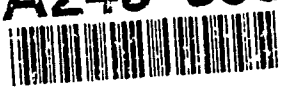





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

*University of Washington*

*Seattle 98195*



SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: 91-10-01 | 2. GOVT ACCESSION NO. |
| 3. RECIPIENT'S CATALOG NUMBER | |
| 4. TITLE (and Subtitle): Optimal Retiming of Multi-Phase, Level-Clocked Circuits | 5. TYPE OF REPORT & PERIOD COVERED: Technical |
| 6. PERFORMING ORG. REPORT NUMBER | |
| 7. AUTHOR(s): Brian Lockyear and Carl Ebeling | 8. CONTRACT OR GRANT NUMBER(s): N00014-91-J-4041 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Northwest Laboratory for Integrated Systems, University of Washington, Dept. of Comp. Science, FR-35 Seattle, WA 98195 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: DARPA-ISTO, 1400 Wilson Boulevard, Arlington, VA 22209 | 12. REPORT DATE: October, 1991 |
| | 13. NUMBER OF PAGES: 35 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Office of Naval Research - ONR, Information Systems Program - Code 1513: CAF, 800 North Quincy Street, Arlington, VA 22217 | 15. SECURITY CLASS. (of this report) |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

retiming, level-clocked circuits, circuit optimization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Using level-sensitive latches instead of edge-triggered registers for storage elements in a synchronous system can lead to faster and less expensive circuit implementations. This advantage derives from an increased flexibility in scheduling the computations performed by the circuit. In edge-clocked circuits, the amount of time available for the computation between two registers is precisely the length of the clock cycle, while in circuits using level-sensitive latches a computation can borrow time across latches, thus reducing the amount of dead time in the clock cycle. In either type of circuit, achieving maximum

CONT.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Cont.

-- 1 OF 1
-- 1 - AD NUMBER: AE43958
--27 - ABSTRACT: USING LEVEL-SENSITIVE LATCHES INSTEAD OF EDGE-TRIGGERED
-- REGISTERS FOR STORAGE ELEMENTS IN A SYNCHRONOUS SYSTEM CAN LEAD TO
-- FASTER AND LESS EXPENSIVE CIRCUIT IMPLEMENTATIONS. THIS ADVANTAGE
-- DERIVES FROM AN INCREASED FLEXIBILITY IN SCHEDULING THE
-- COMPUTATIONS PERFORMED BY THE CIRCUIT. IN EDGE-CLOCKED CIRCUITS THE
-- AMOUNT OF TIME AVAILABLE FOR THE COMPUTATION BETWEEN TWO REGISTERS
-- IS PRECISELY THE LENGTH OF THE CLOCK CYCLE, WHILE IN CIRCUITS USING
-- LEVEL-SENSITIVE LATCHES A COMPUTATION CAN BORROW TIME ACROSS
-- LATCHES THUS REDUCING THE AMOUNT OF DEAD TIME IN THE CLOCK CYCLE.
-- IN EITHER TYPE OF CIRCUIT, ACHIEVING MAXIMUM PERFORMANCE REQUIRES
-- LOCATING THE STORAGE ELEMENTS IN SUCH A WAY AS TO SPREAD THE
-- COMPUTATION UNIFORMLY ACROSS A NUMBER OF CLOCK CYCLES. RETIMING IS
-- THE PROCESS OF REARRANGING THE STORAGE ELEMENTS IN A CIRCUIT TO
-- REDUCE THE CYCLE TIME OR THE NUMBER OF STORAGE ELEMENTS WITHOUT
-- CHANGING THE INTERFACE BEHAVIOR OF THE CIRCUIT AS VIEWED BY AN
-- OUTSIDE HOST. RETIMING IN EFFECT RESCHEDULES THE CIRCUIT
-- COMPUTATIONS IS TIME BASED ON THE LENGTH OF THOSE COMPUTATIONS. IN
-- THIS PAPER, WE EXTEND THE RETIMING TECHNIQUES DEVELOPED FOR EDGED-
-- CLOCKED CIRCUITS BY LEISERSON. ROSE AND SAXE TO A GENERAL CLASS OF
-- MULTI-PHASE, LEVEL-CLOCKED CIRCUITS. WE FIRST DESCRIBE THIS CLASS
-- OF WELL-FORMED CIRCUITS AND DEFINE WHAT IT MEANS FOR A WELL-FORMED,

-
-- LEVEL-CLOCKED CIRCUIT TO OPERATE CORRECTLY. WE THEN SHOW THAT A SET
-- OF CONSTRAINTS CAN THEN CBE USED TO RETIME A LEVEL-CLOCKED CIRCUIT
-- USING EFFICIENT INTEGER LINEAR PROGRAMMING TECHNIQUES SIMILAR TO
-- THOSE USED FOR EDGE-CLOCKED CIRCUITS.

(2)

# Optimal Retiming of Multi-Phase, Level-Clocked Circuits[1]

**Brian Lockyear and Carl Ebeling**

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

Technical Report 91-10-01
October, 1991





[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

Using level-sensitive latches instead of edge-triggered registers for storage elements in a synchronous system can lead to faster and less expensive circuit implementations. This advantage derives from an increased flexibility in scheduling the computations performed by the circuit. In edge-clocked circuits the amount of time available for the computation between two registers is precisely the length of the clock cycle, while in circuits using level-sensitive latches a computation can borrow time across latches thus reducing the amount of dead time in the clock cycle. In either type of circuit, achieving maximum performance requires locating the storage elements in such a way as to spread the computation uniformly across a number of clock cycles.

Retiming is the process of rearranging the storage elements in a circuit to reduce the cycle time or the number of storage elements without changing the interface behavior of the circuit as viewed by an outside host. Retiming in effect reschedules the circuit computations in time based on the length of those computations. In this paper, we extend the retiming techniques developed for edge-clocked circuits by Leiserson, Rose and Saxe to a general class of multi-phase, level-clocked circuits. We first describe this class of well-formed circuits and define what it means for a well-formed, level-clocked circuit to operate correctly. We then show that a set of constraints can be efficiently derived for a circuit which preserve its correctness under retiming. These constraints can then be used to retime a level-clocked circuit using efficient integer linear programming techniques similar to those used for edge-clocked circuits.


## 1 Introduction

Synchronous circuits rely on clocked storage elements to hold values while computation is performed on them. The most widely used storage element is the edge-triggered register which samples its input at the beginning of each clock period, holding that value for the entire clock period. Edge-triggered registers provide a straightforward way to analyze the minimum clock period of a circuit by determining the maximum delay between any two registers. This simplified timing analysis leads to efficient retiming techniques for adjusting the placement of registers to optimize the cycle time or the number of registers [6, 7].

Level-clocked circuits are synchronous circuits that use level-sensitive latches. These latches are clocked storage elements that allow the inputs to flow through the latch during the active phase of the clock, latching the value during the inactive phase. In level-clocked circuits it is less clear how much time is available to the computation placed between latches because the input values may arrive early and flow through the input latch. This borrowing of time between clock cycles makes the determination of the constraints on the clock period difficult. However, the flexibility in scheduling the computation provides more opportunity to optimize the clock period than in the case of edge-clocked circuits.

This difference in scheduling is shown by the example circuits in Figures 1 and 2 where the same computation is implemented using an edge-clocked circuit in the first case and a level-clocked in the second. The circuit of Figure 2 uses a two-equal-phase, non-overlapping clock with each edge-triggered register replaced by a pair of $\phi_1$, $\phi_2$ latches. The edge-clocked circuit of Figure 1 shows an optimal placement of registers which achieves a cycle time of $T_\Phi = 8$. By contrast, the level-clocked circuit of Figure 2 shows an optimal placement of latches that achieves a cycle time of $T_\Phi = 6$. This level-clocked circuit is also cheaper. Assuming that the cost of an edge-triggered

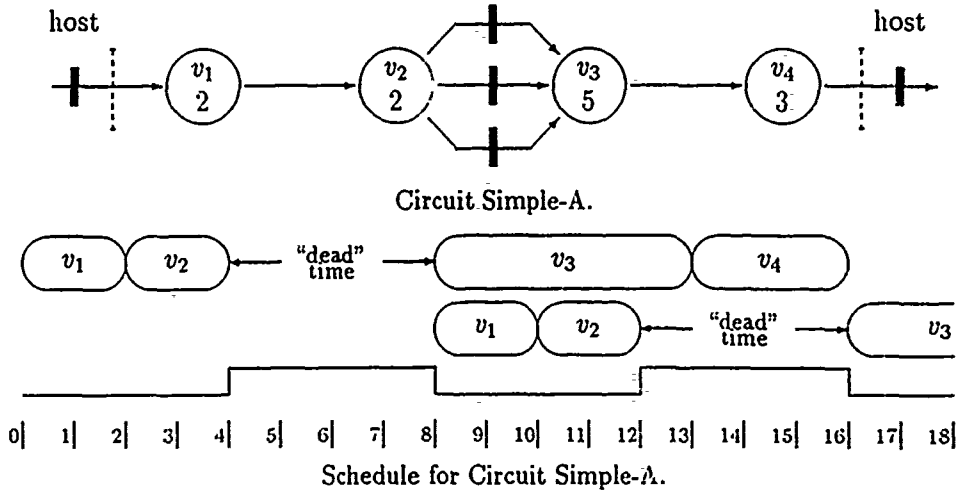


Figure 1: *A simple circuit optimally timed using edge-triggered registers and the resulting clock schedule.*

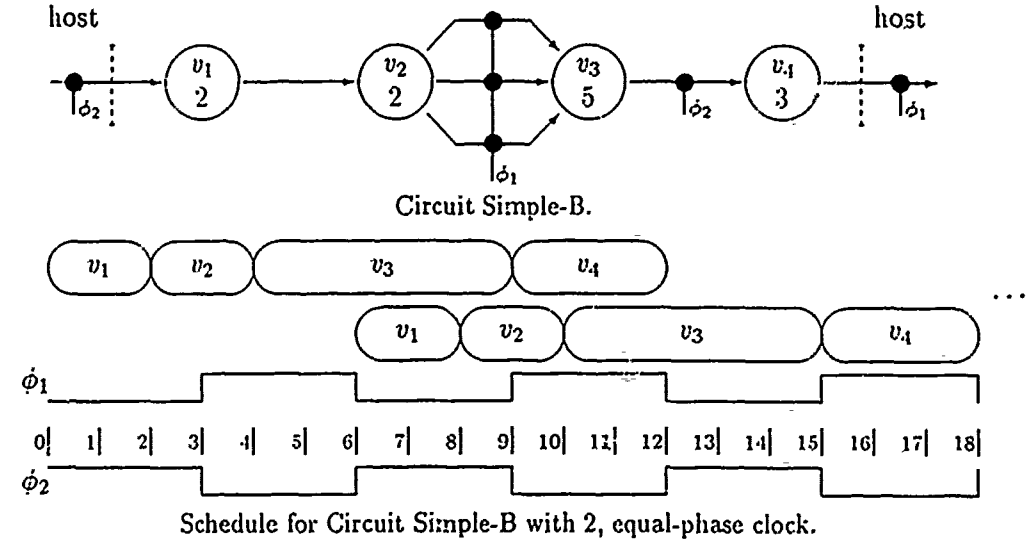


Figure 2: *The simple circuit now timed using level-sensitive latches.*

register is $R$ and that of a latch is $\frac{R}{2}$, then the storage element cost for the edge-clocked circuit is $3R$ while that of the level-clocked circuit is only $2R$. The flexibility provided by level-sensitive latches can also be used to reduce cost while meeting a particular clock period. For instance the circuit in Figure 2 is to be retimed with $P = 7$, the latches on the edge between nodes $v_2$ and $v_3$ may be moved to the single edge $v_1 \rightarrow v_2$ reducing the storage element cost to $R$, one-third that of the slower optimal edge-clocked circuit.

In this paper we show how the retiming techniques developed by Leiserson et. al. for edge-clocked circuits can be extended to optimize level-clocked circuits. Figure 3 shows the edge-clocked correlator example from their paper in its original state, which can be retimed to the circuit in Figure 4 which operates with a clock period of 13. We can convert the circuit of Figure 3 into an equivalent level-clocked circuit by using a two-equal-phase, non-overlapping clock schedule and

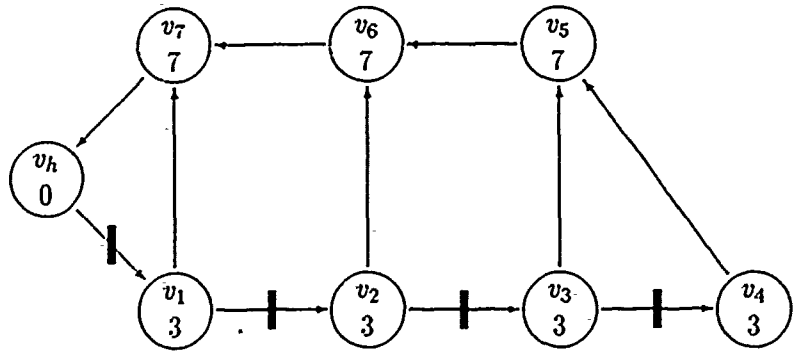


Figure 3: *The correlator circuit from Leiserson et. al. in its initial configuration with registers shown as solid bars.*

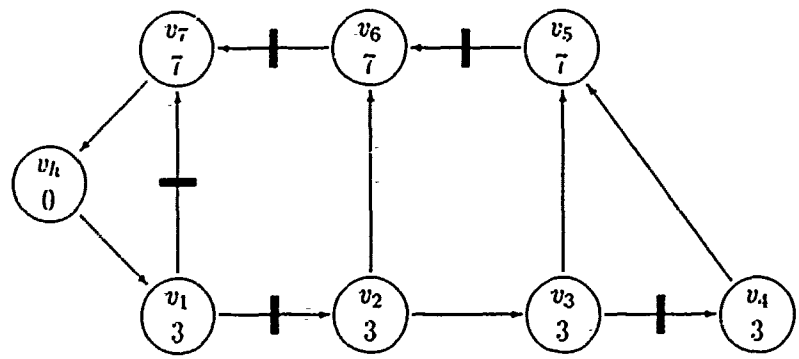


Figure 4: *The edge-clocked correlator circuit optimally retimed to a clock period* $T_\Phi = 13$.

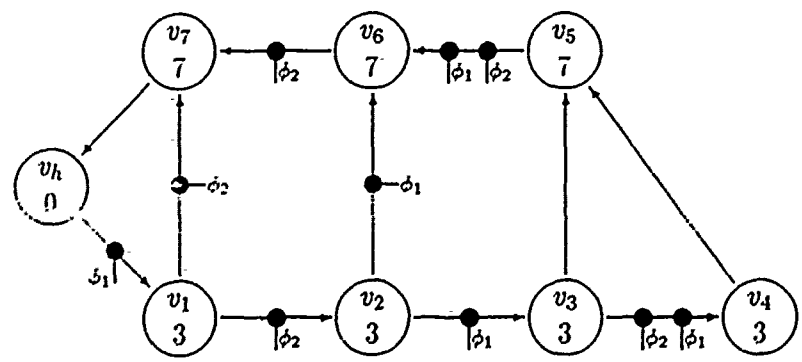


Figure 5: *The correlator circuit optimally retimed using a two-equal-phase clock. Latches are represented by solid circles and marked with controlling clock phase. The resulting clock cycle time is 10 units.*

replacing each register with a pair of $\phi_1, \phi_2$ latches. This circuit can be retimed to the one in Figure 5 using the retiming techniques described in this paper to achieve an optimal clock period of 10. The retiming techniques we describe also handle more complex clock schedules with multiple phases and phase overlap and underlap. For example, retiming the correlator example to a two-equal-phase clock with 10% underlap between phases achieves a clock period of 10.4. These techniques can also be extended to clock schedules with unequal length phases through a technique of adding tightly constrained variables to the system which contain information regarding the current phase of nodes in the circuit graph.

There are a number of obstacles to level-clocked retiming each of which is explored in this paper. These include:

- *Circuit Correctness:* The definition of a correctly operating circuit may vary widely depending on latch phasing and clock schedule. The variety of clocking strategies possible causes a general retiming technique for level-clocked circuits to be much more complex than required for typical cases. We restrict the techniques in this paper to common circuit structures and take corresponding advantage of those structures to simplify the retiming techniques.

- *Minimum vs Maximum Delay Constraints:* An issue related to circuit correctness, some circuit structures combined with particular clock schedules impose minimum as well as maximum delay constraints on combinational logic paths. In this work we restrict legal circuits and clock schedules such that this additional complexity does not arise.

- *Identification of Critical Cycles:* As demonstrated in Figure 2. time allocated to a combinational logic block may be shared across the active period of a latch. We will show that path based constraints which allow the flexibility to share across latches do not sufficiently bound the computational time available around a cycle. Instead cycles in the circuit graphs form an independent lower bound on possible clock periods. We provide a technique for identifying this lower bound initially so that only path-based constraints need be considered above the Critical Cycle period.

- *Identification of Critical Paths:* An additional impact of computational time sharing is that critical paths between two nodes in a circuit graph may differ from those identified for edge-clocked retiming. Moreover, critical paths in a level-clocked graph are not the same for all clock periods. We provide a new definition of critical paths necessary for correct retiming of level-clocked graphs and provide a technique for identifying the critical paths based on that definition.

- *Constraints on Higher Weight Paths:* Computational time sharing requires constraints on paths of non-zero weight in the circuit graph which are redundant to constraints on zero weight paths as they were in edge-clocked circuits. Techniques for correctly generating higher order constraints are provided.

- *Constraints Dependent on Phase of Latch Placement:* In clock schedules utilizing unequal phases, the maximum delay constraints for a given computation path may differ depending on the phase of latches placed along the path. Techniques for writing constraints that correctly restrict maximum delay dependent on latch placement are provided as well as modifications of existing algorithms required to efficiently solve the more complex constraint sets.

### 1.1 Overview of the Paper

We first review the work of Leiserson et. al. [6, 7] on which our work is based and present the underlying circuit graph model. Next we review the clock model we have adopted from the work of Sakallah, Mudge and Olukotun [5]. We then describe the class of well-formed level-clocked circuits to which we will be limited and define what it means for a level-clocked circuit to operate correctly. In Section 5 we then use this model to derive the set of constraints that fully specify the multi-phase, maximum delay timing restrictions of level-clocked circuits. Section 6 applies these timing restrictions to circuits using multi-phase clocks with equal phases to form sets of ILP constraints which restrict the movement of latches through circuit graphs. Finally Section 7 extends our techniques to handle valid clock schedules with arbitrary length phases.

## 2 Background

In this section, we briefly review the terminology and graph model of digital circuits described in Leiserson et. al. [6] and extend it to handle level-sensitive latches. We then review the basic retiming results of their paper. The reader is encouraged to read [6, 7] for full details.

A circuit is modeled as a directed multigraph $G = (V, E, w, d, s)$ whose vertices $V$ model the functional elements of the circuit and whose edges $E$ model the interconnections between the functional elements. Each vertex $v$ is given a delay $d(v)$ that is associated with the corresponding functional element. A unique host vertex $v_h$ with $d(v_h) = 0$ is used to represent the environment of the circuit. Each edge is given a weight $w(e)$ which is the number of registers along the connection. This notion of edge weight is sufficient for edge-clocked circuits which use a single register type, but must be extended for level-clocked circuits which use latches controlled by different clock phases. We do this by associating with each edge $e$ the sequence $s(e) = (l_1, l_2, \ldots, l_{w(e)})$ of latches along the connection.

The notation $u \xrightarrow{e} v$ is used to represent an edge $e$ from vertex $u$ to vertex $v$. A path in the circuit graph is a sequence of vertices and edges from a vertex $u$ to a vertex $v$ and is denoted by $u \xrightarrow{p} v$. A *simple* path contains no vertex twice. For level-clocked circuits, we also refer to paths that begin at a latch $l$ and end at a latch $m$ for which we use the notation $l \xrightarrow{p} m$.

The weight $w(p)$ of a vertex terminated path $p = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{k-1}} v_k$ is the count of registers or latches along the path, that is, the sum of the edge weights along the path: $w(p) = \sum_{i=0}^{k} w(e_i)$. We define the sequence of latches along the path with $k$ edges to be the concatenation of the edge latch sequences along the path: $\|_{i=0}^{k-1} s(e_i)$. Thus for a vertex terminated path $p$, $w(p) = |s(p)|$.

For a latch terminated path $p = \xrightarrow{e_{-1}} v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{k-1}} v_k \xrightarrow{e_k}$ that begins at a latch $l \in s(e_{-1})$ and ends at a latch $m \in s(e_k)$, the path latch sequence $s(p)$ begins with the tail of $s(e_{-1})$ (beginning with $l$) and ends with the head of $s(e_k)$ (ending with $m$). Unlike the vertex terminated path, the weight $w(p)$ of a latch terminated path $l \xrightarrow{p} m$ is defined to be $|s(p)| - 2$; that is, the initial and final latches are not included in the path weight.

The weight of a vertex terminated cycle $c = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{k-1}} v_0$ is identical to the weight of the same sequence of edges and vertices treated as a path. However, in the case of a cycle beginning and ending at a latch $l$, the weight of a path $w(l \xrightarrow{p} l)$ does not include the beginning and ending latch. Thus $w(c) = w(p) + 1$ where $c$ is a cycle beginning and ending at latch $l$ and $p$ is the same cycle treated as a path beginning and ending at latch $l$.

The delay $d(p)$ of a path is the sum of the delays of the vertices along the path: $d(p) = \sum_{i=0}^{k} d(v_i)$. The delay, $d(c)$, of a cycle $c = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{k-1}} v_0$ includes the delay of node $v_0$ only once, hence $d(c) = \sum_{i=0}^{k-1} d(v_i)$.

## 2.1 Correct Operation

In order to retire a circuit, whether edge-clocked or level-clocked, a definition of correct operation must exist. This allows an initial circuit graph to have registers moved within it and to able to determine using the definition whether the result is operating correctly with respect to the initial circuit. For edge-clocked circuits, a simple definition of correct operation is used for retiming which requires that the following conditions be maintained:

- C1. For any path $p$ in $G$, if $d(p) >$ clock period, then $w(p) \geq 1$.
- C2. For any cycle $c$ in $G$, $w(c) \geq 1$.

Retiming a circuit is the process of transforming a circuit graph $G$ into another graph $G_r$ by relocating registers (or latches) such that the input/output behaviors of $G$ and $G_r$ are identical. Transforming a circuit $G$ into a corresponding retimed circuit $G_r$ can be viewed as assigning a *retiming* (or lag) value $r(v)$ to each of the vertices of $G$. This retiming value represents the number of registers (latches) removed from the output edges of vertex $v$ and added to the input edges. More formally, for any edge $u \xrightarrow{e} v$. $w_r(e) = w(e) + r(v) - r(u)$.

The movement of registers in retiming introduces an additional aspect of correctness which is the relative difference in the weight of two paths between the same two vertices. For example, assume two distinct paths $u \xrightarrow{p} v$ and $u \xrightarrow{q} v$. In order to preserve the logical structure of the circuit the difference in path weight $w(p) - w(q)$ must be preserved during retiming. Leiserson et. al. show that retiming by assigning retiming values to vertices maintains a constant difference between the weight of paths with the same endpoints and a constant number of registers on any cycle in the graph. Using the same result, correctness condition C2 is also maintained.

The key retiming result of [6, 7] defines the following set of constraints which must be met by a legal retiming of an edge-clocked circuit graph using a clock period $c$. These constraints are given in terms of the maximum delay along the critical paths in $G$. A critical path in an edge-clocked circuit graph is defined as a minimum-weight path of maximum delay from $u$ to $v$. In reality what is being identified is a particular path such that if that path is retimed correctly then all other paths between the same two end-points will also be retimed correctly. Note that some sub-paths may not be retimed correctly but that fact will be detected independently of the overall path. The edge-clocked definition of critical path is used to define the matrices $W$ and $D$:

$$W(u, v) = min\{w(p) \mid u \xrightarrow{p} v\}.$$

The maximum delay on any critical path from $u$ to $v$ is given by

$$D(u, v) = max\{d(p) \mid u \xrightarrow{p} v \text{ and } w(p) = W(u, v)\}.$$

We will show that the above definitions for are insufficient to identify critical paths in level-clocked circuits and in fact the critical path between two end nodes will vary with the clock period of interest. However the above definitions are sufficient for edge-clocked circuits and using them it is possible to generate a set of constraints on retiming of an edge-clocked circuit such that the

resulting circuit operates correctly under our definition. The constraints on edge-clocked retiming are:

$$\begin{aligned} \text{I/O:} &\quad r(v_h) = 0 \\ \text{Positive edge weight:} &\quad r(u) - r(v) \leq w(e) \text{ for all edges } u \xrightarrow{e} v \\ \text{Maximum path delay:} &\quad r(u) - r(v) \leq W(u,v) - 1 \text{ for all } (u,v) \text{ for which } D(u,v) > c \end{aligned}$$

The I/O constraint maintains the I/O behavior under retiming. Although it is not necessary to require that the host vertex have a retiming value of 0, having the retiming value identified at a particular node is useful in some solution methods. To show that it is not necessary to require $r(v_h) = 0$, note that changing all node retiming values by any constant amount results in the same graph. In other words, since the weight of a retimed edge is $r(u) - r(v)$, if all values of $r(u)$ are changed by a constant amount to a value $\acute{r}(u)$ the resultant edge weight values must be identical: $r(u) - r(v) = \acute{r}(u) - \acute{r}(v)$. Thus for any retiming there is exists an identical circuit graph such that $\acute{r}(v_h) = 0$.

The positive edge weight constraints keep the retiming from assigning negative edge weights which have no physical meaning.[2] The maximum path delay constraints force proper timing by placing at least a single register along any path with delay greater than the clock period of interest. Linear programming techniques can be used to solve this constraint set and return a valid assignment of the retiming variables if one exists. The set of possible optimum clock periods is derived from the delay of critical paths in the graph and a binary search is performed over that set to determine the fastest possible clock period to which the circuit may be retimed.

The host vertex $v_h$ is a zero-delay vertex defined as the source of all circuit inputs and the destination of all circuit outputs. As a result, additional constraints on circuit timing are imposed along paths which pass through the zero-delay host vertex. These cross-host constraints may over-constrain the actual design by implying relationships between output and input signals which are not intended. If such a relationship were intended it should be represented as an explicit edge in the graph rather than an implicit and unavoidable one.

The Correlator circuit example used in this paper retains constraints through the host vertex to allow comparison with [6, 7]. The simple circuits in Figures 1 and 2 omit cross-host constraints and can be thought of as providing implicit registers or latches in the host vertex. Or it may be thought of as dividing the single host vertex into two parts with no edge between them. The I/O constraint can be expanded to prevent retiming of any host vertex. Various additional input and output timing constraints may be represented by placing additional delay vertices on input or output edges and the appropriate constraint on their retiming value.

## 3 Clock Model

We have adopted the clock model of Sakallah, Mudge & Olukotun [5] which provides a convenient way to describe the constraints on multi-phase clocks. A *k-phase clock* is a set of $k$ periodic signals $\Phi = \{\phi_1 \ldots \phi_k\}$ where $\phi_i$ is referred to as *phase i* of the clock $\Phi$. All $\phi_i$ have a common cycle time $T_\Phi$. Each phase divides the clock cycle into two intervals as shown in Figure 6: An *active* interval of duration $T_{\phi_i}$ 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 transitions

[2] In some applications negative edge weights can be useful as an intermediate step [8].

*into* and *out of* the active interval are called the *enabling* and *latching* edges respectively. We refer to the clock phase controlling latch $l$ by $P(l)$.

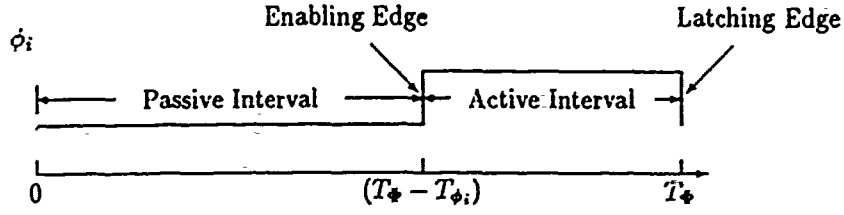


Figure 6: *Diagram from Sakallah et. al. showing a clock phase $\phi_i$ and its local time zone.*

Associated with each phase is a *local time zone* such that its passive interval starts at $t = 0$, its enabling edge occurs at $t = T_\Phi - T_{\phi_i}$, and its latching edge occurs at $T_\Phi$. The domain of the local time zone is defined to be the interval $(0, T_\Phi]$ since the start of the current clock cycle coincides with the end of the previous cycle. Sakallah et. al. additionally introduce an arbitrary *global time reference* and the value $e_i$ which denotes the time relative to the global time reference at which phase $\phi_i$ *ends*.

Phases are ordered relative to the global time reference so that $e_1 \leq e_2 \leq \cdots \leq e_{k-1} \leq e_k$. The global time reference is arbitrarily set such that $e_k \equiv T_\Phi$. The phase sequentially following $\phi_i$ in the clock set is referred to as $\phi_{i+1}$ with phase $\phi_{k+1} \equiv \phi_1$ and $\phi_{1-1} \equiv \phi_k$.

Finally a *phase shift* operator is defined:

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

which takes on positive values in the range $[0, T_\Phi]$. When subtracted from a timing variable in the *current* local time zone of $\phi_i$, $E_{ij}$ changes the frame of reference to the *next* local time zone of $\phi_j$, taking into account a possible cycle boundary crossing (see Figure 7). Because because the period of each clock phase is identical and $e_i \geq e_{i-1}$, the sum of the shifts between all successive phases is $T_\Phi$:

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

We assume that the setup, hold and propagation delay times of latches are zero. The timing characteristics of a given latch may vary as it is moved across combinational logic nodes and thus

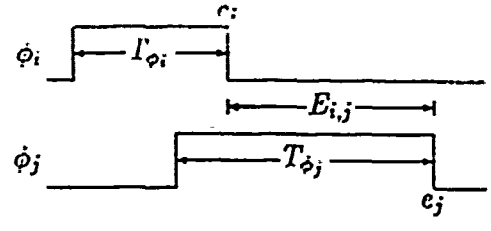


Figure 7: *The phase shift operator provides the relative difference between times in the local time zones of different phases.*

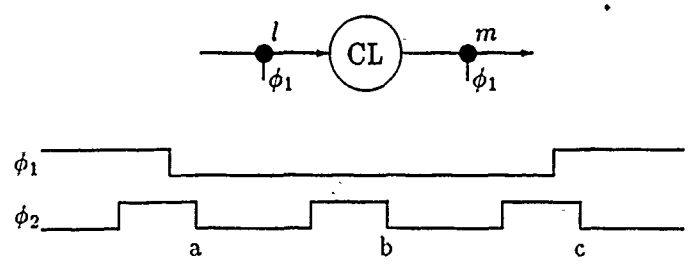


Figure 8: *An illustration of a circuit for which it is not clear what the maximum computational time available for logic block CL is.*

we treat latches as pass gates followed by a non-inverting buffer of zero delay and infinite drive capability. Refining this simplified latch model is a topic for further research.

When enabled, the output value of a latch is defined to be equal to its input value. When disabled, the output value remains that of the input at the time of the most recent latching edge. The final parameters of interest are the values for the arrival and departure times of a latch. The arrival time of a signal at latch $l$ is denoted by $A_l$ and the departure time is denoted by $D_l$ in the local time zone. If $A_l > T_\Phi - T_{P(l)}$ then the latch output is undefined over the interval $(T_\Phi - T_{P(l)}, A_l)$. The departure time is given by:

$$D_l = max\{A_l, T_\Phi - T_{P(l)}\} \tag{2}$$

and the arrival time at a latch $m$ of a signal from latch $l$ connected by a zero-weight path is:

$$A_m = D_l - E_{P(l),P(m)} + d(p) \tag{3}$$

Note that this clock model does not provide for clock phases with differing periods nor for gated clock signals.

# 4 Well-Formed Circuits and Valid Clock Schedules

The goal of the retiming process is one of determining the fastest clock at which latches may be placed in the circuit graph such that the circuit performs "correctly". Thus a definition must exist of when a retimed graph operates correctly. General definitions of correctness for circuits, whether edge-clocked or level-clocked, are difficult to form because the timing constraints which are critical in the initial circuit depend on the designer's intentions of how that circuit is to operate. Given an initial circuit, the clock period for which the circuit operates as intended by the designer may only be determined through the use of a restricted definition of correctness to which the designer adhered. For instance in Figure 8 we see a pair of latches with some amount of combinational logic in between. Without some external knowledge it is not possible to state whether the designer intended that the maximum delay through the logic block is limited such that a signal departing from $l$ arrive at $m$ before latching edge **a**, **b** or **c**.

In this paper a definition of correctness very similar to that for edge-clocked circuits is used. We first restrict the ordering of latch phases as they occur in the circuit graph to allow a simplifed definition of correctness and for retiming constraints to be written which take advantage of knowledge about the graph structure and are least restrictive of the movement of latches during

retiming. The resulting "well-formed" circuits form a large and useful class of circuits including those that are easily produced by automatic synthesis tools. These restrictions can be eased by placing appropriate additional constraints on the retiming, but this is not addressed in this paper.

A *well-formed* circuit is one in which the latches occur in clock phase order along any path through the circuit. More precisely, a circuit graph $G$ is well-formed if:

W1. For every path between latches $l \xrightarrow{p} m$ in $G$, if $w(p) = 0$ then $P(m) = P(l) + 1$.

W2. For every cycle $c$ in $G$, $w(c) \geq 1$.

The first constraint simplifies equations defining the minimum weight along a circuit path by forcing any two $n$-weighted paths which end at the same point in the graph to have the identical ordered latch sequence so that any two paths of equal delay ending at the same vertex require the same number of latches. The second constraint is necessary to avoid races and is the same as that required for edge-clocked graphs. Together these two constraints require every cycle to contain a multiple of $k$ latches for a $k$-phase clock. In the case of level-clock circuits, this constraint must be combined with constraints on the clock schedule to ensure that all cycles contain at least one disabled latch at all times. This will be provided by the valid clock schedules described later in this section.

If we define the clock phase of a vertex $v$, denoted $P(v)$, as the phase of the latch immediately preceding $v$ on any path leading to $v$ then the latch immediately following vertex $v$ on any path has phase $P(v) + 1$.

Retiming a level-clocked graph can now be defined similarly to retiming a edge-clocked graph. The definition of the retiming value $r(v)$ must be extended to include its effect on the latch sequence of adjacent edges. That is, for any edge, $u \xrightarrow{e} v$, the relationship $w_r(e) = w(e) + r(v) - r(u)$ still holds. In addition, $r(v)$ latches (in phase order) are appended to $s(e)$ and $r(u)$ initial latches are deleted from $s(e)$ to form $s_r(e)$. (The case where $r$ is negative is treated symmetrically.)

Well-formed graphs avoid the complexities of identifying when to limit vertex retiming values to prevent movement of latches of differing phases across the vertex. The following lemma assures us that this will not happen in well-formed graphs. However, because the retiming value of the host vertex is restricted to 0, we *can* relax the well-formed definition on paths crossing $v_h$ to allow circuit inputs and outputs to occur on different clock phases as long as cross-host constraints are not used when clocking with unequal phase clocks.

**Lemma 1:** *A well-formed circuit graph remains well-formed under a valid retiming.*

*Proof:* Let $v$ be a vertex in the original graph and $v_r$ the corresponding vertex in the retimed graph. Let $P(v) = \phi_i$ and thus the phase of the latch following $v$ is $\phi_{i+1}$. A retiming value of $r(v) = 1$ removes the first latch from the latch sequence of each output edge and appends a latch of phase $i + 1$ to the latch sequence of each input edge. The case for $r(v) = -1$ is symmetric and induction provides a proof for any value of $r(v)$. Thus latch ordering (W1) is maintained. That W2 is maintained for cycles follows from the retiming results for edge-clocked circuits [6]. □

## 4.1 Correct Operation of Level-Clocked Circuits

There are two conditions which must be met to ensure the correct operation of a level-clocked circuit. The first states that along any path in the circuit, the signal departing a latch must arrive

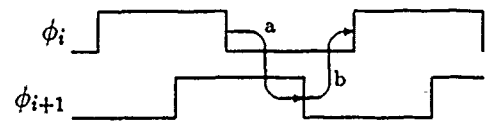


Figure 9: *Graphical representation of the constraints on the clock phases that are required for correct operation of a well-formed level-clocked circuit.*

at the next latch before the next latching edge for that latch. The second states that along any path in the circuit, the signal departing a latch must not arrive at the next latch before the *previous* latching edge for that latch. More precisely,

L1. Maximum delay: For any zero-weight path $l \xrightarrow{p} m, A_m = D_l - E_{P(l),P(m)} + d(p) \leq T_\Phi$.

L2. Non-interference: For any zero-weight path $l \xrightarrow{p} m, A_m = D_l - E_{P(l),P(m)} + d(p) > 0$.

We now want to remove any assumption about minimum delay in a correctly operating level-clocked circuit. This allows us to avoid two-sided delay constraints and allows retiming to relocate latches without concern for retaining vertices between latches.

We now define a valid clock schedule and show that any well-formed circuit operated by a valid clock schedule satisfies the non-interference constraint L2. That is, if a retiming satisfies the maximum delay constraint, then it results in a correctly operating circuit even with zero delays between latches.

A clock schedule is *valid* if it meets the following constraints:

P1. $e_{i+1} \geq e_i$, which follows from the definition of a clock schedule (constraint **a** in Figure 9).

P2. $\left\{ \begin{array}{rl} e_i + T_\Phi - T_{\phi_i} > e_{i+1} & \text{for } i \neq k \\ e_i - T_{\phi_i} > e_0 & \text{for } i = k \end{array} \right\}$ (constraint **b** in the figure).

Note that these constraints allow for multiple phase clock schedules with overlapping and underlapping phases. However, two-phase clocks are required to be non-overlapped.

It follows from constraint P2 that there is no time $t$ where $e_i - T_{P(i)} < t < e_i$ for $i = 1, ..., k$. That is, not all latches can be active simultaneously and thus we avoid race conditions in cycles.

**Theorem 2:** *Any well-formed level-clocked circuit operating with a valid clock schedule meets the non-interference constraint L2.*

*Proof:* By Eqn. 2 on page 10. $D_l \geq T_\Phi - T_{\phi_i}$, that is, the departure time from a $\phi_i$ latch must occur at or after the enabling edge. By constraint P2, $E_{i,i+1} = e_{i+1} - e_i < T_\Phi - T_{\phi_i}$ and so $E_{i,i+1} < D_l$. Since $P(m) = P(l) + 1$ for any path $l \xrightarrow{p} m$ with $w(p) = 0$ in a well-formed graph, $E_{P(l),P(m)} < D_l$ and thus $D_l - E_{P(l),P(m)} > 0$. Thus constraint L2 holds for any $d(p) \geq 0$. □

**Corollary 3:** *The phase $P_r(u)$ of a node $u$ in a well-formed, retimed graph $G_r$ using a k-phase clock and given $P(u)$ in the initial graph $G$ with $r(u)$ the retiming value of $u$, is:*

$$P_r(u) = [P(u) + r(u)] \bmod k.$$

*Where* $\phi_0 \equiv \phi_k$.

*Proof:* $r(u) = n$ is defined as the movement of $n$ latches across node $u$. By the definition of a well-formed graph, $P(l_1) = P(l_0) + 1$ for any $l_0$, $l_1$ connected by a zero-weight path. Thus, $P(l_n) = [P(l_0) + n] \bmod k$ since there are $k$-phases in the clock schedule and $\phi_{k+1} = \phi_1$ as defined. Under retiming:

$$\begin{aligned} P_r(u) &= P(l_{r(u)}) \\ &= [P(l_0) + r(u)] \bmod k \\ &= [P(u) + r(u)] \bmod k. \square. \end{aligned}$$

# 5 Level-Clocked Timing Constraints

This section derives the fundamental Theorem 4 which will provide the basis for ILP path constraint sets that ensure a valid retiming of a graph $G$ for a given multi-phase clock schedule $\Phi$. The theorem provides an upper bound on the delay of an $n$-weight simple path in a level-clocked graph in terms of the departure time of a signal at the beginning of the path and the arrival time at the end. The proof is based on the maximum delay constraint L1 of the previous section extended to paths of non-zero weight. Figure 10 gives a graphical representation of this theorem.

Theorem 4 provides an exact bound on the maximum possible delay of a path based on the departure time of signals from the latch preceding the path and the subsequent arrival time of signals at the latch terminating the path. For retiming purposes we are interested in a maximum bound on path delay which is presented in Corollary 5. We then show in Corollary 6 that cycles additionally constrain the clock period and show how an analysis of critical cycles can be used to derive a lower bound on the clock period.

Finally we demonstrate that the edge-clocked definition of a critical path between two nodes is insufficient to ensure correct retiming of the nodes at all clock periods. A new definition for critical paths is derived and a method of identifying a critical path between two nodes is presented.

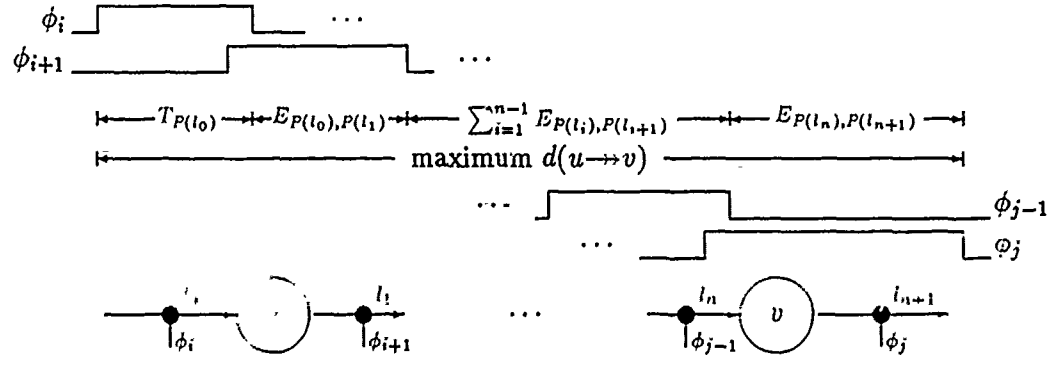


Figure 10: *Graphical representation of the constraint on the simple path delay between two latches* $l_0$ *and* $l_{n+1}$ *in a correctly operating circuit.*

**Theorem 4:** *A multi-phase, level-clocked circuit graph* $G$ *is correctly timed using a valid clock schedule if and only if for every simple path* $l_0 \xrightarrow{p} l_{n+1}$ *with weight* $w(p) = n$ *and latch sequence*

$s(p) = \{l_0, l_1, \ldots l_{n+1}\}$, *the path delay* $d(p)$ *is bounded by:*

$$d(p) \leq A_{l_{n+1}} - D_{l_0} + \sum_{i=0}^{w(p)} E_{P(l_i),P(l_{i+1})}.$$

*Proof:* ($\Rightarrow$) By induction on the weight of a path $w(p)$.

*Basis:* When $w(p) = 0$, $d(p) = A_{l_1} - D_{l_0} + E_{P(l_0),P(l_1)}$ by Eqn. 3 on page 10.

*Induction:* Divide $p$ into two paths $l_0 \xrightarrow{p_1} l_1$ and $l_1 \xrightarrow{p_2} l_{n+1}$. From Eqn. 3, $d(p_1) = A_{l_1} - D_{l_0} + E_{P(l_0),P(l_1)}$. By the inductive hypothesis, $d(p_2) \leq A_{l_{n+1}} - D_{l_1} + \sum_{i=1}^{w(p)} E_{P(l_i),P(l_{i+1})}$. By Eqn. 2 on page 10 $A_{l_1} \leq D_{l_1}$ and for a correctly operating circuit, $A_{l_1} \leq T_\Phi$. Thus $A_{l_1} \leq D_{l_1} \leq T_\Phi$ and so $d(p) = d(p_1) + d(p_2) \leq A_{l_{n+1}} - D_{l_0} + \sum_{i=0}^{w(p)} E_{P(l_i),P(l_{i+1})}$.

($\Leftarrow$) We show that if $d(p) > A_{l_{n+1}} - D_{l_0} + \sum_{i=0}^{w(p)} E_{P(l_i),P(l_{i+1})}$ then the constraint on valid timing defined by Eqn. 3 must be violated at some latch.

*Case 1:* If $w(p) = 0$: Eqn. 3 is violated directly.

*Case 2:* If $w(p) > 0$: We assume that no zero-weight subpath $q$ of $p$ exists such that Eqn. 3 is volated and show by contradiction that this cannot be true. Since $d(p) = \sum_{i=0}^{n} d(q_i)$ where $l_i \xrightarrow{q} l_{i+1}$, and from our assumtion $d(q_i) \leq A_{l_{i+1}} - D_{l_i} + E_{P(l_i),P(l_{i+1})}$, therefore:

$$\sum_{i=0}^{n} d(q_i) \leq \sum_{i=0}^{n} [A_{l_{i+1}} - D_{l_i} + E_{P(l_i),P(l_{i+1})}].$$

Substituting for $d(p)$ and $\sum d(q)$:

$$A_{l_{n+1}} - D_{l_0} + \sum_{i=0}^{w(p)} E_{P(l_i),P(l_{i+1})} > A_{l_{n+1}} - D_{l_0} + \sum_{i=0}^{w(p)} E_{P(l_i),P(l_{i+1})},$$

forming a contradiction. □

The following two corollaries use the minimum departure and maximum arrival times of signals from latches to state the maximum simple path delay and maximum cycle delay in terms of the clock schedule and path weight.

**Corollary 5:** *A multi-phase, level-clocked graph $G$ using a valid clock schedule is correctly timed if and only if the delay of any simple path $l_0 \xrightarrow{p} l_{n+1}$ is bounded by:*

$$d(p) \leq T_{P(l_0)} + \sum_{i=0}^{w(p)} E_{P(l_i),P(l_{i+1})}.$$

*Proof:* The result follows directly from Theorem 4 by observing from Eqn. 2 that the minimum departure time $D_0 = T_\Phi - T_{P(l_0)}$ and from constraint L1 that the maximum arrival time $A_{l_{n+1}} = T_\Phi$. □

**Corollary 6:** *A multi-phase, level-clocked graph $G$ using a valid clock schedule is correctly timed if and only if the delay of any cycle $l_0 \xrightarrow{c} l_{n+1}$ is bounded by:*

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

*Proof:* The result follows directly from Theorem 4 by observing that $l_0 = l_{n+1}$ and thus $A_{l_{n+1}} \leq D_{l_0}$. Additionally, since the weight of a cycle includes all latches placed on the cycle, the weight of a path $l_0 \xrightarrow{p} l_{n+1}$ is $w(c) - 1$. □

Given the result of Corollary 6 we can form a tight lower bound based on cycle delays on clock periods for which the circuit will operate correctly. Unlike in edge-clocked circuits, this lower bound may more restrictive in some cases than path based constraints for the same circuit. Hence the critical cycle period must be found independently of path constraints. The following corollary derives the lower bound on the clock period of a circuit based on cycles in the graph.

**Theorem 7:** *For any correctly operating, well-formed graph $G$ using a $k$-phase clock schedule:*

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

*Proof:* By Corollary 6:

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

In a well formed graph each cycle must contain $\frac{w(c)}{k}$ latches of each phase. By Eqn. 1 on page 9, $\sum_{i=1}^{k} E_{i,i+1} = T_\Phi$. Hence:

$$d(c) \leq \sum_{i=1}^{\frac{w(c)}{k}} T_\Phi = w(c) \cdot \frac{T_\Phi}{k},$$

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

In our search for an optimal retiming, we are restricted to clock values greater than $k(\frac{d(c)}{w(c)})$. A critical cycle, denoted $C_R$, is a cycle which maximally restricts the clock period, that is, a cycle for which $\frac{d(c)}{w(c)}$ is maximum. The value of $\frac{d(C_R)}{w(C_R)}$ for a critical cycle in the graph may be found by setting the values $\alpha(u \xrightarrow{e} v) = d(v)$ and $\varepsilon(e) = w(e)$, and solving the *maximum-ratio-cycle* problem for $\frac{\alpha(c)}{\varepsilon(c)}$. Polynomial-time algorithms are available to solve this problem from Megiddo [9] and Burns [2]. In particular the algorithm by Burns has a provable running time of $O(|E| \cdot |V| \cdot k)$ where $k$ is the maximum weight of a cycle in $G$. The resulting $\frac{d(c)}{w(c)}$ provides a tight lower bound on the cycle time of the circuit. Although this clock cycle may not be realizable due to restrictions of the more general path constraints, it provides a useful starting point in searching for the optimum cycle time of the circuit.

## 5.1 Critical Paths

Now that a lower bound on possible clock periods has been established based on cycles in the graph, a search process must be performed to determine the minimum clock period above that

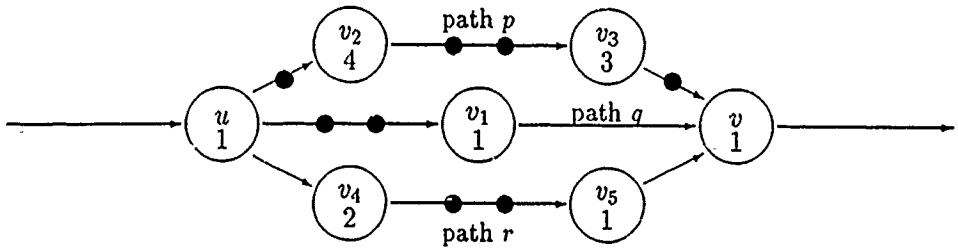


Figure 11: *Critical paths in level-clocked circuits may not be the same as critical paths in edge-clocked circuits.*

bound for which a retiming can be found that satisfies path constraints. To avoid having to determine constraints for all paths in the circuit which is possibly exponential in number of edges, a critical path between any two nodes $u$ and $v$ is found such that the minimum weight constraints for all paths between the nodes can be met by just satisfying the constraints of the critical paths.

We redefine a *critical path* for a circuit to be the path $u \xrightarrow{p} v$ such that if the minimum weight constraint is met for $p$, then it is met for all paths from $u$ to $v$ for any valid retiming. Critical paths are more difficult to determine in level-clocked circuits. The reason is that the path limiting the clock period may not be a zero-weight path as guaranteed for edge-clocked circuits. This is demonstrated in Figure 11. Three paths exist between nodes $u$ and $v$, labeled from top to bottom:

| path | vertices | w(p) | d(p) |
|---|---|---|---|
| p | $u \rightarrow v_2 \rightarrow v_3 \rightarrow v$ | 4 | 9 |
| q | $u \rightarrow v_1 \rightarrow v$ | 2 | 3 |
| r | $u \rightarrow v_4 \rightarrow v_5 \rightarrow v$ | 2 | 5 |

In an edge-clocked circuit, path $r$ is clearly critical since $w(r) = \min\{w(p), w(q), w(r)\}$ and $d(r) = \max\{d(q), d(r)\}$; However, if we consider this a level-clocked circuit, with 2-equal-phases and period $T_\Phi = 2$, and using the techniques presented in Section 6, the minimum required weight of path $p$ is 7 while that of path $r$ is 4. If path $r$ were selected as the critical path between $u$ and $v$, a retiming which results in $w_r(r) = 4$ would be considered successful even though (since retiming maintains a constant difference in path weight between $p$ and $r$) the resulting $w_r(p) = 0$. Under the new definition, the critical path between $u$ and $v$ is path $p$ for clock period $T_\Phi = 2$. Note that the critical path under the new definition will vary with differing clock periods. For instance, the critical path in Figure 11 at $T_\Phi = 10$ is $r$ instead of $p$.

We must now identify the most constraining path from $u$ to $v$ for a given clock period. The following lemma provides the basis for efficiently determining a critical path.

**Lemma 8:** *A path $u \xrightarrow{p} v$ in a well-formed circuit is a critical path if*

$$\{w(p)\frac{T_\Phi}{k} - d(p)\} \leq \{w(q)\frac{T_\Phi}{k} - d(q)\} \text{ for all } u \xrightarrow{q} v.$$

*Proof:* By Contradiction. The delay constraint of Corollary 5 may be restated as:

$$T_{P(l_0)} + \sum_{i=0}^{w(p)} E_{P(l_i),P(l_{i+1})} - d(p) \geq 0,$$

Assume that $u \xrightarrow{p} v$ is a path where $\{w(p)\frac{T_\Phi}{k} - d(p)\} \leq \{w(q)\frac{T_\Phi}{k} - d(q)\}$ for all paths $u \twoheadrightarrow v$ but $p$ is not a critical path. This implies that there exists some $u \xrightarrow{q} v$ and a retiming such that: $T_{P(l_0)} + \sum_{i=0}^{w_r(p)} E_{P(l_i),P(l_{i+1})} - d(p) \geq 0$, and $T_{P(l_0)} + \sum_{i=0}^{w_r(q)} E_{P(l_i),P(l_{i+1})} - d(q) < 0$. That is, the minimum weight constraint is satisfied for $p$ but not for $q$ in the retimed circuit. However, since retiming maintains a constant difference in path weight for $p$ and $q$, and the graph is well formed, $w_r(p) - w_r(q) = km = w(p) - w(q)$ where $k$ = number of clock phases and $m$ is some integer. Combining the weight constraint inequalities gives:

$$\begin{aligned}
T_{P(l_0)} + \sum_{i=0}^{w_r(p)} E_{P(l_i),P(l_{i+1})} - d(p) &> T_{P(l_0)} + \sum_{i=0}^{w_r(q)} E_{P(l_i),P(l_{i+1})} - d(q), \\
\sum_{i=0}^{w_r(p)} E_{P(l_i),P(l_{i+1})} - d(p) &> \sum_{i=0}^{w_r(p)+km} E_{P(l_i),P(l_{i+1})} - d(q), \\
-d(p) &> \sum_{i=w_r(p)+1}^{w_r(p)+km} E_{P(l_i),P(l_{i+1})} - d(q)
\end{aligned}$$

Using the result from Eqn. 1 on page 9, $\sum_{i=1}^{km} E_{l_i,l_{i+1}} = mT_\Phi$. Hence:

$$\begin{aligned}
-d(p) &> mT_\Phi - d(q), \\
-d(p) &> km\frac{T_\Phi}{k} - d(q), \\
-d(p) &> (w(q) - w(p))\frac{T_\Phi}{k} - d(q), \\
w(p)\frac{T_\Phi}{k} - d(p) &> w(q)\frac{T_\Phi}{k} - d(q).
\end{aligned}$$

Which contradicts our initial assumption. □

We now determine the values in the matrices $D(u,v,T_\Phi)$ and $W(u,v,T_\Phi)$ as $d(p_c)$ and $w(p_c)$ for a critical path $u \xrightarrow{p_c} v$. This in turn requires identifying the path which minimizes the quantity $\{w(p)\frac{T_\Phi}{k} - d(p)\}$ over all paths $u \xrightarrow{p} v$. We can find the paths which minimize this value by running an all-pairs shortest path algorithm on $G$ using new edge weights $\tilde{w}(e) = (w(e)\frac{T_\Phi}{k} - d(v_2))$ for each $v_1 \xrightarrow{e} v_2$.[3]

The Floyd-Warshall algorithm may be used to solve the all-pairs shortest path problem since it will handle the possibly negative weight values of $\tilde{w}(e)$ as long as there are no negative weight cycles in the graph. This requires showing that there is no cycle $c$ for which $\tilde{w}(c) = w(c)\frac{T_\Phi}{k} - d(c) < 0$. As a result of Theorem 7, we can place a lower bound on the clock period used to retime a circuit. That is, we will use only clock periods such that $T_\Phi \geq k\left(\frac{d(c)}{w(c)}\right)$ for all cycles $c$ in $G$. Thus there can be no negative weight cycles for clock periods of interest and all critical paths can be determined efficiently.

Note that unlike edge-clocked circuits, in general $D$ and $W$ must be recomputed for every clock period attempted by the retiming; However, returning to Figure 11 we can plot the result of $\{w(p)\frac{T_\Phi}{k} - d(p)\}$ from Lemma 8 for each path against the clock period $T_\Phi$. The resulting plot

[3] The sum of all $d(v_2)$ in $\tilde{w}$ equals $d(u \xrightarrow{p} v) - d(u)$ rather than $d(p)$. However, because node $u$ is the first node in all paths $u \xrightarrow{p} v$, minimization of $\tilde{w}$ will minimize $\{w(p)\frac{T_\Phi}{k} - d(p)\}$ as well.

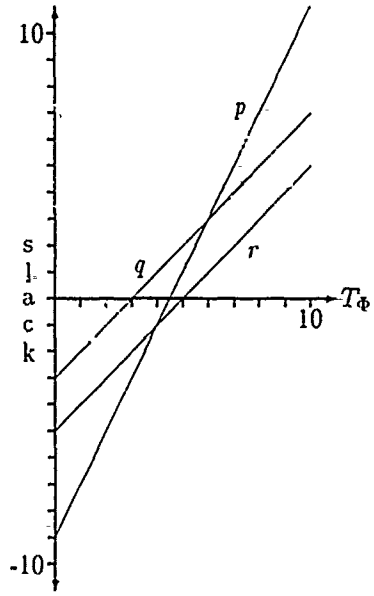


Figure 12: *Plot of slack vs. clock period for paths p, q and r.*

is shown in Figure 12. The slack value for each path $p$ is a linear function in clock period with slope= $\frac{w(p)}{k}$ and y-axis intercept at $-d(p)$. At all clock periods greater than the intersection point between the slack functions for two paths, one of the paths will have less slack than the other and the other path will have less slack at all clock periods less than the intersection point. For any two paths of the same weight, the path with greater delay will always have less slack than the other. Due to these properties, if a particular path is critical for any two clock periods, it will also be critical for all clock periods in between.

Once two clock periods are found, one above the optimal clock period $T_{\Phi_{opt}}+$ and one below the optimal clock period $T_{\Phi_{opt}}-$, for which $W(u,v,T_{\Phi_{opt}}-) = W(u,v,T_{\Phi_{opt}}+)$ for all values $u$ and $v$, then for any clock periods $T_{\Phi_{opt}}- \leq T_\Phi \leq T_{\Phi_{opt}}+$ the arrays $W$ and $D$ will remain constant and need not be recomputed. Additionally, because the slope of each slack function is $\geq 0$, if $W(u,v,T_{\Phi_{opt}}-) = \min\{w(p) \mid u \xrightarrow{p} v\}$ then $W(u,v,T_\Phi) = W(u,v,T_{\Phi_{opt}}-)$ for all $T_\Phi \geq T_{\Phi_{opt}}-$.

# 6 Retiming for Equal Phase Clocks

The theorems in the previous section form the basis for a set of constraints which can be used to determine whether a retiming exists for a particular clock schedule. In this section we investigate a simple clock schedule with equal length phases. In Section 7 we extend the capability to more complex clocks with unequal length phases. The resulting constraint sets can be solved in a manner similar to the original Leiserson et. al. methods to perform retiming of level-clocked circuits.

An *equal-phase clock schedule* is a valid k-phase clock set $\Phi = \{\phi_1, ..., \phi_k\}$ where all active phase periods $T_{\phi_i}$ are equal, and all phase shifts $E_{i,i+1} = \frac{T_\Phi}{k}$. Since the length of the active period is the same for all phases, we use $T_\phi$ to refer to the length of the active period of any phase. Note that this definition allows overlapped clock phases under the general constraints on valid clocks. Because the active periods and phase shift values of the phases of each adjacent pair of latches are equal, the

retiming process can ignore the actual phase of the individual latches. In the identical manner to edge-clocked circuits, a retiming value $r(v)$ is assigned to each vertex of the circuit graph. However, a value of $r(v) = n$ now moves $n$ latches across the vertex rather than $n$ registers.

Proofs in [6, 7] for edge-clocked circuit graphs showing that all cycles maintain the same number of latches and that phase differences between paths with common endpoints remain constant also hold for level-clocked graphs. Additionally, because $r(v_h) \equiv 0$, no new latches will be introduced from or transferred to the outside world. It is not possible to limit path constraints for level-clocked circuits to the length of zero-weight paths as in edge-clocked circuits. The delay of any latch-bounded path is affected by the paths preceding and following it, requiring constraints for higher weight paths as well.

Corollaries 5 and 6 provide a basis for retiming level-clocked circuits operating under an k-equal-phase clock schedule. These constraints take two forms: a minimum possible clock period based on simple cycles and sets of timing constraints for given clock periods on paths.

The method used to form the required constraint set is to first guarantee that the minimum cycle period constraint imposed by Corollary 6 will be met. Following identification of the minimum possible clock period based on cycles we combine the result of Corollary 5 with knowledge of an equal-phase clock to derive $L(u, v)$, the minimum weight for a critical path between $u$ and $v$. Using this result a pass is made through the $W$ and $D$ arrays corresponding to the critical paths in $G$ to form a set of path constraints requiring $L(u, v)$ latches rather than one as in the previous work.

We now restate the maximum delay constraint as a minimum weight constraint which provides a lower bound on the number of latches on a simple path in terms of the path delay.

Corollary 9: *The weight of any simple path $p$ in a correctly operating, well-formed circuit graph $G$ using a k-equal-phase clock schedule is bounded by:*

$$w(p) \geq \left\lceil \frac{d(p) - T_\phi}{\frac{T_\Phi}{k}} \right\rceil - 1$$

*Proof:* The result follows directly from Corollary 5 on page 14 using the fact that for k-equal-phase clock schedule, $E_{i,i+1} = \frac{T_\Phi}{k}$ and $\forall_{i,j}$, $T_{\phi_i} = T_{\phi_j} = T_\phi = T_{P(l_0)}$. □

We define $L(u, v) = \left\lceil \frac{D(u,v) - T_\phi}{\frac{T_\Phi}{k}} \right\rceil - 1$ as the minimum number of latches required on a critical path from $u$ to $v$. This value forms the basis for a set of constraints for retiming well-formed circuit graphs controlled by a $k$-equal-phase clock schedules for a given clock period $T_\Phi$:

| | |
|---|---|
| I/O: | $r(v_h) = 0$ |
| Positive edge weights: | $r(u) - r(v) \leq w(e)$ for all edges $u \xrightarrow{e} v$ |
| Maximum path delay: | $r(u) - r(v) \leq W(u, v) - L(u, v)$ for all $(u, v)$ such that $L(u, v) \geq 1$ |

## 6.1 Correlator Example Revisited

The correlator shown in Figure 3 on page 4 can be transformed into a well-formed, two-phase, level-clocked circuit by replacing each register in the edge-clocked circuit with a pair of $\phi_1, \phi_2$ latches, thereby doubling the weight of each edge. Retiming this example using a two-equal-phase, non-overlapped clock schedule leads to the circuit graph of Figure 5 on page 4. The $W$ and $D$

matrices are the same as in the original example except that all values in $W$ are multiplied by two to reflect the conversion to latches.

Finding $T_{C_R}$ results in a value of 10 units. Several cycles are critical in this particular graph:

| Vertices in cycle | d(cycle) | w(cycle) | d/w |
|---|---|---|---|
| $v_h, v_1, v_7, v_h$ | 10 | 2 | 5 |
| $v_h, v_1, v_2, v_6, v_7, v_h$ | 20 | 4 | 5 |
| $v_h, v_1, v_2, v_3, v_5, v_6, v_7, v_h$ | 30 | 6 | 5 |

Retiming to an ideal two-equal-phase clock schedule with $T_\Phi = 10$ ($d/w = 5$ for any critical cycle, which requires $T_\Phi = 10$ for a 2 phase clock) requires that the following path constraints be met:

| Path $u \rightarrow v$ | # latches req'd | $D(p)$ |
|---|---|---|
| $1 \rightarrow 4, 1 \rightarrow 6, 2 \rightarrow 5$ $5 \rightarrow 6, 6 \rightarrow 7, 7 \rightarrow 2$ | 1 | $> (1 \cdot T_\Phi)$ |
| $1 \rightarrow 5, 2 \rightarrow 7, 3 \rightarrow 6$ $4 \rightarrow 6, 6 \rightarrow 1, 7 \rightarrow 3$ $7 \rightarrow 6$ | 2 | $> (1.5 \cdot T_\Phi)$ |

| Path $u \rightarrow v$ | # latches req'd | $D(p)$ |
|---|---|---|
| $5 \rightarrow 7, 6 \rightarrow 3$ $7 \rightarrow 5$ | 3 | $> (2 \cdot T_\Phi)$ |
| $3 \rightarrow 1, 4 \rightarrow 1, 5 \rightarrow 2$ $6 \rightarrow 4, 6 \rightarrow 5$ | 4 | $> (2.5 \cdot T_\Phi)$ |
| $4 \rightarrow 3, 5 \rightarrow 4$ | 5 | $> (3 \cdot T_\Phi)$ |

The above set of constraints when combined with the necessary edge constraints may be solved successfully to define a set of retiming values resulting in the latch placement in Figure 5.

## 6.2 Determining Potential Optimum Clock Periods

It is not always possible to retime level-clocked circuits to the lower bound clock period given by a critical cycle as in the previous example. For a theoretically precise optimum value, it becomes necessary to determine a set of possible optimum clock periods over which to search for a minimum legal clock period. In an optimal edge-clocked circuit there exists some critical path of zero weight with delay exactly the value of the clock period and thus it is a simple matter to make a list of all path delays from the $D$ array and perform a binary search on that list to determine the optimal $T_\Phi$. In level-clocked circuits the critical path may be many clock phases in length and of weight greater than zero. Additionally the critical path between two vertices may change given two differing clock periods. Enumeration of all possible paths between two vertices can be exponential in the number of edges and so is not practical as a starting point for determining possible critical path weights.

The following theorem uses the fact that for an optimal retiming of a level-clocked circuit graph (not retimed to the critical cycle period) there will exist some critical path which exactly meets the minimum weight requirements. If this were not true there would exist a faster clock period for the same weighting. As we approach the optimal clock period for the graph we identify a range in which the optimal clock period exists and over which the critical paths in the circuit graph will not change; however, in certain cases, when the intersection of slack values for two paths occurs at precisely the optimal clock period, it is not possible to determine a range over which the critical path arrays will match. In this case we our algorithm must exit at some desired level of accuracy. Assuming a range over the optimal period is found with matching $W$ arrays, we change the inequality from the minimum path weight result in Corollary 9 to an equality and form a set of possible optimal clock periods.

**Theorem 10:** *The optimal cycle time $T_{\Phi_{opt}}$ of a well-formed circuit graph $G$ clocked by a $k$-equal-phase clock is in the set $C$:*

$$C = \left\{ \left( \frac{D(u,v)}{\frac{i+1}{k} + T_d} \right) \;\middle|\; u,v \in V; i \in \{0,1 \ldots n\} \right\}$$

*where:* $T_d$ = *duty cycle of each phase* = $\frac{T_\phi}{T_\Phi}$ *and* $n$ *is the maximum integer value for which the resulting clock period computed is greater than the critical cycle period.*

*Proof:* Follows from setting the left and right hand sides of Corollary 9 equal and solving for $T_\Phi$. Because the value $T_\phi$ is proportional to $T_\Phi$, substitute in duty cycle ($T_d \cdot T_\Phi$) instead which remains constant for the clock schedule. □

Real circuit graphs have built-in error due to estimation of combinational logic delays, and thus the value of generating a precise set of possible optimum clock periods is questionable. Large, complex circuits with many combinations of possible path delays and weights have a densely populated set of possible optimum clock periods. A more reasonable approach is simply to perform a search over real values to the desired level of accuracy given some knowledge of the accuracy of the circuit modeling process. However, the technique of finding an exact optimum is presented here to show it is significantly different from the original work.

We can now define a new algorithm for finding the optimal retiming of a $k$-equal-phase, level-clocked circuit graph:

**Algorithm: Optimal $k$-Equal-Phase Retiming:**

1. Determine the critical cycle period $T_{C_R} = k\left(\frac{d(C_R)}{w(C_R)}\right)$.

1. Attempt to retime to $T_{C_R}$; if successful $T_{\Phi_{opt}} = T_{C_R}$.

2. Repeatedly multiply the value of $T_{C_R}$ by $\alpha$ until a legal retiming is found. Set $T_{\Phi_{opt}}+$ to the clock period of the first legal retiming. Set $TPopt-$ to $\frac{T_{\Phi_{opt}}+}{\alpha}$.

3. If for all $u$ and $v$, $W(u,v,T_{\Phi_{opt}}-) = W(u,v,T_{\Phi_{opt}}-)$, proceed to step 4, otherwise perform a binary search over $(T_{\Phi_{opt}}-, T_{\Phi_{opt}}+]$ until matching $W$ arrays are found or a desired level of accuracy is met.

4. Perform a binary search over the set of possible optimum cycle times $C$ computed from $W$ and $D$ to find the minimum value for which a legal retiming can be found.

As pointed out above, finding an exact optimal solution in practice is probably not worthwhile. Instead eliminate Step 4 above and replace Step 3 with:

3. Perform a binary search over $(T_{\Phi_{opt}}-, T_{\Phi_{opt}}+]$ until the desired level of accuracy is reached.

Since in practice $T_{\Phi_{opt}}$ will likely be near the value of $T_{C_R}$, biasing the search pattern such that the lower end is favored is often worthwhile. Our system uses $\alpha = 1.25$. Additionally, because high values of $n$ in Theorem 10 result in the smallest values in set $C$, the set may be generated incrementally as needed rather than all at once.

**Example 2: A 2-equal-phase example with phase underlap**

Real circuits cannot be designed with an ideal clock schedule as was used in the previous example. Instead a typical clock schedule might have each active period $T_\phi = 0.4T_\Phi$ giving an

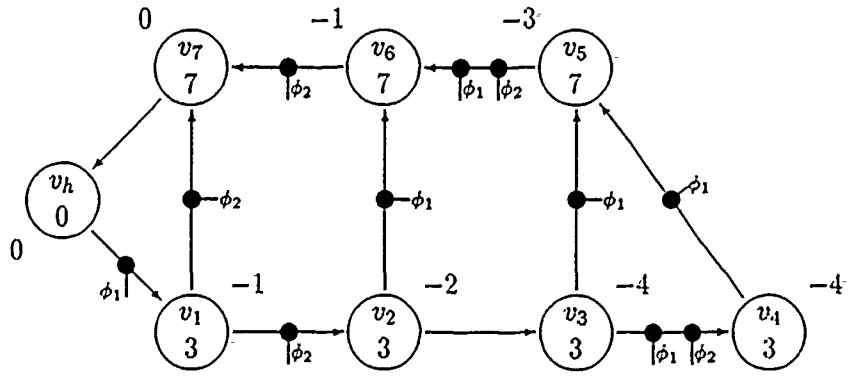


Figure 13: *The correlator circuit optimally retimed using a 2-phase, equal-period clock where* $T_\phi = 0.4T_\Phi$; $T_{\Phi_{opt}} = 10.345$ *units.*

underlap between phases of $0.1T_\Phi$. In this example we retime the correlator circuit graph using such a clock schedule.

As in the previous example $T_{C_R} = 10$; however, in this case a retiming to that clock period cannot be found. A legal retiming is found to $T_\Phi = 20$ and the $W$ and $D$ matrices match at 20 and 10. The set of possible time periods $C$ is:

$$C = \{10.0,\ 10.345.\ 10.526,\ 10.833,\ 11.053,\ 11.111,\ 11.25...\}$$

A binary search over this list finds the fastest time possible $T_{\Phi_{opt}} = 10.345$. The circuit retimed to this clock schedule is shown in Figure 13.

## 6.3 Reducing the Required Number of Constraints

We do not consider the larger number of constraints required for the level-clocked retiming to be much of a problem since the overall number of constraints is still limited to $|V|^2$. However, it is true that the expected number of constraints is much greater than for edge-clocked retiming since long paths usually have a different constraint imposed on them than on their sub-paths whereas in edge-clocked graphs constraints on long paths were usually redundant with a shorter sub-path. The exact relationship between the number of constraints for the two retiming methods is dependent on the graph structure and on the delay of vertices in the graph relative to the length of the clock period of interest. The correlator example is again useful here because it may easily be extended lengthwise and the number of constraints for different-sized graphs compared. Table 1 displays $T_{\Phi_{opt}}$ in relationship to number of nodes in the correlator graph and the number of path constraints required for each technique to retime to the corresponding optimal clock period.

It is possible to limit the number of constraints required for retiming by limiting the number of latches through which borrowing is allowed. If borrowing is allowed only through $N$ latches, path constraints are defined as:

$$\begin{aligned} \text{path constraints:} \quad & r(u) - r(v) \leq W(u,v) - L(u,v) \text{ for any } 0 < L(u,v) < N \\ \text{limited path constraints:} \quad & r(u) - r(v) \leq W(u,v) - L(u,v) - 1 \text{ for any } N \leq L(u,v) \end{aligned}$$

Since long paths are now over-constrained and a greater portion of the path constraints will be redundant to shorter sub-paths, limiting borrowing in this manner reduces the number of constraints

| Correlator size in nodes | $T_{\Phi_{opt}}$ Edge-Trig | # path constraints | $T_{\Phi_{opt}}$ Ideal 2-equal-phase | # path constraints |
|---|---|---|---|---|
| 8 | 13 | 5 | 10 | 23 |
| 10 | 13 | 8 | 10 | 41 |
| 12 | 14 | 16 | 10.286 | 65 |
| 14 | 14 | 20 | 10.286 | 95 |
| 16 | 14 | 24 | 10.5 | 129 |
| 20 | 14 | 32 | 10.5 | 219 |
| 30 | 14 | 52 | 10.5 | 528 |
| 50 | 14 | 92 | 10.5 | 1546 |
| 100 | 14 | 192 | 10.5 | 6354 |

Table 1: *Comparison of optimal clock periods found for varying sizes of correlator circuits.*

| N | # path constraints | $T_{\Phi_{opt}}$ for Ideal 2-equal-phase | N | # path constraints | $T_{\Phi_{opt}}$ for Ideal 2-equal-phase |
|---|---|---|---|---|---|
| 1 | 142 | 14.00 | 8 | 1225 | 11.17 |
| 2 | 285 | 12.74 | 9 | 1309 | 11.02 |
| 3 | 474 | 11.56 | 10 | 1437 | 10.78 |
| 4 | 658 | 11.72 | 12 | 1757 | 10.78 |
| 5 | 838 | 11.72 | 15 | 2098 | 10.78 |
| 6 | 880 | 11.02 | 16 | 2169 | 10.70 |
| 7 | 1056 | 10.94 | 17 | 2240 | 10.50 |

Table 2: *Optimal clock periods found while using restricted constraint sets that allow borrowing over N latches in the 100 node correlator example.*

required to retime the graph at the expense of finding a less than optimal solution. The experimental results shown in Table 2 demonstrate that the 100 node correlator example can be retimed to the optimal clock period with many fewer constraints than those used for the most general case.

The time values in Table 2 were derived using smaller, limited constraint sets. In most cases the resulting graph can actually be operated at a higher speed than that shown, however the over-constrained retiming process cannot show that to be true without a less restrictive constraint set. The difficulty with this heuristic technique is also demonstrated: The optimal time period found does not decrease monotonically with increasing number of levels. This is due to an interaction of the graph granularity with the level at which paths are over-constrained.

# 7 Retiming of Unequal Phase Circuits.

Unlike equal-phase retiming, the minimum weight of a path under an unequal phase clock schedule depends on the phase controlling the latch at which the path begins. This difference is demonstrated in Figure 14 which shows that the maximum length of 0 and 1 weight paths beginning at a vertex is controlled by each phase of a 2-phase clock. Neither the edge-clocked retiming methods from [6, 7] nor the equal-phase retiming developed in Section 6 can differentiate which phase of latch is being moved across a particular vertex in the graph; However, because latch phases alternate along paths in well-formed graphs, the knowledge of what phase latch precedes a given vertex is directly available from the retiming values.

First, the minimum weight value for a critical path is extended to a set of values $L_i(u,v)$, $i \in \{0 \ldots k-1\}$, indexed relative to the initial arrangement of latches in the circuit graph. Similarly,

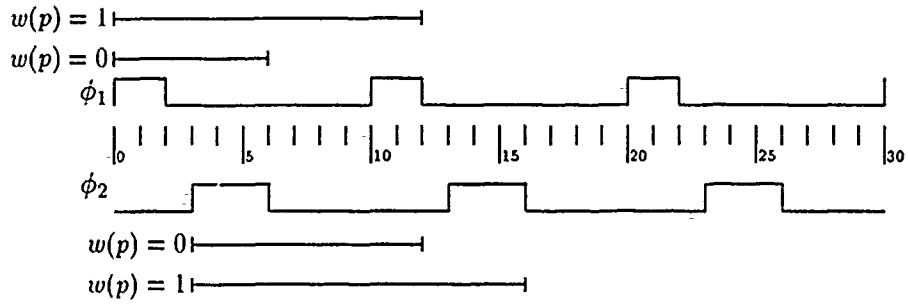


Figure 14: *A two-phase underlapped clock schedule. Distances are the maximum time period for a path of given weight beginning at a vertex preceded by the adjacent clock phase.*

in order to form constraints based on the phase of a latch at which a path begins, the single retiming value $r(u)$ is divided into a set of values $r_i(u)$, $i \in \{1 \ldots k\}$. In a general ILP constraint set, new constraints are added to maintain the sequential movement of latches using these "phased retiming" values. Given these new capabilities, "phase specific" constraints are derived which require the correct number of latches be placed along any path given any legal combination of phased retiming values. Although a complete set of ILP constraints may be formed and solved in this manner it is also possible to find a solution using a modification of the Bellman-Ford technique and a constraint set of approximately the same size as required for equal-phase retiming and using the same ($|V|$) number of variables.

## 7.1 Minimum Weight Requirements

The result of Corollary 5 on 14 may be re-written to provide a general equation for the minimum weight of a path. The equation must be solved on a case by case basis.

**Corollary 11:** *The weight $w(p)$ of any path $u \xrightarrow{p} v$ in a correctly timed, well-formed circuit graph $G$ using a valid $k$-phase clock schedule is bounded by:*

$$w(p) \geq k \left\lfloor \frac{d(p)-T_{P(u)}}{T_\Phi} \right\rfloor + \begin{cases} -1 & \text{if } (d(p)-T_{P(u)}) \bmod T_\Phi = 0 \\ 0 & \text{if } 0 < (d(p)-T_{P(u)}) \bmod T_\Phi \leq E_{P(u),P(u)+1} \\ 1 & \text{if } (d(p)-T_{P(u)}) \bmod T_\Phi \begin{cases} > E_{P(u),P(u)+1} \\ \leq E_{P(u),P(u)+1} \\ \quad + E_{P(u)+1,P(u)+2} \end{cases} \\ \vdots & \vdots \\ k-1 & \text{if } (d(p)-T_{P(u)}) \bmod T_\Phi > \sum_{j=0}^{k-2} E_{P(u)+j,P(u)+j+1} \end{cases}$$

*Proof Sketch:* From Corollary 5 we have:

$$\sum_{j=0}^{w(p)} E_{P(u)+j,P(u)+j+1} \geq d(p) - T_{P(u)}$$

$$\left\lfloor \frac{w(p)}{k} \right\rfloor T_\Phi + \sum_{j=0}^{w(p) \bmod k} E_{P(u)+j,P(u)+j+1} \geq d(p) - T_{P(u)}$$

Applying (mod $T_\Phi$) results in:

$$0 + [\sum_{j=0}^{w(p) \bmod k} E_{P(u)+j,P(u)+j+1}] \bmod T_\Phi \geq [d(p) - T_{P(u)}] \bmod T_\Phi. \quad (4)$$

Case 0: If $[d(p) - T_{P(u)}] \bmod T_\Phi \leq E_{P(u),P(u)+1}$, then $w(p) \bmod k = 0$ and by Eqn. 1 on page 9,

$$w(p) = k \left\lfloor \frac{d(p) - T_{P(u)}}{T_\Phi} \right\rfloor$$

The remaining cases follow similarly. □

We define $L_i(u, v)$ as the minimum number of latches required on a critical path from $u$ to $v$ when $P_r(u) = P(u) + i$, $i \in \{0 \ldots (k-1)\}$. $P_r(u) \equiv$ the phase of node $u$ in the retimed graph $G_r$. Values for $L_i(u, v)$ are computed by substitution of $D(u, v)$ in for $d(p)$ in Corollary 11. For notational convenience we will sometimes refer to $L_k$ which is equivalent to $L_0$ in all cases.

Now that a set of minimum weight values has been determined, it is necessary to form ILP constraint sets which require the correct number of latches on a path given the phase of the first node in the path. For example in a 2-phase system, for any pair of nodes $u$ and $v$ the following two constraints are required:

$$r(u) - r(v) \leq W(u, v) - L_0(u, v) \quad \text{for } P(u) = IP(u)$$
$$r(u) - r(v) \leq W(u, v) - L_1(u, v) \quad \text{for } P(u) = IP(u) + 1$$

These two constraints may not be implemented simultaneously because of the conditional expression on when each is valid. If both were imposed the minimum value of $L_i(u, v)$ would be the value required at all times on critical paths from $u$ to $v$. Instead we formulate a new set of variables which contain knowledge of the current phase of node $u$ and form constraints using those variables such that the correct value of $L_i(u, v)$ is imposed. The new variables for each node are known as "Phased Retiming" variables.

## 7.2 Phased Retiming Values

We now split each retiming value $r(u)$ into a set $\langle r_1(u), r_2(u), \ldots, r_k(u)\rangle$ according to the following definition:

$$r_i(u) = \begin{cases} \left\lfloor \frac{r(u)}{k} \right\rfloor + 1 & \text{for } i \leq r(u) \bmod k; \\ \left\lfloor \frac{r(u)}{k} \right\rfloor & \text{for } i > r(u) \bmod k; \end{cases}$$

Physically, $r_{P(u)+i}$ represents the number of $\phi_{P(u)+i}$ latches moved across vertex $u$ by a retiming. For notational convenience we will sometimes refer to $r_0$ which is equivalent to $r_k$ in all cases. In a sense we are exposing information about the phase of a node under any retiming given knowledge of the well-formed graph structure. The following lemma makes use of this information to form path-weight constraints which are specific to the current phase of the node beginning the path.

**Lemma 12:** *A set of values $\langle r_1(u), r_2(u), \ldots, r_k(u)\rangle$ is a set of phased retiming values as defined above iff the following constraints are met:*

$$\textit{Phased Variables:}\quad r(u) = \sum_{i=1}^{k} r_i(u)$$

$$\textit{Latch ordering:}\quad \left\{ \begin{array}{l} r_j(u) - r_i(u) \leq 1 \\ r_i(u) - r_j(u) \leq 0 \end{array} \right\} \textit{ for all } \left\{ \begin{array}{l} i \in \{1 \ldots k\} \\ j \in \{1 \ldots i-1\} \end{array} \right.$$

*Proof:* (If:) Restating the two latch ordering constraints:

$$\text{for all } r_j(u) \text{ where } j < i, \qquad r_j(u) \leq r_i(u) + 1. \tag{5}$$

$$\text{for all } r_j(u) \text{ where } j < i, \qquad r_j(u) \geq r_i(u). \tag{6}$$

If any value $r_i(u) = r_k(u) + 1$, as allowed by constraint 5, then all $r_k(u) + 1 \leq r_j(u) \geq r_i(u)$ for all $j \leq i$. Therefore $r_j(u) = r_i(u)$. In other words, if there is any $r_i(u)$ greater than $r_k(u)$ it will be greater by 1 and all values $r_j(u) = r_i(u) = r_k(u) + 1$ where $j \leq i$.

Thus, under the constraints, all $r_i(u)$ are equal or there exists exactly one value $r_i(u)$ such that for all $j \leq i$, $r_i(u) = r_j(u)$ and for all $j > i$, $r_i(u) = r_j(u) + 1$.

*Case 1:* All $r_i(u)$ are equal. Since $r(u) = \sum_{i=1}^{k} r_i(u)$ and by Corollary 3 $r(u) \bmod k = 0$ then $r_i(u) = \left\lfloor \frac{r(u)}{k} \right\rfloor$ satisfying the definition.

*Case2:* All $r_i(u)$ are not all equal. Therefore $r_i = R + 1, i \leq m$ and $r_i = R, i > m$ for some $R$ and $m$. Then:

$$\begin{aligned} r(u) &= \sum_{i=1}^{k} r_i(u) = \sum_{i=1}^{m} (R+1) + \sum_{i=m+1}^{k} (R) \\ &= m(R+1) + (k-m)R = kR + m \end{aligned}$$

Thus, $m = r \bmod k$ and $R = \left\lfloor \frac{r(u)}{k} \right\rfloor$ and $r_i(u)$ meets the definition above.

(Only If:) Summing the $r_i(u)$ values results in:

$$\begin{aligned} \sum_{i=1}^{k} r_i(u) &= k \cdot \left\lfloor \frac{r(u)}{k} \right\rfloor + r(u) \bmod k \\ &= r(u). \end{aligned}$$

Let $j = r(u) \bmod k$. Then definition, for all $i \leq j$, $r_i(u) = r_j(u)$, and for all $i > j$, $r_i(u) = r_j(u) - 1$. Thus the Latch Ordering constraints are true. □

## 7.3 Phase Specific Constraints

Using the expressions for minimum path weight and phased retiming values, it is now possible to implement *phase specific* constraints which impose weight restrictions on paths between nodes $u$ and $v$ conditional on the phase of node $u$.

**Theorem 13:** *A well-formed graph $G$ using a $k$-phase clock schedule $\Phi$ operates correctly under a retiming iff for all $u$ and $v$ in $V$:*

$$\sum_{i=1}^{k} r_i(u)[L_i(u,v) - L_{i-1}(u,v) + 1] - r(v) \leq W(u,v) - L_0(u,v).$$

*Proof:* (If:) We expand the above equation to:

$$\begin{aligned}
\sum_{i=1}^{k} r_i(u)L_i(u,v) - \sum_{i=1}^{k} r_i(u)L_{i-1}(u,v) + \sum_{i=1}^{k} r_i(u) - r(v) &\leq W(u,v) - L_0(u,v) \\
\sum_{i=1}^{k} r_i(u)L_i(u,v) - \sum_{i=0}^{k-1} r_{i+1}(u)L_i(u,v) + r(u) - r(v) &\leq W(u,v) - L_0(u,v) \\
\sum_{i=1}^{k} (r_i(u) - r_{i+1}(u))L_i(u,v) + r(u) - r(v) &\leq W(u,v) - L_0(u,v) \qquad (7)
\end{aligned}$$

*Case 1:* If $r(u) \bmod k = 0$, $P_r(u) = P(u)$ and $r_i(u) - r_{i+1}(u) = 0$ for all $i$. Thus Eqn. 7 becomes:

$$r(u) - r(v) \leq W(u,v) - L_0(u,v)$$

as desired.

*Case 2:* If $r(u) \bmod k = j$, $P_r(u) = P(u) + j$ and

$$r_i(u) - r_{i+1}(u) = \begin{cases} 1 & \text{for i=j} \\ -1 & \text{for i=k} \\ 0 & \text{otherwise} \end{cases}$$

Thus Eqn. 7 becomes:

$$\begin{aligned}
L_j(u,v) - L_0(u,v) + r(u) - r(v) &\leq W(u,v) - L_0(u,v) \\
r(u) - r(v) &\leq W(u,v) - L_j(u,v). \square
\end{aligned}$$

(Only If:) If the constraint set is not satisfied then for some constraint:

$$\sum_{i=1}^{k} r_i(u)[L_i(u,v) - L_{i-1}(u,v) + 1] - r(v) > W(u,v) - L_0(u,v).$$

Using the expansion to Eqn. 7:

$$\sum_{i=1}^{k} (r_i(u) - r_{i+1}(u))L_i(u,v) + r(u) - r(v) > W(u,v) - L_0(u,v). \qquad (8)$$

*Case 1:* If $r(u) \bmod k = 0$, $P_r(u) = P(u)$ and $r_i(u) - r_{i+1}(u) = 0$ for all $i$. Thus Eqn. 8 becomes:

$$r(u) - r(v) > W(u,v) - L_0(u,v)$$

For a critical path $u \xrightarrow{p} v$, $w_r(p) = W(u,v) - r(v) + r(u) < L_0(u,v)$. Thus the path weight is less than the minimum path weight required for correct operation.

*Case 2:* If $r(u) \bmod k = j$, $P_r(u) = P(u) + j$ and Eqn. 8 becomes:

$$\begin{aligned}
L_j(u,v) - L_0(u,v) + r(u) - r(v) &> W(u,v) - L_0(u,v) \\
r(u) - r(v) &> W(u,v) - L_j(u,v).
\end{aligned}$$

Again, for a critical path $u \xrightarrow{p} v$, $w_r(p) = W(u,v) - r(v) + r(u) < L_0(u,v)$. And the path weight in the retimed graph is less than that required for correct operation. $\square$

The complete set of constraints that must be met by a retiming of a multi-phase circuit graph $G$ using a valid $k$-phase clock schedule is:

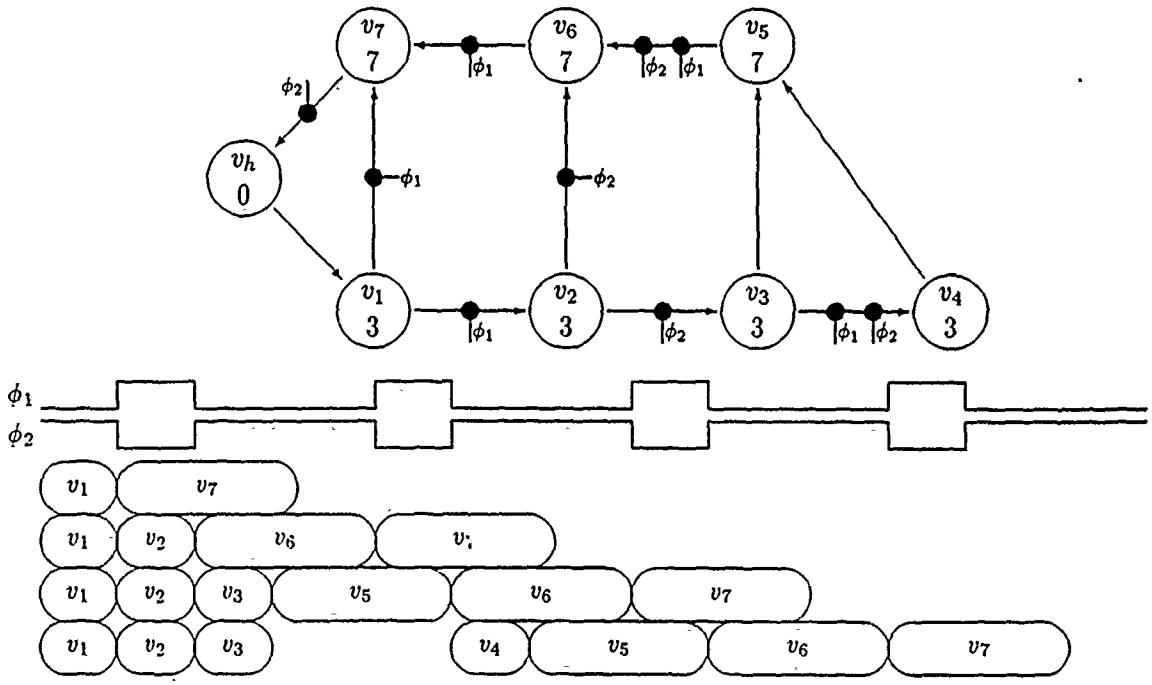


Figure 15: *Level-clocked Correlator example and resulting computational schedule when retimed to a 2-phase clock schedule where* $T_\Phi = 10$, $T_{\phi_1} = 3$, *and* $T_{\phi_2} = 7$.

$$
\begin{aligned}
\textbf{I/O:}\quad & r(v_h) = 0 \\
\textbf{Positive Edge Weight:}\quad & r(u) - r(v) \leq w(e) \text{ for all edges } u \xrightarrow{e} v \\
\textbf{Phased Variables :}\quad & r(u) - \textstyle\sum_{i=1}^{k} r_i(u) = 0 \\
\textbf{Latch Ordering:}\quad & \left\{ \begin{array}{l} r_j(u) - r_i(u) \leq 1 \\ r_i(u) - r_j(u) \leq 0 \end{array} \right\} \text{ for all } \left\{ \begin{array}{l} i \in \{1 \ldots k\} \\ j \in \{1 \ldots i-1\} \end{array} \right. \\
\textbf{Maximum Path Delay:}\quad & \textstyle\sum_{i=1}^{k} r_i(u)[L_i(u,v) - L_{i-1}(u,v) + 1] - r(v) \leq W(u,v) - L_0(u,v)
\end{aligned}
$$

Because $L_i(u,v)$ is a constant value throughout the retiming process, each of the above equations is a legal ILP constraint with a summing of variables multiplied by constants on the left hand side and a constant bound on the right hand side. Additionally, because of the highly restricted relationship between $r_i(u)$ and $r(u)$, the constraint set may still be solved using the Bellman-Ford algorithm as in [6, 7]. Intuitively, the Bellman-Ford technique holds one of the two variables in a two variable constraint constant while modifying the other variable such that the constraint is met. Holding $r(u)$ constant also holds each value of $r_i(u)$ constant, allowing manipulation of $r(v)$ to meet the constraint requirement.

$P(u) = [r(u) \bmod k]$ in a well formed graph, so an actual implementation of the Bellman-Ford approach can make use of a modified path constraint where the correct value of $L_{P(u)}$ is stored in a look-up table array indexed by $[r(u) \bmod k]$. This allows access to the correct number of registers required for a particular retiming value while ignoring the individual phased retiming variables required in the general approach. A variation on the Bellman-Ford algorithm provided in Appendix A makes use of this technique.

# 8 Summary and Future Work

We have described an efficient method for optimally retiming the class of well-formed, multi-phase, level-clocked circuits using valid clock with arbitrary-length phases. This not only is a large class of circuits widely used in practice; they are also circuits that can be easily produced by current sequential synthesis tools and optimal retiming of these circuits will become increasingly important.

Our next goal is to remove some of the restrictions we have placed on both circuit structure and clock schedules. Valid clock schedules can be redefined to assume a delay greater than zero between latches of specific phases. This introduces two-sided constraints and the manipulation of minimum delays as well as maximum delays. Work along these lines but in a different context has already been done by Shenoy [11] and Sakallah [5]. Extending the class of circuits beyond well-formed circuits places additional constraints on the movement of latches in the circuit. These constraints depend largely on the clock schedule itself and the implications of removing the ordering constraint on the correctness constraints.

The idea of retiming has also been used in the area of logic synthesis as a way of exposing and applying more of the functional relationships in a sequential circuit. Malik, Sentovich and Brayton describe the technique of peripheral retiming which allows registers in a sequential circuit to be moved to the periphery of the circuit, thus allowing the global resynthesis of the combinational logic as an single unit [8], and Borriello, Bartlett and Raju have explored the use of localized retiming combined with logic resynthesis to reduce the overall clock period. Our techniques allow this work to be extended to level-clocked circuits.

Sakallah et. al. describe a technique whereby the cycle time is minimized by adjusting the clock schedule instead of the circuit [5]. Typically there is not much freedom in the design of a clock schedule as it must conform to larger system constraints. However, it would be interesting to consider simultaneously adjusting the clock schedule and latch placement to minimize the cycle time.

In our circuit graphs, combinational components do not interact with the clock. In CMOS circuit design, however, there are circuits such as precharged logic gates whose inputs and outputs are synchronized to the clock. A future topic of research is to represent these types of combinational logic circuits in our circuit graphs so that retiming can be extended to more of the circuits encountered in practice.

Level-sensitive circuits have long been used for circuits where performance is important. Only recently, however, have algorithms for analyzing and manipulating these circuits become available. The potential benefits of level-sensitive circuits will make this a very fertile area of CAD research for some time to come.

## Acknowledgments

We want to thank Gaetano Borriello for valuable discussions and comments on this paper and Steve Burns for pointing us to the algorithm for finding the maximum-ratio-cycle and the construction for finding critical paths in level-clocked circuits.

# A Algorithm: Modified Bellman-Ford for Unequal-Phase Retiming

An important impact of introducing phased retiming variables to allow retiming of unequal phase clock schedules is that the resulting ILP constraints no longer have the form of the difference between two variables. The fact that all constraints were of this form was used in edge-clocked and equal-phase retiming to allow solution of the constraint sets by the Bellman-Ford single-source shortest paths problem [6, 3]. If this efficient short-cut to solution of the constraint set is no longer available, solving for the optimal clock period of the circuit will now require a general ILP solution method and a large additional expense incurred due to the increased number of variables and constraints required to guarantee legal combinations of those variables.

Fortunately it is possible to format the unequal-phase constraint sets in such a manner that a slightly modified version of the Bellman-Ford approach can solve them. This modification makes use of the fact that all phased retiming variables $\langle r_i(u)\rangle$ can be uniquely determined from any given value of $r(u)$. The process of mapping $r(u) : \langle r_i(u)\rangle$ imposes all of the additional constraints added to the constraint set for *Phased Variables* and *Latch Ordering*, and the proper number of latches required on an edge is stored as a complex weighting function where $w(u - v)$ is dependent on the value of $r(u)$ for each edge.

To present the modified version of the Bellman-Ford we proceed through the identical steps used in [3] to prove the correctness of the standard algorithm. The new problem being solved may be stated in the following manner:

*Problem:* Given a weighted, directed graph $G = (V, E)$, with weight function $w : (E, \delta(s, u)) \rightarrow \mathbf{R}$ mapping edges to real valued weights dependent on the shortest-path $\delta(s, u)$ from a source vertex $s$ to the head of the edge $u - v$, determine $\delta(s, u)$ for all $u \in V$.

*Example:* For a real-world analogy of the problem, assume you wish to travel by plane from New York to Chicago using the least expensive route. Fares are discounted at each departure point based on the expense of travel to that location. In keeping with airline tradition, for a given arrival cost the amount of discount is completely arbitrary and is contained in a look-up table accessed by arrival cost.

Figure 16 provides an illustration. The the source node, New York, is shown with two paths existing to the destination node Chicago. The value $\delta(s, s) \equiv 0$ so the edge weights from New York to Chicago and Houston are 6 and 10 respectively. This provides the value of $\delta(s, Houston) = 10$, therefore the edge weight from Houston to Chicago is -5. The shortest path from New York to Chicago is then the path through Houston and has a weight of 5.

In our modification of the shortest paths problem we are given a weighted, directed graph $G = (V, E)$, with weight function $w : (E, \delta(s, u)) \rightarrow \mathbf{R}$ mapping edges to real-valued weights using a function based on the weight of the shortest path leading to the node at the beginning of the edge. The weight of a path $p = \langle v_0, v_1, \ldots, v_k\rangle$ is the sum of the weights of the edges along the path in the identical manner to the original:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

The shortest path weight from $u$ to $v$ is:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{if there is a path from } u \text{ to } v, \\ \infty & \text{otherwise.} \end{cases}$$

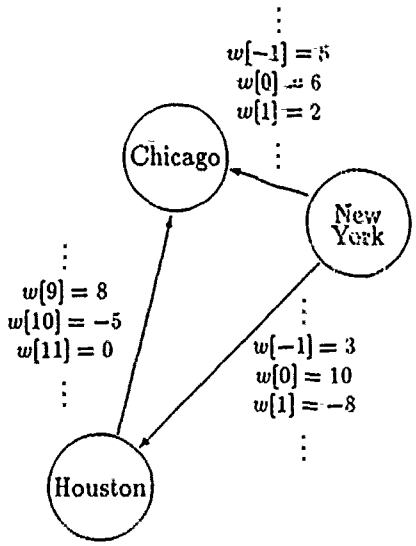


Figure 16: *A simple graph illustrating the shortest path problem with edge weights dependent on weight of shortest path leading to a the node beginning the edge. In this graph the source node is New York and edge weights of interest are shown in the form $w[\delta(s,u)]$ for each edge $u$ — $v$.*

Beginning with Lemma 25.1 in [3] we substitute our new definition for edge weight and show that each of the proofs leading to use of the Bellman-Ford algorithm still apply.

**Lemma 25.1′:** *(Subpaths of shortest paths are shortest paths)*

*Given a weighted, directed graph $G = (V, E)$ with weight function $w : (E.\delta(s,u)) \rightarrow \mathbf{R}$, let $p = \langle v_0, v_1, \ldots, v_k \rangle$ be a shortest path from vertex $v_1$ to vertex $v_k$ and, for any $i$ and $j$ such that $1 \leq i \leq j \leq k$, let $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ be the subpath of $p$ from vertex $v_i$ to vertex $v_j$. Then $p_{ij}$ is a shortest path from $v_i$ to $v_j$.*

*Proof:* The definition of the weight of a path has not changed from the original, hence this proof follows identically from the original: decomposing the path $p$ into $v_1 \xrightarrow{p_{1i}} v_i \xrightarrow{p_{ij}} v_j \xrightarrow{p_{jk}} v_k$, then $w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$. If there is a path $\acute{p}_{ij}$ from $v_i$ to $v_j$ with weight $w(\acute{p}_{ij}) < w(p_{ij})$. Then $v_1 \xrightarrow{p_{1i}} v_i \xrightarrow{\acute{p}_{ij}} v_j \xrightarrow{p_{jk}} v_k$ is a path from $v_1$ to $v_k$ whose weight $w(p_{1i}) + w(\acute{p}_{ij}) + w(p_{jk})$ is less than $w(p)$, which contradicts the premise that $p$ is the shortest path from $v_1$ to $v_k$. □

**Corollary 25.1′**

*Let $G = (V, E)$ be a weighted, directed graph with weight function $w : (E.\delta(s,u)) \rightarrow \mathbf{R}$. Suppose that a shortest path $p$ from a source $s$ to a vertex $v$ can be decomposed into $s \rightarrow u \rightarrow\!\!\!\!\rightarrow v$ for some vertex $u$ and path $\acute{p}$. Then the weight of a shortest path from $s$ to $v$ is:*

$$\delta(s,v) = \delta(s,u) + w(u,v,\delta(s,u)).$$

*Proof:* By Lemma 25.1′, subpath $\acute{p}$ is a shortest path from source $s$ to vertex $u$. Thus:

$$\delta(s,v) \;=\; w(p)$$

$$= w(p') + w(u, v, \delta(s, u))$$
$$= \delta(s, u) + w(u, v, \delta(s, u)). \square$$

It is now necessary to redefine the relaxation technique to make use of our new definition of edge weighting. Identically to the original work, for each vertex $v \in V$ we maintain an attribute $d[v]$, which is an upper bound on the weight of a shortest path from source $s$ to $v$. $d[v]$ is called a *shortest-path estimate.* The shortest-path estimates are initialized using the following procedure identical to the original:

INITIALIZE-SINGLE-SOURCE$(G, s)$

1. for each vertex $v \in V[G]$ do {
2. $d[v] \leftarrow \infty$;
3. $\pi[v] \leftarrow$NIL; }
4. $d[s] \leftarrow 0$;

The relaxation algorithm tests whether we can improve the shortest path to $v$ found so far by going through $u$ and if so, updating $d[v]$ and $\pi[v]$. The code for performing the relaxation step is only slighuly modified from the original in order to account for the more complex edge-weight function.

RELAX$(u, v, w)$

1. if $d[v] > d[u] + w(u, v, d[u])$ then {
2. $d[v] \leftarrow d[u] + w(u, v, d[u])$;
3. $\pi[u] \leftarrow u$; }

As shown in the following Lemmas and Corollaries, this new definition of relaxation supports the same properties required of the original relaxation function. Because these key proofs are supported, algorithms for finding shortest paths based on the relaxation method work for the new weighting function as well as for the old.

**Lemma 25.4´:**

*Let $G = (V, E)$ be a weighted, directed graph with weight function $w : (E, \delta(s, u)) \rightarrow$R, and let $u \rightarrow v \in E$. Then, immediately after relaxing edge $u \rightarrow v$ by executing RELAX(u,v,w), we have $d[v] \leq d[u] + w(u, v, d[u])$.*

*Proof:* If just prior to relaxing $u \rightarrow v$ we have $d[v] > d[u] + w(u, v, d[u])$, then $d[v] = d[u] + w(u, v, d[u])$ afterward. If, instead $d[v] \leq d[u] + w(u, v, d[u])$ just before the relaxation, then neither $d[u]$ nor $d[v]$ changes, and so $d[v] \leq d[u] + w(u, v, d[u])$ afterward. $\square$

**Lemma 25.5´:**

*Let $G = (V, E)$ be a weighted, directed graph with weight function $w : (E, \delta(s, u)) \rightarrow$R. Let $s \in V$ be the source vertex. and let the graph be initialized by INITIALIZE-SINGLE-SOURCE$(G, s)$. Then $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps on the edges of $G$. Moreover, once $d[v]$ achieves its lower bound $\delta(s, v)$, it never changes.*

*Proof:* The invariant $d[v] \geq \delta(s, v)$ is true after initialization since $d[s] = 0 \geq \delta(s, s)$ and $d[v] = \infty$ implies $d[v] \geq \delta(s, v)$ for all $v \in V - \{s\}$. Using contradiction, let $v$ be the first vertex for

which a relaxation set of an edge $u \rightarrow v$ causes $d[v] < \delta(s,v)$. Then, just after relaxing $u \rightarrow v$ we have:

$$\begin{aligned} d[u] + w(u,v,d[u]) &= d[v] \\ &< \delta(s,v) \\ &\leq \delta(s,u) + w(u,v,\delta(s,u)) \end{aligned}$$

which implies that $d[u] < \delta(s,u)$. But because relaxing $u \rightarrow v$ does not change $d[u]$, this inequality must have been true just before the edge was relaxed, which contradicts the choice of $v$ as the first vertex for which $d[v] < \delta(s,v)$. □

**Corollary 25.6′:**

*Suppose that in a weighted, directed graph $G = (V,E)$ with weight function $w : (E, \delta(s,u)) \rightarrow \mathbf{R}$, no path connects a source vertex $s \in V$ to a given vertex $v \in V$. Then after the graph is initialized by INITIALIZE-SINGLE-SOURCE$(G,s)$, we have $d[v] = \delta(s,v)$, and this equality is maintained as an invariant over any sequence of relaxation steps on the edges of $G$.*

*Proof:* By Lemma 25.5′, we always have $\infty = \delta(s,v) \leq d[v]$; thus $d[v] = \infty = \delta(s,v)$. □

**Lemma 25.7′:**

*Let $G = (V,E)$ be a weighted, directed graph with weight function $w : (E, \delta(s,u)) \rightarrow \mathbf{R}$, let $s \in V$ be a source vertex, and let $s \rightarrow u \rightarrow v$ be a shortest path in $G$ for some vertices $u, v \in V$. Suppose that $G$ is initialized by INITIALIZE-SINGLE-SOURCE$(G,s)$ and then a sequence of relaxation steps that includes the call RELAX$(u,v,w)$ is executed on the edges of $G$. If $d[u] = \delta(s,u)$ at any time prior to the call then $d[v] = \delta(s,v)$ at all times after the call.*

*Proof:* By Lemma 25.5′, if $d[u] = \delta(s,u)$ at some point prior to relaxing edge $u \rightarrow v$, then this equality holds thereafter. In particular, after relaxing $u \rightarrow v$ we have:

$$\begin{aligned} d[v] &\leq d[u] + w(u,v,d[u]) && \text{(by Lemma 25.4′)} \\ &= \delta(s,u) + w(v,v,\delta(s,u)) \\ &= \delta(s,v) && \text{(by Corollary 25.2′).} \end{aligned}$$

By Lemma 25.5′, $\delta(s,v)$ bounds $d[v]$ from below, thus $d[v] = \delta(s,v)$, and this equality is maintained thereafter.

Now that the above properties of relaxation have been proven for the modified relaxation technique, proofs for shortest paths algorithms dependent on the original relaxation technique may be shown to work for algorithms using the modified technique as well. This includes both Dijkstra's and the Bellman-Ford algorithm. Because the Bellman-Ford algorithm can handle negative weight edges, it is possible to restate the linear programming problem using difference constraints as a graph upon which a single-source, shortest paths algorithm is run to determine if a feasible solution to the constraint set exists or not.

Using the modified relaxation technique it is possible to restate the more complex constraints formed for unequal-phase retiming as a constraint graph where the weight of each edge is dependent on the weight of a shortest path to the beginning of the edge. In essence the set of variables $\langle r_1(u), \ldots, r_k(u) \rangle$ are determined directly from the modulo function on $r(u)$ and the corresponding constraint weighting selected. Thus $w(u,v)$ is a function of $r(u)$.

Path and edge constraints for unequal phase retiming may be written as a set $S$ of $m$ sub-sets of linear inequalities where each sub-set is of the form:

$$r(u) - r(v) \leq W(u,v) - L_i(u,v) \quad \text{for } i \in \{0 \ldots k\}$$

The constraints may be represented as a constraint graph $G = \langle V, E, w[0 \ldots k-1] \rangle$. For each variable $r(u)$ there is a vertex in $V$. For each set of constraints in $S$ there is an edge $u \xrightarrow{e} v$ in $E$ with weights $w(e)[i] = W(u,v) - L_i(u,v)$, $i \in \{0 \ldots k-1\}$. For each constraint in $S_2$ there is an edge $u \xrightarrow{e} v$ in $E_2$ with weight $w_2(e) = w(u,v)$.

BELLMAN-FORD($G, w, s$)

1. INITIALIZE-SINGLE-SOURCE($G, s$);
2. for $i \leftarrow 1$ to $|V[G] - 1|$ do {
3. for $u \rightarrow v \in E[G]$ do {
4. RELAX($u, v, w$); }}
5. for $u \rightarrow v \in E[G]$ do {
6. if $d[v] > d[u] + w(u, v, d[u])$ then return FALSE: }
7. return TRUE:

# References


[1] Karen Bartlett, Gaetano Borriello, and Sitaram Raju. Timing optimization of multiphase sequential logic. *IEEE Transactions on Computer-Aided Design*, 10(1):51–62, January 1991.

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

[3] Thomas H. Corman. Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* The MIT Press. 1990. Chapter 25.

[4] Alexander T. Ishii and Charles E. Leiserson. A timing analysis of level-clocked circuitry. In *Advanced Research in VLSI: Proc. of the 6th MIT Conference*, 1990.

[5] K.Sakallah, T. Mudge, and O. Olukotun. Analysis and design of latch-controlled synchronous digital circuits. In *Proc. 27th ACM-IEEE Design Automation Conf.*, pages 111–116, January 1990.

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

[7] C. Leiserson and J. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5-35, 1991.

[8] Sharad Malik, Ellen M. Sentovich, and Robert K. Brayton. Retiming and resynthesis: Optimizing sequential networks with combinational techniques. In *Proc. of the 23rd Hawaii Int. Conf. on System Sciences*, Kailua-Kona, III, January 1990.

[9] Nimrod Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, 1979.

[10] Giovanni De Micheli. Synchronous logic synthesis: Algorithms for cycle-time minimization. *IEEE Transactions on Computer-Aided Design*, 10(1):63–73, January 1991.

[11] Narendra Shenoy, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli. Retiming of circuits with single phase transparent latches. In *International Workshop on Logic Synthesis*, North Carolina, May 1991. MCNC.