9] P. Kogge. *The Architecture of Pipelined Computers.* Prentice-Hall, Englewood Cliffs, N.J., 1981.

[10] W. Lam, R. Brayton, and A. Sangiovanni-Vincentelli. Circuit delay models and their exact computation using timed boolean functions. *IEEE/ACM Design Automation Conference '93*, 1993.

[11] W. Lam, R. Brayton, and A. Sangiovanni-Vincentelli. Exact minimum cycle times for finite state machines. *1993 ACM Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Malente, Germany, Sept. 1993.

[12] 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.

[13] 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.

[14] W. Lam, R. Brayton, and A. Sangiovanni-Vincentelli. Valid clocking in wavepipelined circuits. *IEEE/ACM International Conference on Computer Aided Design, '92*, Nov., 1992.

[15] W. Lien and W. Burleson. Wave-domino logic: Timing analysis and applications. *1992 ACM Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 1992.

[16] H. Loomis. The maximum rate accumulator. *IEEE Transactions on Electronic Computers*, pages 628–639, August 1966.

[17] K. Sakallah, T. Mudge, T. Burks, and E. Davidson. Synchronization of pipelines. *Technical Report CSE-TR-97-91, Computer Science and Engineering, University of Michigan*, Feb., 1990.

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

[19] D. Wong, D. DeMicheli, and G. Flynn. Inserting active delay elements to achieve wave pipelining. *IEEE/ACM International Conference on Computer-Aided Design*, pages 270–273, Nov., 1989.

[20] D. Wong, D. DeMicheli, G. Flynn, and R. Huston. A bipolar population counter using wave pipelining to achieve 2.5x normal clock frequency. *IEEE Journal of Solid State Circuits*, pages 745–753, May, 1992.