Minimizing the Effect of Clock Skew Via Circuit Retiming[1]

Brian Lockyear and Carl Ebeling

Department of Computer Science and Engineering
University of Washington
Seattle, Washington 98195

Technical Report 93-5-04
May, 1992

[1]This research was funded in part by the Defense Advanced Research Projects Agency under Contract N00014-J-91-4041. Carl Ebeling is supported in part by an NSF Presidential Young Investigator Award with matching funds provided by IBM Corporation and Sun Microsystems.

Abstract

Clock skew is often cited as an impediment to designing high-performance synchronous circuits. Clock skew reduces the performance of a circuit by reducing the time available for computation, and it may even cause circuit failure by allowing race conditions. In this paper, we show how to use retiming to reduce the effect of clock skew in both edge-clocked and level-clocked circuits. We include both *fixed* clock skew, for example skew which results from clock distribution wiring and buffering, and *variable* clock skew which results from uncontrolled variation in process parameter and operating conditions. By including fixed skew in the maximum constraint equations retiming can find the fastest circuit given that skew. By including variable skew, retiming can also generate the circuit that tolerates the most variation in clock skew for a given clock frequency. Clock skew can also be used to speed up circuits using a technique similar to retiming described by Fishburn [2]. We describe a method for combining this technique which uses added clock skew with retiming to optimize the performance of edge-clocked circuits.

1 Introduction

Retiming is a circuit optimization technique well-known to hardware designers which is used to reduce the clock period of a circuit by relocating registers to maximize the amount of computation done in each clock period. For example, in order to pipeline a circuit, a designer places the registers to spread computation out evenly over several cycles, thereby minimizing the cycle period. While retiming has long been an ad hoc technique used in design, it was not until 1983 that an efficient algorithm was reported for automatically retiming circuits [4]. In this work, Leiserson, Rose and Saxe formalized retiming as an optimization technique that improved the performance of a circuit without changing its external behavior. This early work was limited to circuits using edge-triggered registers. However, high-performance systems make wide use of level-sensitive latches to allow computations to borrow time across clock cycle boundaries. Only recently have efficient algorithms for retiming been described for these level-clocked circuits [7, 3].

In this paper we extend retiming to handle the problem of clock skew in both edge-clocked and level-clocked circuits. Clock skew slows down circuits by reducing the time available for computation and in some cases it may cause race conditions. As circuits become faster, clock skew will become even more of a problem for synchronous circuit design. Clock skew can be divided into two components: *fixed* skew, which is built into the delivery of the clock, and *variable* skew, which is caused by variations in process parameters, temperature, power supply voltage and other operating conditions. Fixed skew is that which the designer controls or at least can measure in the clock distribution tree, while variable skew is the unpredictable variation in delay that can occur on the clock signal. Including clock skew in the retiming constraints allows the retiming to find faster circuits and also increase tolerance to clock skew variation.

We begin in Section 2 by reviewing the circuit and clock model used by the retiming algorithms. We then describe how clock skew is added to this model, along with synchronizer[2] delay and setup time.

The remainder of the paper describes four related results. Section 4 first describes the relatively straightforward extension to retiming that adds clock skew to the edge-triggered circuit model and retiming algorithms developed by [4]. Section 5 then extends our level-clocked retiming algorithms [7] to incorporate clock skew as well. In neither case does the inclusion of clock skew increase the asymptotic time complexity of the algorithms. These two sections show that fixed clock skew can be treated as modifying the delay of circuit elements. Thus retiming can generate the fastest circuit given a fixed skew, which can be faster or slower than the fastest possible circuit with zero clock skew. Retiming also ensures that the circuit generated will operate correctly in the presence of variable skew, which always slows down the circuit.

Section 6 describes how level-clocked circuits can be retimed to maximize the tolerance to variation in clock skew in level-clocked circuits. Although minimizing the clock period is equivalent to maximizing clock skew tolerance in edge-clocked circuits, this is not always true for level-clocked circuits.

Finally, Section 7 shows how fixed clock skew can be used to actually increase the performance of an edge-clocked circuit. In a technique closely related to circuit retiming presented by Fishburn in [2],

[2]We will use the term synchronizer when referring to either registers or latches.

delay elements are placed in the clock distribution network to intentionally skew the clock and allow longer computational delay between selected registers at the expense of reduced delay between others. For a fixed register placement in an edge-triggered circuit, Fishburn presents a Linear Program that solves for the skews required to produce the minimum possible clock period. We present here our preliminary work on combining retiming with intentional clock skew to produce faster circuits than can be produced with either technique used alone.

2 Background

We begin this paper with a review of the circuit and clock models presented in [4, 7] for edge-triggered and level-clocked circuits. The reader is encouraged to read these earlier papers for full details.

2.1 Circuit Graph Model

A circuit is represented as a graph with a vertex, v, for each functional element and an edge, $u \xrightarrow{e} v$, for each interconnecting wire. Each vertex has delay, $d(v)$, the maximum delay of the corresponding functional element. A unique host vertex v_h, with $d(v_h) \equiv 0$, represents the environment external to the circuit. Registers and latches are placed on the edges connecting vertices and each edge has a weight, $w(e)$, indicating the number of registers or latches on the connection.

A path $u \overset{p}{\rightsquigarrow} v$ is a sequence of vertices and edges from u to v. A *simple* path contains no vertex twice. The weight $w(p)$ of a path $p = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} v_n$ is the number of registers or latches placed along it, that is, the sum of the edge weights: $w(p) = \sum_{i=0}^{n-1} w(e_i)$. We say that registers or latches are *adjacent* if the path connecting them has zero weight. The weight of a cycle is the weight of the same sequence of edges and vertices treated as a path. Similarly, the delay of a path $d(p)$ is the sum of the delays of the vertices along the path: $d(p) = \sum_{i=0}^{n} d(v_i)$. The delay of a cycle $d(c)$, $c = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots v_{n-1} \xrightarrow{e_{n-1}} v_0$, includes the delay of vertex v_0 only once; hence $d(c) = \sum_{i=0}^{n-1} d(v_i)$.

A circuit G is transformed into a corresponding retimed circuit G_r through assignment of a *retiming* (or lag) value $r(v)$ to each of the vertices in G. This retiming value represents the number of registers (latches) removed from the output edges of vertex v and added to the input edges. The resulting weight of an edge $u \xrightarrow{e} v$ in the retimed graph is: $w_r(e) = w(e) + r(v) - r(u)$.

2.2 Clock Model

For our retiming work we have adopted the clock model of Sakallah, Mudge & Olukotun [8] which provides a convenient way to describe the resulting timing constraints. A *k-phase clock* is a set of k periodic signals, $\Phi = \{\phi_1 \ldots \phi_k\}$, where ϕ_i is *phase i* of the clock Φ. All ϕ_i have a common cycle time T_Φ. An edge-triggered circuit has a single clock phase and values are passed through the registers once per cycle on the "clocking" edge. For level-clocked circuits, each phase divides the clock cycle into two intervals as shown in Figure 1: an *active* interval of duration T_{ϕ_i} and a *passive* interval of duration $(T_\Phi - T_{\phi_i})$. The latches controlled by a clock phase are *enabled* during its active interval and *disabled* during its passive interval. The clock transitions *into* and *out of* the active interval are called the *enabling* and *latching* edges respectively. We refer to the clock phase controlling latch l as $P(l)$.

Relative to the beginning of its passive interval at time $t = 0$, the enabling edge of a phase occurs at $t = T_\Phi - T_{\phi_i}$, and its latching edge at T_Φ (Figure 1). Sakallah et al. additionally introduce an arbitrary *global time reference* and values e_i denoting the time relative to the global time reference at which phase ϕ_i *ends* for some specified cycle. Phases are ordered relative to the global time reference so that $e_1 \leq e_2 \leq \cdots \leq e_{k-1} \leq e_k$ and $e_k \equiv T_\Phi$. The phase following ϕ_i in the clock set is referred to as ϕ_{i+1} with phase $\phi_{k+1} \equiv \phi_1$ and $\phi_{1-1} \equiv \phi_k$.

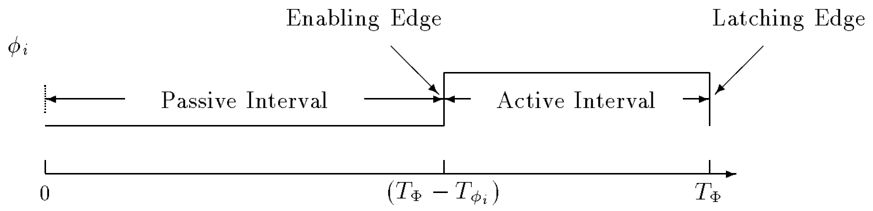


Figure 1: *Diagram from Sakallah et al. showing a clock phase ϕ_i and its local time zone.*

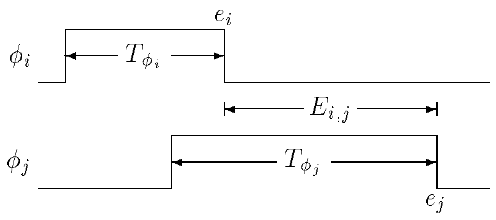


Figure 2: *The phase shift operator provides the relative difference between times in the local time zones of different phases.*

Finally, a *phase shift* operator, $E_{i,j}$, is defined as:

$$E_{i,j} \equiv \begin{cases} (e_j - e_i), & \text{for } i < j \\ (T_\Phi + e_j - e_i), & \text{for } i \geq j \end{cases}$$

$E_{i,j}$ takes on positive values in the range $(0, T_\Phi]$. When subtracted from a time point in the *current* period of ϕ_i, it changes the frame of reference to the *next* period of ϕ_j, taking into account a possible cycle boundary crossing (Figure 2). Because the period of each phase is identical and $e_i \geq e_{i-1}$, the sum of the shifts between k successive phases is T_Φ:

$$\sum_{i=1}^{k} E_{i,i+1} = T_\Phi. \tag{1}$$

A symmetric level-clocked schedule is one in which all active phase periods T_{ϕ_i} are equal and all phase shifts $E_{i,i+1} = \frac{T_\Phi}{k}$.

The latest arrival time of a signal at latch l is denoted by A_l and the latest departure time by D_l, both in terms of the local time zone.

Note that this clock model does not provide for clock phases with differing periods nor for gated clock signals. For simplicity, we assume that the delay characteristics of synchronizers do not vary as they are moved across combinational logic.

2.3 Correct Operation, Valid Schedules and Well-formed Circuits

We define a circuit to be correctly timed if for any pair of adjacent synchronizers the signal leaving the first arrives at the second during the *next* clock period in edge-triggered circuits or the next clock phase in level-clocked ones. Thus the definition of correctness for a level-clocked circuit is a straightforward extension of the definition of correct operation commonly used for edge-clocked circuits. Timing correctness can be summarized as a pair of timing constraints for each case (illustrated for level-clocked circuits in Figure 3).

For any two adjacent registers r and s in an edge-clocked circuit:

E1. Maximum Delay: $A_s = d(p) \leq T_\Phi$

E2. Non-interference: $A_s = d(p) > t_{hold}$

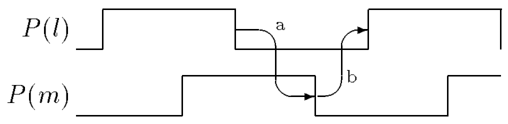


Figure 3: *Graphical representation of the constraints on the clock phases that are required for correct operation of a well-formed level-clocked circuit. Latches l and m are any pair connected by a zero-weight path beginning from l.*

For any two adjacent latches l and m in a level-clocked circuit:

L1. Maximum Delay: $A_m = D_l + d(p) - E_{P(l),P(m)} \leq T_\Phi$

L2. Non-interference: $A_m = D_l + d(p) - E_{P(l),P(m)} > t_{hold}$

These constraints assume that clock skew, synchronizer propagation delay and setup time are all zero. These parameters will be added later in this paper.

The retiming techniques in [7] restrict the type of clocks and circuits that can be retimed. Clock schedules must be "valid" so that only maximum delay constraints need to be satisfied for correct operation. Valid clock schedules do not allow races to occur even if all circuit delays are zero. This is guaranteed if for any two latches l and m connected by a zero-weight path $l \twoheadrightarrow m$, $E_{P(m),P(l)} > T_{\phi_l}$. In general, the class of valid clocks includes schedules with phase overlap, underlap, or both; however, two-phase clocks are required to be non-overlapping and no single phase schedules are allowed.

In addition, level-clocked circuits must be "well-formed," that is, latches must occur in phase order along every path.[3] Retiming is always free to move latches across vertices in well-formed circuit graphs and well-formed circuits remain well-formed following retiming. Furthermore, the minimum number of registers required on a path can be computed solely from the path delay and the phase of the first latch.

Since this work extends our previous work, these restrictions on clock schedules and level-clocked circuits also apply to this paper. Although the designer is constrained to work within this model, it applies to most circuits commonly used in practice.

3 Clock Skew Parameters

We use two parameters to model the fixed clock skew: $\sigma_r(l)$ and $\sigma_f(l)$, the fixed skew of the rising and falling edge respectively of the clock at latch l.[4] In edge-triggered circuits only one clock edge is of interest, thus $\sigma(r)$ is used to represent the skew of that edge at register r. If skew is positive the clock arrives late relative to the reference clock and if negative it arrives early. For an edge e, $\sigma_r(e)$ and $\sigma_f(e)$ are the fixed skew of a clock at the physical circuit location of the wire represented by the edge. Adding to the clock skew effectively moves an equivalent amount of delay from the fanins of the synchronizer to the fanouts.

Variable skew occurs due to variations in circuit fabrication and operating conditions. The maximum amount by which these variations may shift clock edges in either direction away from the fixed skew is modeled by the parameter σ_ϵ. Thus a rising edge may arrive as late as $\sigma_r(l) + \sigma_\epsilon$ and as early as $\sigma_r(l) - \sigma_\epsilon$. In this work we assume that variable skew is the same across the circuit, although our techniques can easily be extended to make the variable skew depend on physical location.

In order for us to be able to use the efficient algorithms we have already developed for retiming circuits, we must make the following restriction on clock skew. The relative clock skew between the input and output edges of a vertex, that is the difference in the clock skew between the synchronizers on these edges, must

[3] A simple exception to this rule allows input and output signals at the host node to occur on different phases as long as timing constraints across the host are not imposed.

[4] Properly these parameters should refer to the enabling and latching edges rather than rising and falling, however, the letters e and l are commonly used elsewhere in the text to refer to edges and latches. We assume that the enabling edge is the rising edge and the falling edge is the latching one.

be no greater than the maximum delay of the vertex. We call this the skew monotonicity constraint and it ensures that we need not consider the case where increasing the length of a path reduces the number of synchronizers required. It is easy to see that this constraint is met by practical circuits. Note that this does not impose a minimum on the combinational delay. If the actual delay is less than the skew, a more conservative circuit than necessary will be generated by the algorithms. Formally, the requirement is stated as:

- For every vertex v and pair of edges $e \in \text{fanin}(u)$ and $\acute{e} \in \text{fanout}(u)$:

  SR: $\sigma_r(\acute{e}) - \sigma_r(e) \leq d(u)$

  SF: $\sigma_f(\acute{e}) - \sigma_f(e) \leq d(u)$

In addition to parameters for clock skew, it is straightforward to add register or latch propagation delay, P, and required setup time, S, into our model. We make the simplifying assumption that P and S are the same for all synchronizers in the circuit. Algorithms for asymmetric level-clocked circuits in [7, 3] could be extended to solve problems in which propagation delays and setup requirements vary with physical placement of synchronizers, with the exception of propagation delay in level-clocked circuits, a limitation discussed in Section 5.

4 Clock Skew in Edge-Triggered Circuits

This section incorporates the parameters of clock skew, register propagation delay and setup time into the edge-triggered timing algorithms developed in [4]. While this is a relatively simple extension of the previous work, it provides the necessary first step towards algorithms which combine retiming and intentional clock skew (Section 7) and a basis of understanding for the effects of skew on level-clocked circuits.

Register propagation delay and setup time directly reduce the maximum delay allowed on a zero-weight path. Propagation delay causes the signal at the beginning of the path to be delayed from the clocking edge by P, while setup time requires that signals arrive S time prior to the next clocking edge. Thus the time available for computation is reduced by both P and S.

Clock skew increases the maximum delay allowed on a path between two registers if the clocking edges occur farther apart and decreases it if they occur closer together. The latest possible arrival of a clocking edge at register r is $\sigma(r) + \sigma_\epsilon$ and the earliest at s is $\sigma(s) - \sigma_\epsilon$. Thus the maximum delay allowed from r to s is increased by $\sigma(s) - \sigma(r) - 2\sigma_\epsilon$. Thus the maximum delay constraint for each pair of adjacent registers r and s in an edge-clocked circuit becomes:

E1'. Maximum Delay: $A_s = d(p) \leq T_\Phi - S - P + \sigma(s) - \sigma(r) - 2\sigma_\epsilon$

Clock skew can also cause edge-triggered circuits to fail by causing race conditions due to short paths. The delay of the path is increased by the register propagation delay P, but the relative skew between the two clocks decreases the effective hold time of the register s. Thus the minimum delay constraint becomes:

E2'. Non-interference: $A_s = P + d(p) > t_{hold} + \sigma(s) - \sigma(r) + 2\sigma_\epsilon$

We can enforce these constraints with a static check that ensures that E2' holds independently for each vertex in the circuit. This can be done because the fixed skew is associated with edges, not registers which may be moved by the retiming. If this constraint holds for all vertices, then it holds for all longer paths. Thus E2' holds for all retimings of the circuit and minimum delay retiming constraints are not required.

To retime a circuit to satisfy the delay constraint E1', the value of $r(u) - r(v)$ is constrained so that there is at least one register on each path $u \overset{p}{\twoheadrightarrow} v$ whose delay exceeds that given by E1'. That is, for each path $u \overset{p}{\twoheadrightarrow} v$ and pair of edges $e \in \text{fanin}(u)$ and $\acute{e} \in \text{fanout}(v)$, if:

$$d(p) \quad > \quad T_\Phi - P - S + (\sigma(\acute{e}) - \sigma(e)) - 2\sigma_\epsilon, \tag{2}$$

then a constraint of the form $r(u) - r(v) \leq w(p) - 1$ is imposed to insure that the path contains at least one register. If registers are not actually placed on edges e and $\acute{e}$ the skew of those edges will not impact the clock period; however, the clock skew monotonicity constraint prevents the possibility that placing registers farther apart will reduce skew more than the path delay is increased. Thus it is only possible to satisfy the circuit timing constraints by placing registers closer together, *i.e.* by requiring the weight of the path in the retimed circuit to be at least one.

Clearly a potentially exponential number of path constraints exist; however, because each constraint is of the form:

$$r(u) - r(v) \leq C(u \overset{p}{\twoheadrightarrow} v),$$

where $C(u \overset{p}{\twoheadrightarrow} v)$ is the constant $w(p) - 1$, it is possible to identify critical paths which result in a minimum value of $C(u \overset{p}{\twoheadrightarrow} v)$. Satisfaction of the smallest value of $C(u \overset{p}{\twoheadrightarrow} v)$ implies that all other constraints on the same pair of variables are satisfied as well. Because propagation delay, setup and skew are unaffected by selection of the path joining u and v, a critical path p remains one with maximum delay of all paths with minimum weight as identified in [4]. $W(u, v)$ and $D(u, v)$ are defined as the weight and delay of critical paths. The critical fanin and fanout edges are those for which:

$$\begin{aligned} \sigma_{\max}(u) &= \max\{\sigma(e) \ : \ e \in \text{fanin}(u)\} \\ \sigma_{\min}(v) &= \min\{\sigma(e) \ : \ e \in \text{fanout}(v)\}. \end{aligned}$$

Thus the following theorem allows identification of a retiming R which satisfies all circuit timing constraints in the presence of register propagation delay, setup time and clock skew.

Theorem 1: *An edge-triggered circuit G under retiming R is correctly timed iff:*
1) For all vertices u and v connected by a path in G such that

$$D(u, v) > T_\Phi - P - S + (\sigma_{\min}(v) - \sigma_{\max}(u)) - 2\sigma_\epsilon,$$

$$r(u) - r(v) \leq W(p) - 1;$$

and 2) For all edges $u \xrightarrow{e} v \in G$: $r(u) - r(v) \leq w(e)$.

Proof Sketch: Satisfying the critical path constraints guarantees that the timing constraints in Eqn. 2 are satisfied. Alternatively, if a critical path constraint is not satisfied, skew monotonicity guarantees that some timing constraint from Eqn. 2 is also unsatisfied. The edge constraints prevent negative weight edges which are physically meaningless. ∎

The retiming constraints presented in Theorem 1 are in the form of an Integer Linear Program (ILP) and may be solved directly using the Bellman-Ford technique or translated to a form solvable using the Mixed-ILP or Feas algorithms presented in [5]. Because the clock period is determined by some timing constraint relationship expressed in Eqn. 2 exactly met by the circuit, the minimum possible clock period exists in the set:

$$T_{\Phi_{opt}} \in \{D(u, v) + P + S - (\sigma_{\min}(v) - \sigma_{\max}(u)) + 2\sigma_\epsilon\} \qquad \text{for } u \text{ and } v \in V.$$

A binary search over this set identifies the optimum clock period. The minimum period and a retiming to it may be found in $O(|V|^2 \log |V|)$ time by using the simple Feas algorithm from [5] for constraint solution.

Figure 4 shows a correlator circuit in which two edges have early arriving clocks modeled by a fixed skew of $\sigma = -1$. All other edges have fixed skew of 0 and $\sigma_\epsilon = 0$. Unlike the original circuit from [4], cross-host timing constraints are prevented by assuming that a pair of registers are hidden inside the host vertex. Thus the circuit is acyclic. At the optimal clock period of $T_\Phi = 9$, the early arrival of the clock at $v_3 \rightarrow v_4$ prevents the placement of a register on that edge from satisfying the delay constraint on path $v_1 \twoheadrightarrow v_3$. On the other hand, the early arrival at $v_2 \rightarrow v_3$ allows path $v_3 \twoheadrightarrow v_5$ with delay $d(v_3 \twoheadrightarrow v_5) = 10$ to have zero weight.

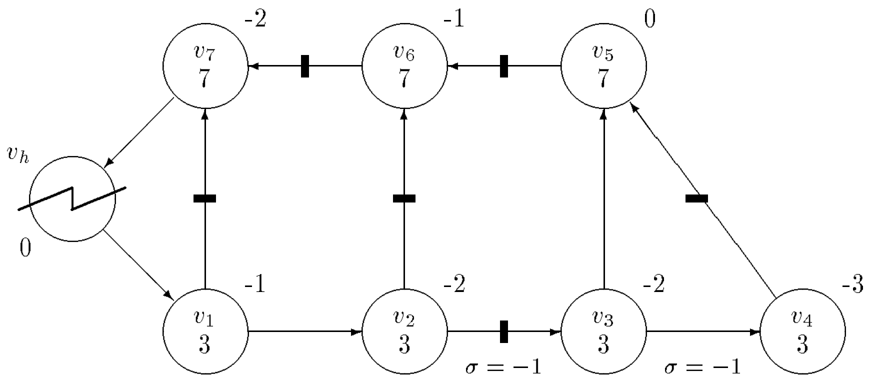


Figure 4: *The edge-triggered correlator from [4] with fixed skew in clock signals delivered to edges $v_2 \rightarrow v_3$ and $v_3 \rightarrow v_4$.*

5 Level-Clocked Circuits and Synchronization Parameters

As in edge-triggered circuits, timing constraints in level-clocked designs are derived from the time span between clock edges, from the rising edge enabling the first latch on the path to the falling edge latching the final latch. Latch propagation delay and setup time again directly reduce the time available for computation along circuit paths, while clock skew changes the time span between the relevant edges (see Figure 5).

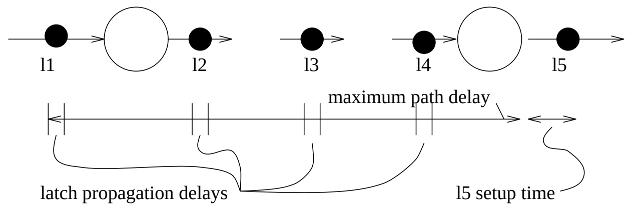


Figure 5: *The propagation delay of each latch placed along the path must be counted against the maximum path delay. Only the setup time of the final latch placed along the path is counted against the maximum delay.*

Unlike edge-triggered timing constraints, level-clocked timing constraints extend across paths with multiple latches. Signals are still required to arrive at the final latch of a path S time prior to the arrival of the falling edge; however, the constraints must account for the propagation delay of all intermediate latches as well as the initial one (see Figure 5). Since retiming constraints are stated in terms of the retiming values $r(u)$ and $r(v)$ of the vertices at the beginning and end of a path and they can refer only to the weight of the path, not the location of latches on the path. Thus it is not possible to vary the latch propagation delay based on latch location.

With skew, the departure time of a signal from a latch l is given by:

$$D_l = \max\left\{A_l, T_\Phi - T_{P(l)} + \sigma_r(l) + \sigma_\epsilon\right\} + P, \tag{3}$$

and the arrival time at a subsequent latch m connected by a zero-weight path is:

$$A_m = D_l + d(p) - E_{P(l),P(m)}. \tag{4}$$

Correct timing requires that signals arrive at latch m prior to S time before the earliest occurrence of its falling edge at $T_\Phi - \sigma_f(m) - \sigma_\epsilon$ (see Figure 6). Thus the maximum delay constraint for each pair of adjacent latches l and m in a level-clocked circuits becomes:

L1'. Maximum Delay: $A_m = D_l + d(p) - E_{P(l),P(m)} \leq T_\Phi - S - \sigma_f(m) - \sigma_\epsilon$

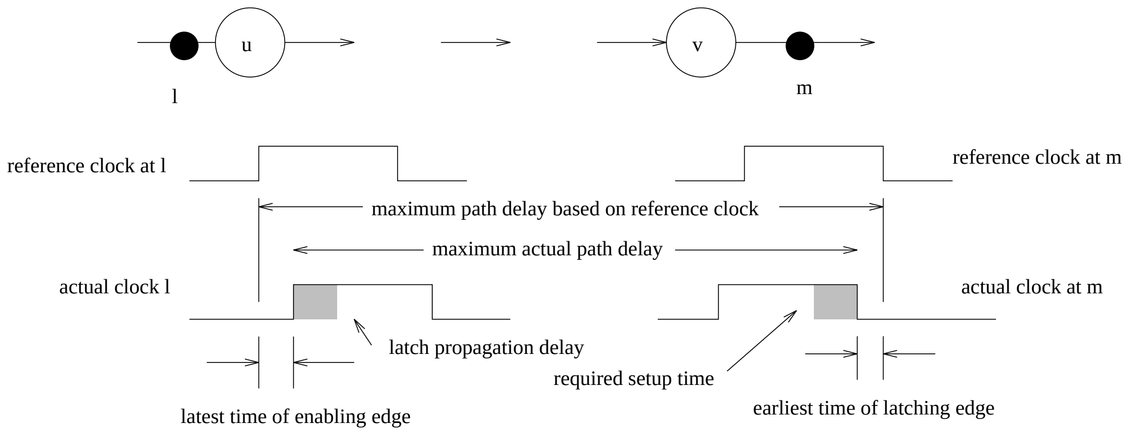

u
v
l
m
reference clock at l
reference clock at m
maximum path delay based on reference clock
maximum actual path delay
actual clock l
actual clock at m
latch propagation delay
required setup time
latest time of enabling edge
earliest time of latching edge

Figure 6: *Late arrival of a signal at latches l early arrival at m reduces the maximum allowable delay of path u to v and may require a latch to be placed on the path.*

L2'. Non-interference: $A_m = D_l + d(p) - E_{P(l),P(m)} > t_{hold} + \sigma_f(m) + \sigma_\epsilon$

We first address the L2' constraint which constrains the minimum delay in the circuit. By combining this constraint with the earliest possible departure time of a signal from latch l which is $T_\Phi - T_{P(l)} + P + \sigma_r(l) - \sigma_\epsilon$, we get a minimum delay constraint of:

$$d(p) > t_{hold} - P + \sigma_f(m) - \sigma_r(l) + 2\sigma_\epsilon - T_\Phi + T_{P(l)} + E_{P(l),P(m)}$$

where the term $T_\Phi - T_{P(l)} - E_{P(l),P(m)}$ is the amount of underlap between the two clock phases. In our previous work we have ignored short paths by relying on valid clock schedules to avoid races even when the minimum delay is zero. If the latch hold, propagation delay, and clock skew are all zero as we assumed in previous work, then this reduces to a constraint that the path delay be greater than the phase overlap (the amount of negative underlap). Allowing minimum delays to be zero then leads to the definition of a valid clock schedule, which for the case of 2-phase clocks forbids phase overlap. We can weaken the valid clock schedule constraint somewhat by following the same approach we took for edge-clocked circuits, ensuring via a static check that the minimum delay constraint will be satisfied regardless of retiming. We do this by checking that the minimum delay constraint L2' holds for each vertex independently. This allows retiming the freedom to place latches on any edge without violating the minimum delay constraint. Note that in the case of level-clocked circuits, the underlap value used in the static check must be the smallest over all pairs of successive phases, since the latches assigned to the edges may use any pair.

5.1 Maximum Path Delay Constraints

By combining the earliest possible departure time of a signal from l, $T_\Phi - T_{P(l)} + \sigma_r(l) + \sigma_\epsilon + P)$, and the latest permissible arrival at each successive latch on a path in turn, we can compute the maximum allowable delay along a signal path of any weight. The next theorem uses the definition of correct circuit timing to identify the latching clock edge for each latch. Since latches occur in phase order along paths of well-formed circuits, the available computation time is computed by summing successive phase shifts. An important aspect of this result is that only the skew of the initial rising and final falling edges appears in the maximum path delay constraint. Although each internal latch along the path may be affected by skew, the skew is added to the time on one side of the latch and subtracted from the time on the other side, thus canceling out. Thus clock skew can vary with latch location since the skew appearing in the maximum delay constraints can be associated with specific edges.

Theorem 2: *A multi-phase, level-clocked graph using a valid clock schedule is correctly timed only if the delay of any simple path* $l_0 \overset{p}{\twoheadrightarrow} l_{n+1}$ *is bounded by:*

$$d(p) \le T_{\phi_{l_0}} + \sum_{i=0}^{w(p)} \left(E_{P(l_i),P(l_{i+1})} - P\right) - \sigma_r(l_0) + \sigma_f(l_{n+1}) - 2\sigma_\epsilon - S.$$

Proof sketch. Through induction on the path weight, and substitution of $D_l = T_\Phi - T_{P(l)} + \sigma_r(l) + \sigma_\epsilon + P$ for the earliest possible departure time of a signal from latch l and $A_{l_{n+1}} = T_\Phi - \sigma_f(l_{n+1}) - \sigma_\epsilon$ for the maximum permissible arrival time at latch m, Eqn. 4 becomes:

$$\begin{aligned} d(p) \quad \le \quad & T_{\phi_{l_0}} - \sigma_r(l_0) - \sigma_\epsilon + \sigma_f(l_0) - \sigma_\epsilon - S + \\ & \sum_{i=0}^{w(p)} \left(E_{P(l_i),P(l_{i+1})} + (\sigma_f(l_{i+1}) + \sigma_\epsilon - \sigma_f(l_i) - \sigma_\epsilon)\right) - \sum_{i=0}^{w(p)} P. \end{aligned}$$

This equation simplifies to the desired result. ■

5.2 Critical Cycles

Because the latch at the start of the cycle is the same as the one at the end, the maximum delay allowed around a complete cycle in the circuit is unaffected by clock skew or setup time; however, it is reduced by the total latch propagation delay. This amount is fixed since propagation delay is constant and the number of latches on a cycle is unchanged by retiming. The lower bound on possible clock periods caused by circuit cycles may again be found using the max-ratio-cycle algorithm.

Theorem 3: *A multi-phase, level-clock graph using a valid clock schedule is correctly timed only if the delay of any cycle* $l_0 \overset{c}{\twoheadrightarrow} l_{n+1}$ *is bounded by:*

$$d(c) \le \sum_{i=0}^{w(c)-1} \left(E_{P(l_i),P(l_{i+1})} - P\right)$$

Proof omitted.

Corollary 4: *A well-formed graph using a k-phase clock schedule is correctly timed if and only if:*

$$\forall \text{ cycles } c \in G : T_\Phi \ge k\left(\frac{d(c)}{w(c)} + P\right).$$

Proof sketch: In a well formed graph each cycle must contain some multiple of k latches, therefore the value $\sum_{i=0}^{w(c)-1}\left(E_{P(l_i),P(l_{i+1})} - P\right)$ reduces to $\frac{w(c)}{k}T_\Phi - w(c)P$. If cycle constraints are satisfied and the previous departure time of a signal was prior to the setup period (which must be guaranteed independently by path constraints), then the signal's arrival after traversing the cycle will be prior to the setup period as well. Thus required setup time S does not appear as a parameter in cycle constraints. ■

Using a maximum-ratio-cycle algorithm such as the one in [1] we solve for the maximum value of $\frac{d(c)}{w(c)}$ over all circuit cycles. This in turn identifies the critical cycle bound, or the minimum possible value of T_Φ to which the circuit can be retimed. To identify the minimum period possible through retiming, a search is performed at and above this bound for the minimum period at which a retiming satisfying path constraints can be found.

5.3 Critical Paths

As in edge-triggered retiming, identifying the critical paths in the circuit graph instead of enumerating the constraints for all paths is crucial to finding a polynomial time bound on the algorithm. Lemma 5 provides a characteristic of critical paths which allows them to be identified using an all-pairs-shortest-paths algorithm.

Lemma 5: (modified from Lemma 5.5 [7]) *A path $u \overset{p}{\twoheadrightarrow} v$ in a well-formed circuit is a critical path iff:*

$$\{w(p)\left(\frac{T_\Phi}{k} - P\right) - d(p)\} \leq \{w(q)\left(\frac{T_\Phi}{k} - P\right) - d(q)\} \text{ for all } u \overset{q}{\twoheadrightarrow} v.$$

Proof sketch. Clock skew and setup time reduce the slack equally along all paths between two vertices, and thus do not affect which path is critical. ■

The weight and delay of critical paths are again defined as $W(u,v)$ and $D(u,v)$ respectively. The critical fanin and fanout edges are again those for which:

$$\begin{aligned} \sigma_{\mathrm{rmax}}(u) &= \max\{\sigma_r(e) \,:\, e \in \mathrm{fanin}(u)\} \\ \sigma_{\mathrm{fmin}}(v) &= \min\{\sigma_f(e) \,:\, e \in \mathrm{fanout}(v)\}. \end{aligned}$$

Substitution of these values into Theorem 2 and combining that result with Theorem 3 leads to Corollary 6, which defines all timing requirements which must be satisfied for correct timing of a symmetric-phase, level-clocked circuit:

Corollary 6: *A well-formed, level-clocked circuit graph G, using a symmetric, k-phase clock schedule is correctly timed if and only if the weights $W(u,v)$ of all critical path are bounded by:*

$$W(u,v) \geq \left(\frac{D(u,v) - T_\phi + \sigma_{\mathrm{rmax}}(u) - \sigma_{\mathrm{fmin}}(v) + 2\sigma_\epsilon + S}{\frac{T_\Phi}{k} - P}\right) - 1.$$

and, for all cycles c:

$$T_\Phi \geq k\left(\frac{d(c)}{w(c)} + P\right).$$

Proof omitted.

For simplicity, several parameters from Corollary 6 can be combined into a single value which represents the effective delay of a path:

$$\gamma(u,v) = D(u,v) + \sigma_{\mathrm{rmax}}(u) - \sigma_{\mathrm{fmin}}(v) + S. \qquad (5)$$

Using γ, the path constraints of Corollary 6 can be written as:

$$W(u,v) \geq \left(\frac{\gamma(u,v) - T_\phi}{\frac{T_\Phi}{k} - P}\right) - 1. \qquad (6)$$

The ceiling of this value is the minimum number of latches required along the critical path between vertices u and v and is referred to as $L(u,v)$. ILP constraints are now formed as

$$r(u) - r(v) \leq W(u,v) - L(u,v).$$

Mixed-ILP constraints may also be formed and the more efficient solution technique ($O(|V|^2 \log |V|)$) for them used [3]. The formulation may also be extended for unequal phase schedules as shown in [7] and solved using extensions to the Bellman-Ford algorithm in $O(k \cdot |V|^3)$ time.

6 Making Circuits Robust to Parameter Variations

Retiming is usually cast as a way to find the minimum clock period for a circuit, but often the problem is to meet some goal clock period rather than find the minimum period. In this section, we show that we can use extra freedom in the clock period to make a circuit robust with respect to clock skew variations.

Tolerance to parameter variation is defined as the maximum amount by which the *actual* parameter values can vary without a resulting constraint violation. Operating a particular circuit at a faster clock period naturally implies less tolerance to parameter variation. Determining a circuit's tolerance to variation in a particular parameter involves enumerating the path and cycle maximum delay constraints that depend on that parameter. The tolerance is the amount of "slack" in the most constraining of these constraints.

The advantages of identifying the most robust circuit retiming are clear; each of the parameters used in the circuit model, including critical path delay $D(u,v)$, clock skew values $\sigma_r(u)$ and $\sigma_f(v)$, latch setup S, and propagation delay P, may vary from the expected value either due to poor estimates or variations in the implemented circuit. If a timing constraint is violated in the actual circuit because a delay exceeds the modeled values used in its retiming, the circuit will fail to operate correctly.

Retiming cannot increase the tolerance in parameters involved in cycle constraints in level-clocked circuits because it cannot change the number of latches on a cycle. Thus tolerance of a cycle constraint to parameter variation is simply $\frac{w(c)T_\Phi}{k} - d(p)$. Similarly an edge-clocked circuit most tolerant of parameter error is found by retiming to the minimum cycle period and then operating at a higher speed. However, retiming level-clocked circuits can increase the slack in path constraints by increasing the number of latches on the path. Moreover, as shown in Theorem 3 and Corollary 4, clock skew is not involved in cycle constraints but is included in path constraints. Thus, tolerance to variation in clock skew can be improved by retiming paths even if the tolerance to delay variation is limited by cycle delays.

6.1 An Algorithm to Generate Robust Circuits

To improve circuit tolerance to clock skew variations we add a tolerance, τ, to the effective path delay defined in Eqn. 5. Using τ, the required path weight in Eqn. 6 is written:

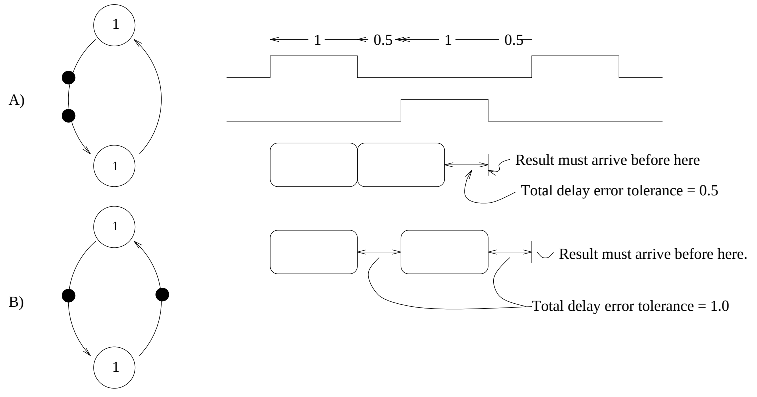
$$W(u,v) \quad \geq \quad \left(\frac{\gamma(u,v) + \tau - T_\phi}{\frac{T_\Phi}{k} - P} \right) - 1.$$

An increase in τ increases effective path delay as would be caused by an increase in variable clock skew, an increase in the relative skew between rising and falling edges, or an increase in the setup time. For a given clock period, if a circuit is retimed with a value of τ, then that circuit can tolerate variations in these parameters summing to τ.

A circuit retimed with some value of τ can also tolerate increases in path and latch propagation delays; however, changing either of these delay parameters may also change which paths are critical. Thus a retiming using τ does not necessarily guarantee a circuit that can tolerate this much variation in element delays.

To find a circuit retiming that tolerates the most clock skew variation, we fix T_Φ to the desired clock period and search over increasing values of τ for the maximum one at which a retiming can be found. Finding the maximum value of τ is equivalent to finding the minimum clock period unless a critical cycle bound limits the minimum clock period. Further decreases of T_Φ are prevented because negative weight cycles appear in the slack graph on which a shortest path algorithm is used to identify critical paths. Thus it may be possible to increase τ further but not to further decrease T_Φ. In other words, retiming to the fastest clock possible and then running with a slower clock does not give the most robust circuit when the cycle constraints determine the optimal clock period.

Increased circuit tolerance is illustrated in Figure 7. Using the clock schedule shown with increasing values of τ causes the initial circuit in (A) to be retimed to the one in (B). Even though the circuit in (B) cannot run faster because some other constraint defines the clock period, it can tolerate a significantly greater amount of clock skew than the one in (A). Because there are no changes in critical paths, the increased tolerance to error in vertex delay estimates is also clear.

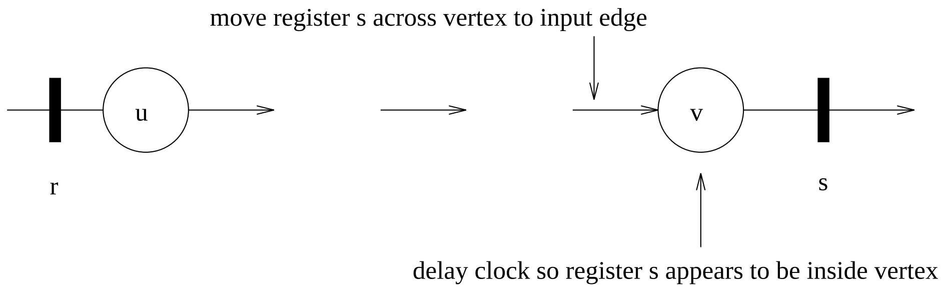


1
0.5
1
0.5
A)
B)
1
Result must arrive before here
Total delay error tolerance = 0.5
Result must arrive before here.
Total delay error tolerance = 1.0

Figure 7: *A simple example showing the tolerance gain from optimizing circuit paths. Each circuit operates correctly under the given schedule; however the circuit in A) can only tolerant a total of 0.5 units of delay estimation error in the two nodes while B) can tolerate 0.5 units error in each of the nodes. (The clock period shown here is determined by some other cycle in the circuit.)*

7 Retiming with Forced Clock-Skew

This section presents preliminary results of an algorithm which combines retiming via register movement with retiming via intentional added clock skew. Retiming using clock skew allows the "virtual" movement of registers on a finer grain than the usual retiming [2]. Using the two techniques together allows the greatest flexibility in synchronizer placement and potentially faster circuit designs. Particularly interesting is the resulting ability of edge-triggered circuits to achieve speeds comparable to level-clocked designs. We describe here an algorithm for combined retiming of edge-triggered circuits. We are currently extending these techniques to level-clocked circuits as well. For simplicity, we remove the synchronizer parameters from our maximum delay constraint and return to the original requirement on edge-triggered circuits that depends only on the delay of functional units. That is, the delay between two adjacent registers must be less than the clock period. This requirement may be satisfied either by moving registers closer together by retiming, or by skewing the clock signals to the registers to lengthen the clock period for just that pair (see Figure 8). The smallest amount by which retiming can increase the available time for a delay path is one full clock period by adding one register to the path. But clock skew can be used to increase the available time by amounts up to one clock period. Thus the two techniques are complementary, with small amounts of time being added by skewing the clock signals and larger amounts of time added by moving registers.



move register s across vertex to input edge
u
v
r
s
delay clock so register s appears to be inside vertex

Figure 8: *The delay between registers r and s may be reduced by moving s across the vertex physically with retiming or virtually by delaying its clock signal.*

We assign a value $\nu(e)$ to each edge, where $T_\Phi \cdot \nu(e)$ is the amount by which the clock signal to the *first* register placed on the edge, starting from the edge input, is skewed. Because there is no delay between registers on an edge, no subsequent register on the same edge requires a skewed clock to prevent late signal arrival. Furthermore, skewing a subsequent register unnecessarily shortens the time available to paths following it. $\nu(e)$ is limited in value to the range $0 \leq \nu(e) < 1$. Thus clock signals are delayed relative to the base clock by an amount up to the clock period. In addition, we require that for each edge $e \in \text{fanout}(e)$, $\nu(e) = \nu(v)$ so that we can associate the skew value $\nu(v)$ with vertices and not edges. Skewing all the registers following a vertex by the same amount has the effect of virtually moving the registers inside the vertex. Note, however, that not all edges following a vertex are required to have a register.

The difference in added clock skew at two successive registers determines how much extra time is available to the vertices between them. We first extend maximum delay constraints to included intentional skew, omitting here the other synchronization parameters:

Theorem 7: *An edge-triggered circuit G is correctly timed iff, for all paths $u \xrightarrow{p} v$ such that $w(p) = 0$ and edges $e \in fanin(u)$ and $\acute{e} \in fanout(v)$, $d(p)$ is bounded by:*

$$d(p) \leq T_\Phi(1 - \nu(e) - \nu(\acute{e}))$$

Proof Omitted.

We now have circuit paths $u \xrightarrow{p} v$ with retiming variables $r(u)$, $r(v)$ and skew $\nu(v) \cdot T_\Phi$ assigned to each edge in the fanout of v. For each path p and edge $e \in \text{fanin}(u)$, a constraint is written:

$$(r(u) - \nu(e)) - (r(v) + \nu(v)) \quad \leq \quad W(u,v) - \frac{D(u,v)}{T_\Phi} \qquad (7)$$

Our algorithm uses an extension of the reversed Bellman-Ford (RevBF) relaxation technique [6] to solve constraint sets of the form in Eqn. 7. The RevBF method satisfies difference of two variable constraints, such as $r(u) - r(v) \leq C(u,v)$, by setting the value of $r(v) = r(u) - C(u,v)$ (instead of setting $r(u) = C(u,v) - r(u)$ as in the standard Bellman-Ford approach). The RevBF algorithm was developed in our work on retiming unequal-phase level-clocked circuits which used constraint weights dependent on one of the two constrained variables because timing constraints dependent on $r(u)$ could be expressed with greater ease than those depending on $r(v)$. For constraints of the Eqn. 7, we adjust register placement by assigning a new value to $r(v)$ which satisfies the integral part of $(r(u) - \nu(e)) - (W(u,v) - \frac{D(u,v)}{T_\Phi})$ and then adjust the clock skew by setting $\nu(v)$ to satisfy the real part. Thus all edges in the fanout of each vertex have the same value of ν while edges in its fanin may have differing values.

To show that races cannot occur because of short paths, we must show that the E2' minimum delay constraint will not be violated following an assignment of intentional skew to circuit edges and any retiming of registers. In particular, we show that the constraints on values of $\nu(e)$ ensure that E2' is not violated.

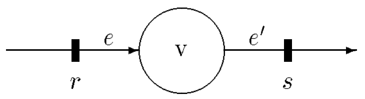


Figure 9: *A vertex with surrounding latches.*

If two registers r and s are placed across any vertex in the circuit as shown in Figure 9 (a placement possible under retiming), constraint E2' requires:

$$E2' : A_s = d(v) + P > t_{hold} + \sigma(e') - \sigma(e) + \sigma_\epsilon. \qquad (8)$$

Intentional skew produces a circuit in which $\sigma(e) = \nu(e)T_\Phi$ for each edge. Thus Eqn. 8 requires:

$$d(v) + P \quad > \quad t_{hold} + \nu(e')T_\Phi - \nu(e)T_\Phi + \sigma_\epsilon.$$

In the worst case, $\nu(e) = 0$, resulting in a bound on $\nu(e')$ of:

$$\nu(e') < \frac{d(v) + P - t_{hold} - 2\sigma_\epsilon}{T_\Phi},$$

in addition to the existing requirement that $\nu(e') < 1$. A value of $\nu(e')$ greater than $\frac{d(v)}{T_\Phi}$ is never assigned by our algorithm because it exceeds the amount of time necessary to satisfy the vertex delay. That is, any greater constraint would instead move the register across the vertex by setting $r(v) = 1$. Note that the worst case for races occurs when the maximum value is assigned to $\nu(e') = \frac{d(v)}{T_\Phi}$ and the constraint reduces to $P > t_{hold} + 2\sigma_\epsilon$, which is the familiar skew constraint for edge-clocked circuits. Note that this also describes the case where two or more registers appear on an edge, that is when $d(v) = 0$ and $\nu(e') = 0$.

Figures 10 and 11 provide two correlator circuit examples of edge-triggered circuits with intentional clock skew. In Figure 10 includes timing constraint across the host vertex resulting in cycles which limit the minimum clock period to 10. Level-clocked circuits are able to achieve this bound [7], while edge-triggered retimings without delayed clock signals are have a minimum period of $T_\Phi = 13$ [4]. By using delayed clock signals the circuit in Figure 10 has a minimum period of $T_\Phi = 10$, equal to the cycle imposed lower bound.

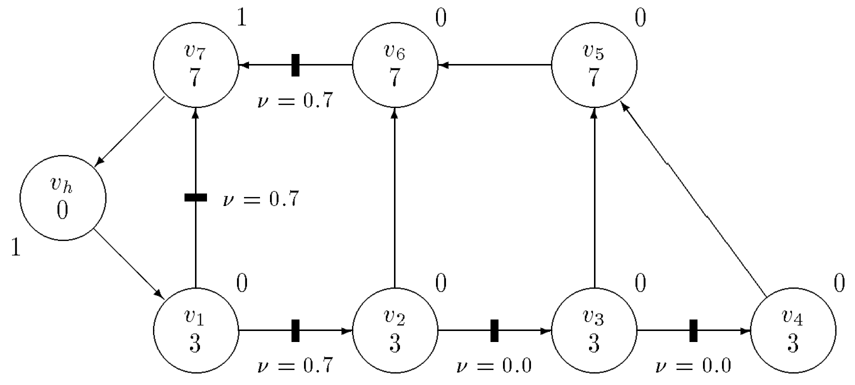


Figure 10: *An edge-triggered correlator retiming with intentional clock skew. Cross host timing constraints are imposed. The optimal period is $T_\Phi = 10$.*

In Figure 11, no timing constraints across the external environment are imposed and thus there are no circuit cycles. This may be thought of as the inclusion of a register inside the host vertex controlled by a clock signal which cannot be delayed. Without intentionally skewed clocks, the minimum edge-triggered period is $T_\Phi = 9$ while the minimum symmetric, level-clocked period is $T_\Phi = 8$. The edge-triggered example in Figure 11 achieves a minimum period of $T_\Phi = 7.5$ though the use of skewed clock arrival time.

As in the standard Bellman-Ford algorithm, we conjecture that the relaxation routine still has $O(1)$ time complexity and that it must be applied $|V|$ times to each ILP constraint. Each of the $O(|V|^2)$ critical paths has an average of $\frac{|E|}{|V|}$ fanin edges and a single constraint must exist for each fanin edge and critical path. Thus the overall number of constraints is $O(|V|^2 \frac{|E|}{|V|})$. Real circuit graphs are typically sparse and $\frac{|E|}{|V|} = O(1)$, resulting in a linear increase in the expected running time of the algorithm. The Mixed-ILP and FEAS algorithms presented in [5] may also be modified to solve these problems with a linear increase in expected running time.

8 Summary

We have presented in this paper techniques for including clock skew parameters in the retiming algorithms for both edge-clocked and level-clocked circuits. While this allows us to handle clock skew in many situations, there is still much work left to be done. We have described how to satisfy minimum delay constraints using

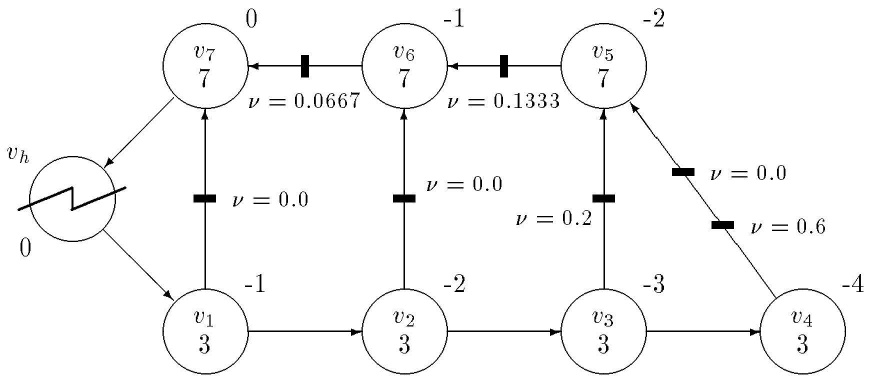


Figure 11: *An edge-triggered correlator retiming with intentional clock skew. Cross-host timing constraints are prevented by assuming that a pair of registers are hidden inside the host vertex. The optimal period is* $T_\Phi = 7.5$.

a static analysis, but a more general approach would incorporate minimum delay constraints in the retiming algorithm itself. Perhaps the most interesting problem is extending the combined intentional skew and retiming algorithm to level-clocked circuits. We conjecture that such an algorithm in conjunction with a reasonable approach to minimum delays will allow us to retime circuits to achieve the same performance in the presence of variable skew as in the case when the clock skew is zero.

References

[1] S. Burns. *Performance Analysis and Optimization of Asynchronous Circuits.* PhD thesis, California Institute of Technology, 1991. Caltech-CS-TR-91-01.

[2] J. P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, 39(7):945–951, 1990. Correspondence.

[3] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. Optimizing two-phase, level-clocked circuitry. In *Advanced Research in VLSI and Parallel Systems: Proc. of the Brown/MIT Conference*, pages 245–264, 1992.

[4] C. E. Leiserson, F. Rose, and J. B. Saxe. Optimizing synchronous circuitry by retiming. In *Proc. of the 3rd Caltech Conference on VLSI*, Mar. 1983.

[5] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5–35, 1991. Also available as MIT/LCS/TM-372.

[6] B. E. Lockyear and C. Ebeling. Retiming of multi-phase, level-clocked circuits. Technical Report 91-10-1, University of Washington, Dept. of Computer Science, Oct. 1991.

[7] B. E. Lockyear and C. Ebeling. Retiming of multi-phase, level-clocked circuits. In *Advanced Research in VLSI and Parallel Systems: Proc. of the Brown/MIT Conference*, pages 265–280, Mar. 1992.

[8] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun. Analysis and design of latch-controlled synchronous circuits. In *Proc. 27th ACM-IEEE Design Automation Conf.*, 1990.