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

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

[12] S. Malik, E. M. Sentovich, and R. K. Brayton. Retiming and resynthesis: Optimizing sequential networks with combinational techniques. In *Proc. of the 23rd Hawaii Int. Conf. on System Sciences*, Kailua-Kona, HI, Jan. 1990.

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

[14] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall Inc, Englewood Cliffs, New Jersey, 1982.

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

[16] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun. Optimal clocking of synchronous digital circuits. In *Proc. of the 1990 IEEE International Conf on CAD*, pages 552–555, Nov. 1990.

[17] J. B. Saxe. *Decomposable Searching Problems and Circuit Optimization by Retiming: Two Studies in General Transformations of Computational Structures.* PhD thesis, Carnegie-Mellon University, Aug. 1985.

[18] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Retiming of circuits with single phase transparent latches. In *International Workshop on Logic Synthesis at MCNC*, North Carolina, May 1991. MCNC.

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.

References

[1] K. Bartlett, G. Boriello, and S. Raju. Timing optimization of multiphase sequential logic. *IEEE Transactions on Computer-Aided Design*, 10(1):51–62, Jan. 1991.

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

[3] T. Corman, C. E. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990. Chapter 25.

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

[5] M. Hartmann and J. Orlin. Finding minimum cost to time ratio cycles with small integral transit times. Technical Report UNC/OR/TR/91-19, University of North Carolina, Chapel Hill, Oct. 1991.

[6] A. T. Ishii. *Timing in Level-Clocked Circuits*. PhD thesis, Massachusetts Institute of Technology, 1991. Available as MIT/LCS/TR-522.

[7] A. T. Ishii and C. E. Leiserson. A timing analysis of level-clocked circuitry. In *Advanced Research in VLSI: Proc. of the 6th MIT Conference*, pages 113–130, 1990.

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

[9] E. Lawler. *Combinatorial Optimization: Networks and Matriods*. Hold, Rinehart and Winston, New York, 1976.

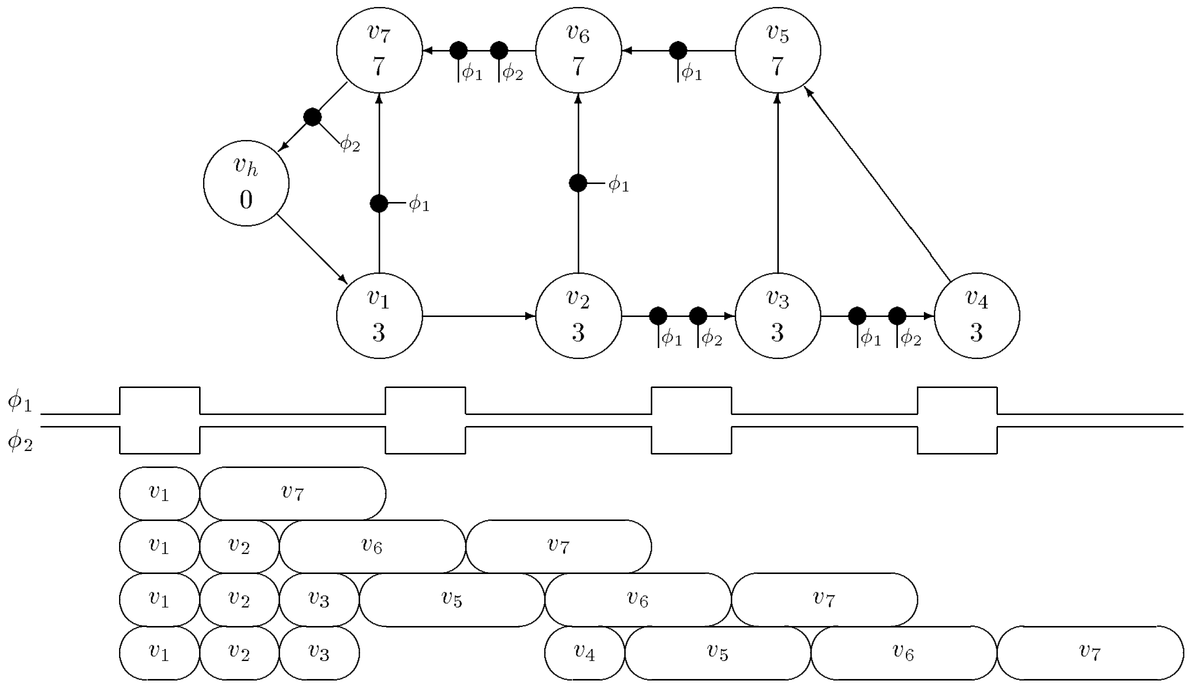

Figure 18: *Level-clocked Correlator example and resulting computational schedule when retimed to a 2-phase clock schedule where $T_\Phi = 10$, $T_{\phi_1} = 3$, and $T_{\phi_2} = 7$. Input and output signal phases are the same as in the original circuit. Retiming values are:* $\langle 4\ 2\ 0\ 0\ 0\ 0\ 1\ 3 \rangle$

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 [18] and Sakallah [15]. 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 [12] 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, and Borriello, Bartlett and Raju [1] 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, Mudge and Olukotun [15] describe a technique whereby the cycle time is minimized by adjusting the clock schedule instead of the circuit. 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

7.3.5 Cost of Using Bellman-Ford for Unequal-Phase Retiming

The resulting cost of the modified Bellman-Ford algorithm depends on the complexity of the weighting function $f(e)$. When used for unequal-phase retiming, computation of the weight function is linear; hence the overall algorithm complexity is not increased over equal phase retiming other than by a constant factor.

7.4 Unequal Phase Retiming of Correlator

Returning to the correlator circuit example as converted to a two-phase, level-clocked circuit, the circuit may be retimed using the unequal phase retiming techniques described. For example, given a clock schedule where $\phi_1 = 3$ and $\phi_2 = 7$, the circuit in Figure 17 is a legal retiming of the circuit which has interchanged the required phases of the input and output signals. For the same clock schedule the circuit in Figure 18 is a legal retiming which has not altered the required phases of input or output signals.

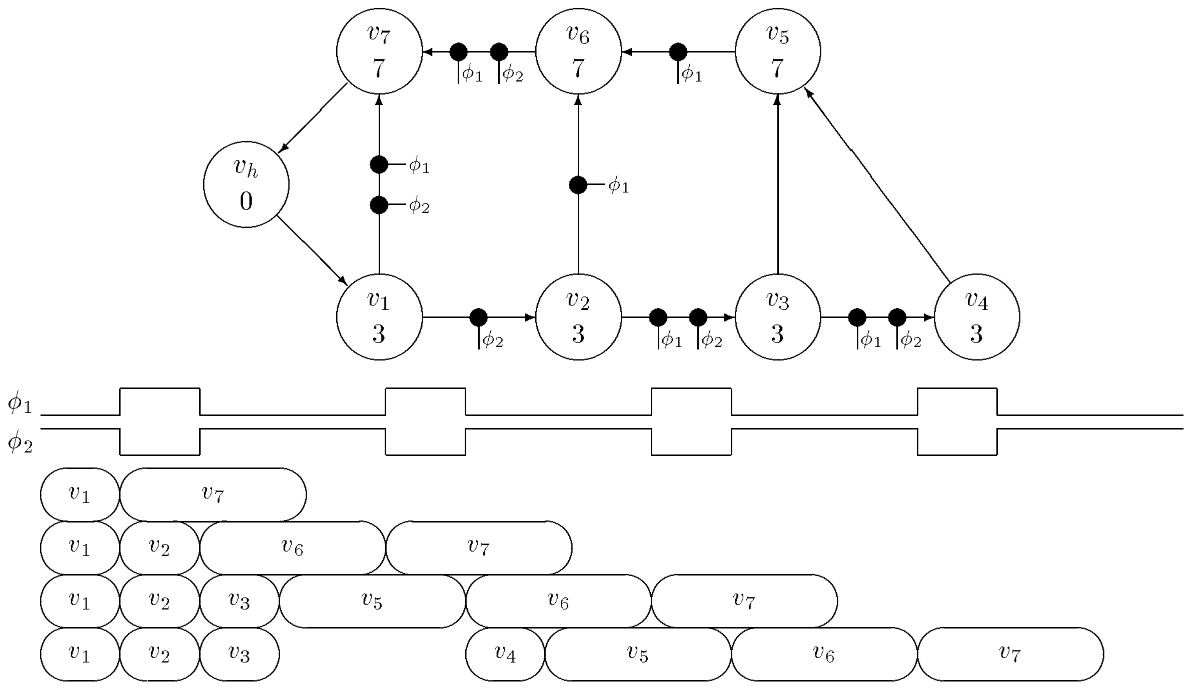

Figure 17: *Level-clocked Correlator example and resulting computational schedule when retimed to a 2-phase clock schedule where $T_\Phi = 10$, $T_{\phi_1} = 3$, and $T_{\phi_2} = 7$. Phases of input and output signals are reversed from the original circuit. Retiming values are:* $\langle 3\ 1\ 0\ 0\ 0\ 0\ 1\ 3 \rangle$

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 schedules 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

If maintaining the phase of input and output signals is important (as it probably is), a more complex weighting function may be applied to all edges $u \xrightarrow{e} v_h$ in the constraint graph which terminate at the host node. This new weight function $h(e, d_u)$ guarantees that the value of $r(v_h)$ is always set to an exact multiple of k and is written:

$$h(e, d_u) \equiv k \left\lfloor \frac{f(e, -d_u) + d_u}{k} \right\rfloor - d_u,$$

where $f(e, -d_u) \equiv f(e, r(u)) \equiv W(e) - L_{r(u)}(e)$ is the previous weighting function used. The following lemma shows that a constraint graph where $h(u \rightarrow v_h, d_u)$ is used for any constraint edge terminating at the host node, and where otherwise the edges are weighted with $f(e, r(u))$, will satisfy the restriction that subpaths of shortest paths are shortest paths as required by Lemma 14 without any additional restriction on valid clock schedules beyond ordering of enabling edges as was previously imposed.

Lemma 16: *For any path $u \overset{p}{\rightsquigarrow} v$ in a well-formed graph under a k-phase, valid clock schedule $\Phi = \{\phi_1 \dots \phi_k\}$, if $T_{\phi_i} + E_{i,i+1} \geq T_{\phi_{i+1}}$, then:*

$$h(u \rightarrow v_h, d_u) \leq h(u \rightarrow v_h, (d_u + \epsilon)) + \epsilon$$

Proof: By contradiction. We assume that $T_{\phi_i} + E_{i,i+1} \geq T_{\phi_{i+1}}$ but

$$h(u \rightarrow v_h, d_u) > h(u \rightarrow v_h, (d_u + \epsilon)) + \epsilon.$$

Using the minimum allowable value of $\epsilon = 1$ and proving the rest of the relationship holds through induction:

$$h(u \rightarrow v_h, d_u) \quad > \quad 1 + h(u \rightarrow v_h, (d_u + 1))$$

Expanding the functions $h(e, d_u)$ and $f(e, -d_u)$, first on the right hand side:

$$\begin{aligned} h(u \rightarrow v_h, d_u) &> 1 + k \left\lfloor \frac{W(u, v_h) - L_{(r(u)-1)} + (d_u + 1)}{k} \right\rfloor - (d_u + 1) \\ h(u \rightarrow v_h, d_u) &> k \left\lfloor \frac{W(u, v_h) - L_{(r(u)-1)} + (d_u + 1)}{k} \right\rfloor - d_u. \end{aligned}$$

And now expanding the left hand side:

$$\begin{aligned} k \left\lfloor \frac{W(u, v_h) - L_{r(u)} + d_u}{k} \right\rfloor - d_u &> k \left\lfloor \frac{W(u, v_h) - L_{(r(u)-1)} + (d_u + 1)}{k} \right\rfloor - d_u \\ \left\lfloor \frac{W(u, v_h) - L_{r(u)} + d_u}{k} \right\rfloor &> \left\lfloor \frac{W(u, v_h) - L_{(r(u)-1)} + d_u + 1}{k} \right\rfloor \\ \frac{W(u, v_h) - L_{r(u)} + d_u}{k} &> \frac{W(u, v_h) - L_{(r(u)-1)} + d_u + 1}{k} \\ -L_{i+1} &> -L_i + 1 \end{aligned}$$

However, by Theorem 15 on page 33, if $T_{\phi_i} + E_{i,i+1} \geq T_{\phi_{i+1}}$, then $L_{i+1}(p) \geq L_i(p) - 1$. Thus, $-L_{i+1}(p) \leq -L_i(p) + 1$, and the premise is contradicted. ∎

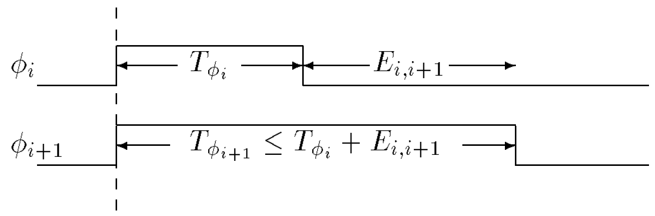

Figure 15: Using the modified Bellman-Ford algorithm requires that the enabling edge of ϕ_i must precede the enabling edge of ϕ_{i+1}

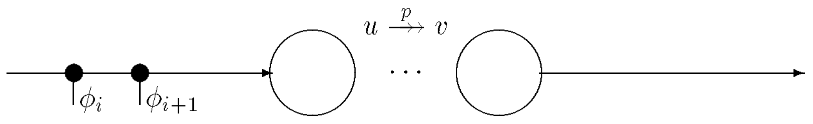

Figure 16: The maximum delay constraint beginning at the ϕ_i latch is less than the maximum delay constraint beginning at ϕ_{i+1}. Thus, even if $L_{i+1}(p) + 1 < L_i(p)$, paths beginning at a ϕ_{i+1} latch are still required to have weight at least $L_i(p) - 1$.

illustrated in Figure 15; however, this restriction does not eliminate any previously valid schedules of interest. For example, assume a particular path p requires $L_{i+1}(p) < L_i(p) - 1$. In a well-formed graph every ϕ_{i+1} latch is immediately preceded by a ϕ_i latch; thus the retiming can satisfy the path beginning with the ϕ_{i+1} latch and *not* satisfy the path beginning with the ϕ_i latch as shown in Figure 16. This is clearly an undesirable property of a clock schedule.

7.3.4 Maintaining Input/Output Phase

Under the definition of retiming presented in Section 2, the requirement that $r(v_h) = 0$ was said to be trivially satisfied since for any legal retiming where $r(v_h) \neq 0$ there existed an identical retiming $\acute{r}(v)$ where the I/O constraint was satisfied. The adjusted values for $\acute{r}(v)$ could be simply computed by:

$$\acute{r}(v) = r(v) - r(v_h).$$

However, for unequal phase retiming this transformation is no longer necessarily correct; instead, the phase of latches now have significance to the the retiming process that they did not have previously.

Specifically, if a solution to the retiming problem is found such that $(r(v_h) \bmod k) \neq 0$, then although the retiming does satisfy all timing constraints on the circuit, the phase of circuit inputs and outputs will differ from those in the original circuit. Additionally, if the transformation to $\acute{r}(v)$ such that $\acute{r}(v_h) = 0$ is performed, the phase of individual latches under $\acute{r}$ differ from the phases under $r(v)$. Thus the graph under $\acute{r}(v)$ may no longer satisfy the timing constraints. Unlike edge-triggered or equal-phase retiming, unless the retiming value of the host node is an integer multiple of the number of clock phases, an identical circuit does not exist where $r(v_h) = 0$ so that the input and output phases match those of the original circuit.

Since $r(u)$ is required to be integer-valued, the minimum increment of ϵ is 1:

$$f(e_{ij}, r(u)) \quad \leq \quad f(e_{ij}, r(u)-1)+1 \tag{9}$$

By induction on ϵ, if the above relationship holds true for $\epsilon = 1$, it must also hold true for any $\epsilon > 1$.

Unequal phase retiming requires the following constraints to be satisfied:

$$r(u) - r(v) \leq W(u,v) - L_{r(u) \bmod k}(p) \qquad \text{for all paths } u \xrightarrow{p} v$$

The restriction on the variant weight function requires the following condition be satisfied:

$$\begin{aligned} W(p) - L_{r(u) \bmod k}(p) &\leq W(p) - L_{(r(u)-1) \bmod k}(p) + 1 \\ -L_{r(u) \bmod k}(p) &\leq -L_{(r(u)-1) \bmod k}(p) + 1 \\ L_{r(u) \bmod k}(p) &\geq L_{(r(u)-1) \bmod k}(p) - 1 \\ L_{i+1}(p) &\geq L_i(p) - 1 \end{aligned}$$

The following theorem shows that the above condition is satisfied by any valid clock schedule on a well-formed graph if the enabling edges of the clock phases are ordered in time.

Theorem 15: *For any path $u \xrightarrow{p} v$ in well-formed graph under a k-phase, valid clock schedule $\Phi = \{\phi_1 \ldots \phi_k\}$, if $T_{\phi_{i+1}} \leq T_{\phi_i} + E_{i,i+1}$, then $L_{i+1}(p) \geq L_i(p) - 1$.*

Proof: By Contradiction. Assume that $T_{\phi_i} + E_{i,i+1} \geq T_{\phi_{i+1}}$ but $L_{i+1}(p) < L_i(p) - 1$ for some phase ϕ_i. Thus when $P_r(u) = i$, $L_{i+1}(p) + 1$ does not satisfy the required weight constraint on the path p:

$$d(p) > T_{\phi_i} + \sum_{j=i}^{i+L_{i+1}(p)+1} E_{j,j+1}.$$

On the other hand, when $P_r(u) = i+1$, the path weight requirement *is* satisfied by $L_{i+1}(p)$:

$$d(p) \leq T_{\phi_{i+1}} + \sum_{j=i+1}^{i+1+L_{i+1}(p)} E_{j,j+1}.$$

Combining the above relationships:

$$\begin{aligned} T_{\phi_i} + \sum_{j=i}^{i+L_{i+1}(p)+1} E_{j,j+1} &< T_{\phi_{i+1}} + \sum_{j=i+1}^{i+1+L_{i+1}(p)} E_{j,j+1} \\ T_{\phi_i} + E_{i,i+1} + \sum_{j=i+1}^{i+L_{i+1}(p)+1} E_{j,j+1} &< T_{\phi_{i+1}} + \sum_{j=i+1}^{i+1+L_{i+1}(p)} E_{j,j+1} \\ T_{\phi_i} + E_{i,i+1} &< T_{\phi_{i+1}} \end{aligned}$$

Contradicting the initial assumption. ■

The result of Theorem 15 shows that the Bellman-Ford algorithm can be used if the valid clock schedule also meets the constraint that both enabling *and* latching edges are ordered by phases, as

```
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 $d[v] > d[u] + w(u \twoheadrightarrow v, d[u])$ then return FALSE; }
7. return TRUE;
```

7.3.2 Applying the Bellman-Ford Algorithm

A constraint set for the unequal-phase retiming problem may be formed using constraints based either on the phase of the latch *preceding* the path, $r(u) - r(v) \leq W(u,v) - L_i(u,v)$ where the value $L_i(u,v)$ is selected based on the value of $r(u)$, or by constraints based on the phase of the latch *following* the path, $r(u) - r(v) \leq W(u,v) - \bar{L}_i(u,v)$ where the value $\bar{L}_i(u,v)$ is selected based on the value of $r(v)$. The trade off between the two constraint sets is the added complexity of inverting the constraint graph as shown below versus the added complexity of computing $\bar{L}_i(u,v)$. We have chosen to present the former approach because of the simpler formulation of $L(u,v)$ based on the results of Corollary 11 on Page 24.

We solve the unequal phase retiming problem using constraints of the form $r(u) - r(v) \leq f(r(u))$. A constraint graph is formed with an edge $r(u) \rightarrow r(v)$ for each constraint and with edge weights:

$$f(e_{uv}, r(u)) \equiv W(u,v) - L_{r(u) \bmod k}(u,v).$$

We solve the single source shortest path (SSSP) problem beginning from an additional source node s connected by a zero weight edge to every other node in the circuit graph to guarantee overall graph connectivity. Following a solution of the SSSP problem on a graph with no negative weight cycles, there exists a value d_u for each node u such that for any edge e_{uv} in the constraint graph:

$$d_v \leq d_u + f(e_{uv}, -d_u).$$

Setting the values of retiming variables $r(u) = -d_u$ and $r(v) = -d_v$ ensures that:

$$\begin{aligned} -r(v) &\leq -r(u) + f(e_{uv}, r(u)) \\ r(u) - r(v) &\leq f(e_{ij}, r(u)), \end{aligned}$$

guaranteeing a correct solution to the constraint set.

7.3.3 Restrictions on Clock Schedules

The restriction on the edge weighting function used in Lemma 14 can be written for our graph formulation as:

$$f(e_{uv}, -d_u) \leq f(e_{uv}, -(d_u + \epsilon)) + \epsilon.$$

Substituting the retiming variables $r(u) = -d_u$ and $r(v) = -d_v$ leads to:

$$f(e_{ij}, r(u)) \quad \leq \quad f(e_{ij}, (r(u) - \epsilon)) + \epsilon$$

Thus $d(v_j) < \acute{d}(v_j)$, and by Lemma 14, $d(v_k) = d(v_j) + w(v_j \twoheadrightarrow v_k, d(v_j)) \leq \acute{d}(v_k)$, contradicting our assumption that p_{ij} was not a subpath of a shortest path from v_1 to v_k. ■

The proof of Corollary 25.2 from [3] must be changed to account for the new method of weighting edges, although the statement of the corollary itself is only changed to add the required edge weight restrictions.

Corollary 25.2´: *Let $G = (V, E)$ be a weighted directed graph with weight function $w : (E, d(u)) \rightarrow \mathbf{R}$ such that:*

$$w(e, d(u)) \leq w(e, d(u) + \epsilon) + \epsilon.$$

Suppose that a shortest path p from source s to vertex v can be decomposed into $s \overset{\acute{p}}{\twoheadrightarrow} u \overset{e}{\rightarrow} v$ for some vertex u and path $\acute{p}$. Then the weight of a shortest path from s to v is: $\delta(s, v) = \delta(s, u) + w(e, \delta(u))$.

Proof: By contradiction. Assume that p is a shortest path but

$$\delta(v) = w(\acute{p}, 0) + w(e, w(\acute{p}, 0)) < \delta(u) + w(e, \delta(u)).$$

Because the weight of path $\acute{p}$ cannot be less than the weight of a shortest path to node u, $w(\acute{p}, 0) \geq \delta(u)$. Under the weight restriction:

$$\begin{aligned} w(e, \delta(u)) &\leq w(e, \delta(u) + (w(\acute{p}, 0) - \delta(u))) + (w(\acute{p}, 0) - \delta(u)) \\ \delta(u) + w(e, \delta(u)) &\leq w(\acute{p}, 0) + w(e, w(\acute{p}, 0)) \end{aligned}$$

Contradicting the assumption. ■

The Bellman-Ford (and Dijkstra's) algorithm determines a shortest path through the graph by successive "relaxation" of edges. Because negative weight constraints exist in the constraint graphs we apply the Bellman-Ford algorithm to finding a shortest paths solution. The relaxation subroutine of our modified algorithm is as follows:

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

It is necessary to show that the properties of relaxation proven in [3] hold for this new definition of relaxation when used with the variant weight edges in our constraint graphs. Specifically, Lemmas 25.3, 25.4, 25.5, 25.7 and Corollaries 25.2 and 25.6 from [3] hold for this new definition of relaxation when used on the variant weight graphs with the restricted weighting function. Because these key proofs are supported, the Bellman-Ford and Dijkstra's algorithms for finding shortest paths based on the relaxation method work using variant weight functions as well as for the standard constraint weight edges.

```
BELLMAN-FORD($G, w, s$)
  1. INITIALIZE-SINGLE-SOURCE($G, s$);
```

Lemma 14: *Given a weighted, directed graph $G = (V, E)$ with weight function $w : (E, d(u)) \rightarrow \mathbf{R}$ such that $w(e, d(u)) \leq w(e, d(u) + \epsilon) + \epsilon$, for any path $p = \langle v_0, v_1 \dots v_k \rangle$, if there are two paths $s \overset{q}{\twoheadrightarrow} v_0$ and $s \overset{\acute{q}}{\twoheadrightarrow} v_0$ such that $d(v_0) \leq \acute{d}(v_0)$, then the delays $d(v_k)$ and $\acute{d}(v_k)$ of the overall paths $s \overset{q}{\twoheadrightarrow} v_0 \overset{p}{\twoheadrightarrow} v_k$ and $s \overset{\acute{q}}{\twoheadrightarrow} v_0 \overset{p}{\twoheadrightarrow} v_k$ are also ordered such that $d(v_k) \leq \acute{d}(v_k)$.*

Proof: By induction on the number of edges in path p.
Basis: Since $d(v_0) \leq \acute{d}(v_0)$, by the weight restriction:

$$\begin{aligned} w(v_0 \rightarrow v_1, d(v_0)) &\leq w(v_0 \rightarrow v_1, d(v_0) + (\acute{d}(v_0) - d(v_0))) + \acute{d}(v_0) - d(v_0), \\ w(v_0 \rightarrow v_1, d(v_0)) &\leq w(v_0 \rightarrow v_1, \acute{d}(v_0)) + \acute{d}(v_0) - d(v_0), \\ d(v_0) + w(v_0 \rightarrow v_1, d(v_0)) &\leq w(v_0 \rightarrow v_1, \acute{d}(v_0)) + \acute{d}(v_0), \\ d(v_1) &\leq \acute{d}(v_1). \end{aligned}$$

Induction: Assume $d(v_i) \leq \acute{d}(v_i)$, then:

$$w(v_i \rightarrow v_{i+1}, d(v_i)) \leq w(v_i \rightarrow v_{i+1}, \acute{d}(v_i)) + \acute{d}(v_i) - d(v_i)$$

and $d(v_{i+1}) \leq \acute{d}(v_{i+1})$. ■

A modification to Lemma 25.1 from [3] which uses our new definition for edge weights provides the first step necessary to apply each of the subsequent proofs for the Bellman-Ford algorithm. The original lemma stated that all subpaths of any shortest path were also shortest paths. Proving the same for variant weight constraints would require that weight functions be strictly increasing with increasing weight of a path to the beginning of an edge, ie. $w(e, d(u)) < w(e, d(u) + \epsilon) + \epsilon$. This stronger limitation would change the result of Lemma 14 such that $d(v_k) < \acute{d}(v_k)$, leading to the identical formulation of the original Lemma 25.1; however, the shortest path algorithm requires only that some shortest path from the source to any node be entirely composed of subpaths which are also shortest paths. The relaxation process identifies shortest paths with this specific property. Other alternate shortest paths to a node may have subpaths which are not shortest paths. Allowing these alternative paths to exist allows the edge weight function to be restricted by a less-than-or-equal relationship as in Lemma 14.

Lemma 25.1´:
Given a weighted, directed graph $G = (V, E)$ with weight function $w : (E, d(u)) \rightarrow \mathbf{R}$ such that:

$$w(e, d(u)) \leq w(e, d(u) + \epsilon) + \epsilon,$$

if there is a path from source v_1 to v_k then there is a shortest path $p = \langle v_1, v_2, \dots, v_k \rangle$ such that every subpath of p from vertex v_i to vertex v_j is a shortest path from v_i to vertex v_j.

Proof: By contradiction. Assume two subpaths p_{ij} and $\acute{p}_{ij}$ exist such that $w(p, d(v_i)) < w(\acute{p}, d(v_i))$ but p_{ij} is not a subpath of any shortest path to node v_k. The delay of the corresponding paths from source s to v_j are:

$$\begin{aligned} d(v_j) &= d(v_1) + w(p_{1j}, d(v_1)) + w(p_{ij}, d(v_i)) \\ \acute{d}(v_j) &= d(v_1) + w(p_{1j}, d(v_1)) + w(\acute{p}_{ij}, d(v_i)). \end{aligned}$$

suitable weight functions f. 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. The restriction on f places constraints on the clock schedules allowed under retiming; these, however, include all schedules of interest in practice.

7.3.1 Using the Bellman-Ford Algorithm with Variant Weights

We now show how the Bellman-Ford algorithm can be used to solve "variant-weight" constraints where all constraints are of the form:

$$x_j - x_i \quad \leq \quad f(x_i) \tag{7}$$

or of the form:

$$x_j - x_i \quad \leq \quad f(x_j). \tag{8}$$

The variant weight technique may be used to solve the first form of constraint directly; however, the second form requires reversing the direction of edges in the constraint graph over which the shortest paths algorithm is to be run. We first show that the shortest paths algorithm can be used to solve constraints of the first form and later reverse the constraint graph to allow solution of the second form.

As in the standard method of using shortest paths algorithms to solve constraints on the difference between two variables, we first form a graph from the constraint set with a node for each variable x_i to be constrained and an edge $x_i \rightarrow x_j$ with weight $f(x_i)$ for each constraint $x_j - x_i \leq f(x_i)$. Additionally a single source node s is added and a zero-weight edge from the source node to each other node in the constraint graph.

Each node u_i is assigned a value $d(u_i)$ which is the weight of some path from the source s to u_i. The goal of the Bellman-Ford algorithm is to minimize $d(u_i)$ by finding the shortest path from s to u_i. The resulting shortest path weight $\delta(u_i)$ is the solution for x_i.

Each edge $u \overset{e}{\rightarrow} v$ in the graph is assigned a weight function $w(e, d(u))$ (or $w(u \rightarrow v, d(u))$. The weight of a path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is:

$$w(p, d(v_0)) = \sum_{i=1}^{k} w(v_{i-1} \rightarrow v_i, d(v_{i-1})).$$

The correctness of the shortest-path algorithm relies on the monotonicity of shortest paths in the graph. That is, the weight of a shortest path from the source node s to the end of a path $u \twoheadrightarrow v$ must increase monotonically with the value of $d(u)$:

$$\text{if } w(s \overset{p}{\twoheadrightarrow} u, 0) < w(s \overset{\acute{p}}{\twoheadrightarrow} u, 0) \text{ then } d(s \overset{p}{\twoheadrightarrow} u \twoheadrightarrow v) \leq d(s \overset{\acute{p}}{\twoheadrightarrow} u \twoheadrightarrow v).$$

We show in Lemma 14 that the following property on the edge weight function is sufficient to ensure monotonicity:

$$\forall \text{ edges } e \in E \qquad w(e, d(u)) \leq w(e, d(u) + \epsilon) + \epsilon,$$

where $\epsilon \in \mathbf{R}$ is any positive value.

Thus Eqn. 5 becomes:

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

(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. 5:

$$\sum_{i=1}^{k}(r_i(u) - r_{i+1}(u))L_i(u,v) + r(u) - r(v) > W(u,v) - L_0(u,v). \tag{6}$$

Case 1: If $(r(u) \bmod k) = 0$, then $P_r(u) = P(u)$ and $r_i(u) - r_{i+1}(u) = 0$ for all i. Thus Eqn. 6 becomes:

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

For a critical path $u \overset{p}{\rightsquigarrow} v$, $w_r(p) = W(u,v) - r(v) + r(u) < L_0(u,v)$. Thus the path weight is less than the minimum path weight required for correct operation.
Case 2: If $(r(u) \bmod k) = j$, then $P_r(u) = P(u) + j$ and Eqn. 6 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 $u \overset{p}{\rightsquigarrow} v$, $w_r(p) = W(u,v) - r(v) + r(u) < L_j(u,v)$. And the path weight in the retimed graph is less than that required for correct operation. ■

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:

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

7.3 Retiming Using Variant Weight Constraints

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 these constraints can be written as $r(u) - r(v) \leq f(r(u))$, the constraint set may also be solved by the Bellman-Ford algorithm for

Thus, $l = r(u) \bmod k$, $R = \left\lfloor \frac{r(u)}{k} \right\rfloor$ 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 $l = r(u) \bmod k$. Then, by definition, for all $j \leq l$, $r_l(u) - r_j(u) \leq 0$, and for all $j > l$, $r_j(u) - r_l(u) \leq 1$. Thus the Latch Ordering constraints are true. ■

7.2.2 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 critical 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 Φ 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:

$$\begin{aligned} \sum_{i=1}^{k} r_i(u)L_i(u,v) - \sum_{i=1}^{k} r_i(u)L_{i-1}(u,v) + \sum_{i=1}^{k} r_i(u) - r(v) &\leq W(u,v) - L_0(u,v) \\ \sum_{i=1}^{k} r_i(u)L_i(u,v) - \sum_{i=0}^{k-1} r_{i+1}(u)L_i(u,v) + r(u) - r(v) &\leq W(u,v) - L_0(u,v) \\ \sum_{i=1}^{k}(r_i(u) - r_{i+1}(u))L_i(u,v) + r(u) - r(v) &\leq W(u,v) - L_0(u,v) \qquad (5) \end{aligned}$$

Case 1: If $(r(u) \bmod k) = 0$, then $P_r(u) = P(u)$ and $r_i(u) - r_{i+1}(u) = 0$ for all i. Thus Eqn. 5 becomes:

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

as desired.

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

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

array is no longer obviously *Totally Uni-Modular* (TUM) as described in [14], although there is some evidence that the array may be TUM due to the fact that a modification of the Bellman-Ford algorithm can solve the constraint array as discussed in Section 7.3. In general, solving an ILP constraint array which is not TUM is a NP-complete problem.

7.2.1 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, $r_i(u)$ 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 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:*

$$\begin{array}{rl} \textit{Phased Variables:} & r(u) = \sum_{i=1}^{k} r_i(u) \\ \textit{Latch ordering:} & \left\{ \begin{array}{c} r_j(u) - r_i(u) \leq 1 \\ r_i(u) - r_j(u) \leq 0 \end{array} \right\} \textit{for all } j < i \end{array}$$

Proof: (If:) The two latch ordering constraints can be combined as:

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

In other words, if there is any $r_l(u)$ greater than $r_k(u)$, it will be greater by at most 1 and for all $j < l$, $r_j(u) = r_l(u) = r_k(u) + 1$. Thus, under the constraints, all $r_i(u)$ are equal (*Case 1* below) or there exists exactly one value $r_l(u)$ such that for all $j \leq l$, $r_l(u) = r_j(u)$ and for all $j > l$, $r_l(u) = r_j(u) + 1$ (*Case 2* below).

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.

Case 2: The $r_i(u)$ are not all equal. Therefore for $j \leq l$, $r_j = R + 1$, and for $j > l$, $r_j = R$, for some R and l. Then:

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

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

Applying (mod T_Φ) results in:

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

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 $P_r(u) = P(u) + i$, $0 \leq i < k$, where $P_r(u) \equiv$ the phase of node u in the retimed graph G_r. For notational convenience we will sometimes refer to L_k which is equivalent to L_0. Values for $L_i(u,v)$ are computed by substitution of $D(u,v)$ in for $d(p)$ in Corollary 11. Note that the identification of a critical path based on Lemma 8 on 16 is *not* dependent on the phase of the initial node, $P(u)$. Thus we can use the same methodology for finding critical paths in graphs controlled by an unequal phase clock as we did for equal phase clocks. This technique was provided in Section 5.1.

Now that a set of minimum weight values has been determined, it is necessary to form ILP constraint sets that 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:

$$\begin{array}{ll} r(u) - r(v) \leq W(u,v) - L_0(u,v) & \text{for } P_r(u) = P(u) \\ r(u) - r(v) \leq W(u,v) - L_1(u,v) & \text{for } P_r(u) = P(u) + 1 \end{array}$$

These two constraints are not in effect simultaneously because of the conditional expression on which each depends. 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 encode 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 A General ILP Solution Method

In solving the unequal-phase retiming problem using a general ILP approach, the retiming values $r(u)$ are first split into sets of retiming values $r_i(u)$, for $1 \leq i \leq k$. Each "phased retiming" value $r_i(u)$ indicates the number of phase $P(u) + i$ latches moved across node u. New constraints are added to maintain the sequential movement of latches using these phased retiming values. Given these new variables, "phase specific" constraints can be derived which require the correct number of latches to 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 while optimizing a non-trivial cost function, the solution process may be very inefficient. In particular, the resulting ILP coefficient

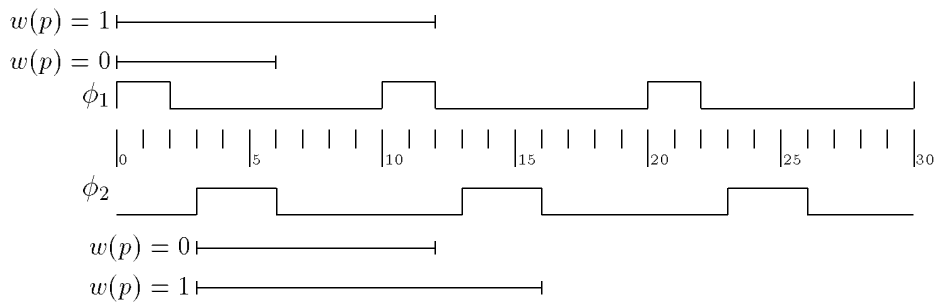

Figure 14: *A two-phase underlapped clock schedule. Distances are the maximum time period for a path of given weight beginning at a vertex preceded by the adjacent clock phase.*

in 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 $L_i(u,v)$, $0 \leq i < k$, indexed relative to the initial arrangement of latches in the circuit graph. Given the new required weighting dependent on the phase of the latch currently beginning the path, three methods of solution are available. The first is a standard ILP solution. The second method makes use of the fact that all interesting constraints in the ILP solution may be formed as a constraint involving the difference of two variables and a weight which is a function of one of those variables. A modification to the Bellman-Ford algorithm can be used to solve this variant weight constraint set efficiently. Finally, the third solution method makes use of the modification to the Bellman-Ford algorithm to solve an asymptotically faster algorithm designed around a Mixed-Integer-Linear programming problem.

7.1 Minimum Weight Requirements

The result of Corollary 5 on page 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 $u \overset{p}{\twoheadrightarrow} v$ in a correctly timed, well-formed circuit graph G using a valid k-phase clock schedule is bounded by:*

$$w(p) \geq k \left\lfloor \frac{d(p)-T_{P(u)}}{T_\Phi} \right\rfloor + \begin{cases} & \textit{if } (d(p)-T_{P(u)}) \textit{ mod } T_\Phi \textit{ is:} \\ -1 & = 0 \\ 0 & \leq E_{P(u),P(u)+1} \\ 1 & \begin{cases} > E_{P(u),P(u)+1} \\ \leq E_{P(u),P(u)+2} \end{cases} \\ \vdots & \vdots \\ k-1 & > E_{P(u),P(u)+k-1} \end{cases}$$

Proof Sketch: From Corollary 5 we have:

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

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
1	142	14.00
2	285	12.74
3	474	11.56
4	658	11.72
5	838	11.72
6	880	11.02
7	1056	10.94

N	# path constraints	$T_{\Phi_{opt}}$ for Ideal 2-equal-phase
8	1225	11.17
9	1309	11.02
10	1437	10.78
12	1757	10.78
15	2098	10.78
16	2169	10.70
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.*

Maximum path delay: $r(u) - r(v) \leq W(u,v) - L(u,v)$ for any $0 < L(u,v) < N$
Limited path delay: $r(u) - r(v) \leq W(u,v) - (L(u,v) - 1)$ for any $L(u,v) \geq N$

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 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. 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. This example also shows that borrowing is in fact done over long paths in the optimally timed circuit.

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 the maximum length of paths of weight 0 and 1, beginning at a latch controlled by each phase of a 2-phase clock. Neither the edge-clocked retiming methods from [10, 11] 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

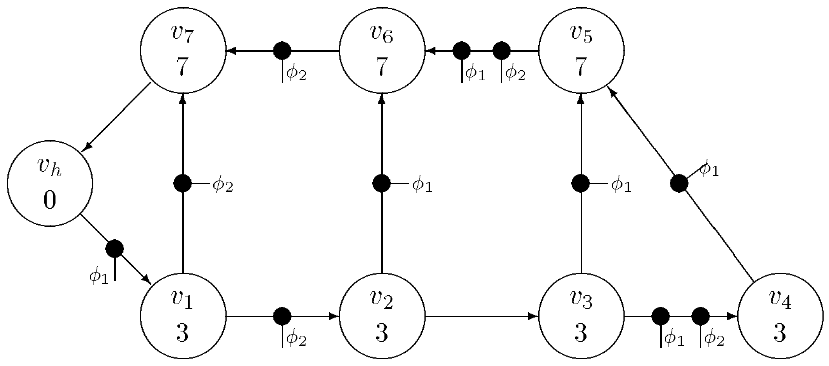

Figure 13: *The correlator circuit optimally retimed using a 2-phase, equal-period clock where $T_\phi = 0.4 \cdot T_\Phi$; ${T_\Phi}_{opt} = 10.345$ units.*

step has cost $O(|V|^3)$ providing theoretical bounds of $O\left(|V|^3 \log\frac{{T_\Phi}_{opt}}{T_{c_c}}\right)$ for Step 3 and $O(b \cdot |V|^3)$ for Step 4 where b is the number of bits of accuracy desired.

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 $T_\phi = 0.4 \cdot T_\Phi$ giving an underlap between phases of $0.1 \cdot T_\Phi$. In this example we retime the correlator circuit graph using such a clock schedule.

As in the previous example $T_{c_c} = 10$; 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 subpaths whereas in edge-clocked graphs constraints on long paths are usually redundant with a shorter subpath. 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:

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

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

For many cases the critical paths for $T_{\Phi_{opt}}-$ and $T_{\Phi_{opt}}+$ are the same and so we can determine an exact set of potential optimum clock periods. As shown in Figure 11 on page 16, the value of slack for a given path is linear over all clock periods, thus there can be at most one intersection point between the slack values for two paths. Below this intersection point one path will always have less slack than the other and above will always have greater slack. Once two clock periods $T_{\Phi_{opt}}-$ and $T_{\Phi_{opt}}+$ have been determined for which both W matrices are equivalent for all u, v, no further computation of critical paths is needed.

The following theorem uses the fact that for an optimal retiming of a level-clocked circuit graph (for a period greater than the critical cycle period) there will exist some critical path which exactly meets the minimum weight requirement. If this were not true there would exist a faster clock period for the same weighting. Assuming a range over the optimal period is found such that the critical paths do not change, 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. Step 4 of the previous algorithm is terminated when matching critical paths are found and is followed by a binary search over the set of potential optimum 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_Φ. Because the value T_ϕ is proportional to T_Φ, substitute in duty cycle $(T_d \cdot T_\Phi)$ instead which remains constant for the clock schedule. ∎

Note that real circuit graphs have built-in error due to estimation of combinational logic delays, and thus the value of generating a set of precise possible optimum clock periods is questionable. Moreover, large, complex circuits with many combinations of possible path delays and weights have a densely populated set of possible optimum clock periods. Thus a search for the precise optimum clock period is of limited utility.

In the worst case, the algorithm stated provides an estimate of the optimal retiming of the graph to the desired level of accuracy. The complexity of Step 1 is provably $O(d \cdot |E|)$ using the algorithm by Hartmann et. al. [5]. Although $|E|$ may be as large as $|V|^2$, in practice circuit graphs have $|E| = O(|V|)$. Moreover, Burns' algorithm typically performs in $O(|E|)$ [2]. (Lower worst case bound algorithms may be found in Lawler [9]; however, in practice, this step is not limiting.) Thus algorithmic complexity is driven by either Step 3, the number of steps upward required to find $T_{\Phi_{opt}}+$, or Step 4, the number of steps downward required to find the desired result. Each trial

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 ϕ_1, ϕ_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_c} 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 $u \twoheadrightarrow v$	# latches req'd	$D(p)$
$1 \twoheadrightarrow 4, 1 \twoheadrightarrow 6, 2 \twoheadrightarrow 5$ $5 \twoheadrightarrow 6, 6 \twoheadrightarrow 7, 7 \twoheadrightarrow 2$	1	$> (1 \cdot T_\Phi)$
$1 \twoheadrightarrow 5, 2 \twoheadrightarrow 7, 3 \twoheadrightarrow 6$ $4 \twoheadrightarrow 6, 6 \twoheadrightarrow 1, 7 \twoheadrightarrow 3$ $7 \twoheadrightarrow 6$	2	$> (1.5 \cdot T_\Phi)$

Path $u \twoheadrightarrow v$	# latches req'd	$D(p)$
$5 \twoheadrightarrow 7, 6 \twoheadrightarrow 3$ $7 \twoheadrightarrow 5$	3	$> (2 \cdot T_\Phi)$
$3 \twoheadrightarrow 1, 4 \twoheadrightarrow 1, 5 \twoheadrightarrow 2$ $6 \twoheadrightarrow 4, 6 \twoheadrightarrow 5$	4	$> (2.5 \cdot T_\Phi)$
$4 \twoheadrightarrow 3, 5 \twoheadrightarrow 4$	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. 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_Φ. In level-clocked circuits the critical path may be of non-zero weight, and the critical path between two vertices may differ for differing clock periods. In general we cannot identify a set of potential optimum clock periods over which to search, and instead perform a binary search over a range to the desired accuracy.

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 $T_{c_c} = \max\left\{k\left(\frac{d(c)}{w(c)}\right)\right\}$.

2. Attempt to retime to T_{c_c}; if successful $T_{\Phi_{opt}} = T_{c_c}$.

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_{ϕ_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_ϕ 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 [10, 11] 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_Φ:

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

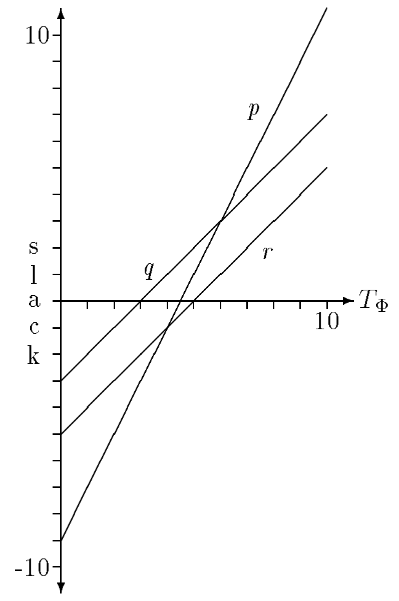

Figure 12: *Plot of slack vs. clock period for paths p, q and r.*

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_Φ. The resulting plot 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 ≥ 0, if $W(u,v,T_{\Phi_{opt}}-) = \min\{w(p) \mid u \overset{p}{\rightsquigarrow} v\}$ 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

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 $u \overset{p}{\twoheadrightarrow} v$ 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 $u \overset{q}{\twoheadrightarrow} v$ 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, $w_r(q) - w_r(p) = km = w(q) - w(p)$ where $k =$ the 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 $u \overset{p_c}{\twoheadrightarrow} v$. This in turn requires identifying the path which minimizes the quantity $\{w(p)\frac{T_\Phi}{k} - d(p)\}$ over all paths $u \overset{p}{\twoheadrightarrow} v$. 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.

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

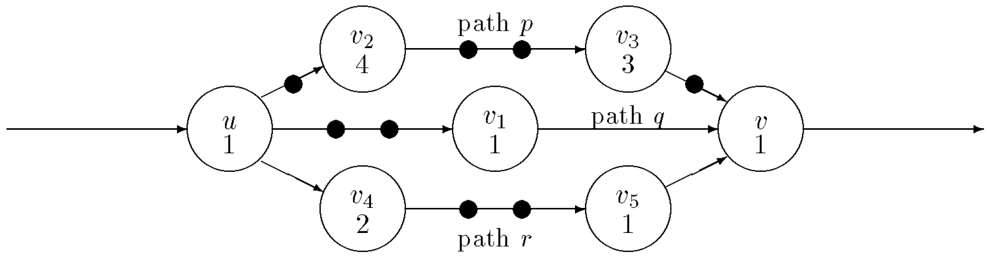

Figure 11: *Critical paths in level-clocked circuits may not be the same as critical paths in edge-clocked circuits.*

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 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 $u \overset{p}{\twoheadrightarrow} v$ 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	$u \to v_2 \to v_3 \to v$	4	9
q	$u \to v_1 \to v$	2	3
r	$u \to v_4 \to v_5 \to v$	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 $w_r(p) = 6$. 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 $u \overset{p}{\twoheadrightarrow} v$ in a well-formed circuit is a critical path if*

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

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 $l_0 \overset{c}{\twoheadrightarrow} l_{n+1}$ 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 $l_0 \overset{p}{\twoheadrightarrow} l_{n+1}$ 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:*

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

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:

$$\begin{aligned} d(c) &\leq \sum_{i=1}^{\frac{w(c)}{k}} T_\Phi = w(c) \cdot \frac{T_\Phi}{k}, \\ T_\Phi &\geq k\left(\frac{d(c)}{w(c)}\right). \end{aligned}$$

■

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_c, 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 $\frac{d(c_c)}{w(c_c)}$ for a critical cycle in the graph may be found by setting the values $\alpha(u \overset{e}{\rightarrow} v) = d(v)$ 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 [13], Hartmann et. al. [5] and Burns [2]. In particular, the algorithm by Hartmann has a provable running time of $O(d \cdot |E|)$ where d is the longest delay of a path with $|V|$ edges. The resulting $\frac{d(c_c)}{w(c_c)}$ value provides a fast 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.

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 $l_0 \overset{p}{\twoheadrightarrow} l_{n+1}$ 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 $l_0 \overset{p_1}{\twoheadrightarrow} l_1$ and $l_1 \overset{p_2}{\twoheadrightarrow} l_{n+1}$. 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 violated and show by contradiction that this cannot be true. Since $d(p) = \sum_{i=0}^{n} d(q_i)$ where $l_i \overset{q}{\twoheadrightarrow} l_{i+1}$, and from our assumption $d(q_i) \leq A_{l_{i+1}} - D_{l_i} + E_{P(l_i),P(l_{i+1})}$, 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 $l_0 \overset{p}{\twoheadrightarrow} l_{n+1}$ 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 $A_{l_{n+1}} = T_\Phi$. ■

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:

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

■

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 Φ. 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.

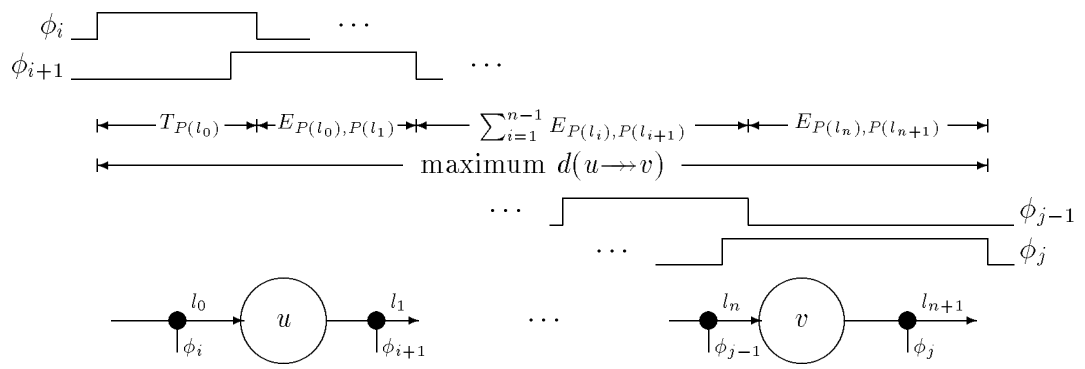

Figure 10: *Graphical representation of the constraint on the simple path delay between two latches* l_0 *and* l_{n+1} *in a correctly operating circuit.*

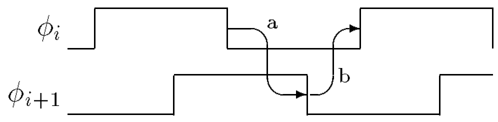

Figure 9: *Graphical representation of the constraints on the clock phases that are required for correct operation of a well-formed level-clocked circuit.*

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 $l \overset{p}{\twoheadrightarrow} m, A_m = D_l - E_{P(l),P(m)} + d(p) \leq T_\Phi$.

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

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 ϕ_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 $l \overset{p}{\twoheadrightarrow} m$ 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:*

$P_r(u) = [P(u) + r(u)]$ *mod* k.

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 $l \overset{p}{\rightsquigarrow} m$ 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 ϕ_{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 [10]. ■

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

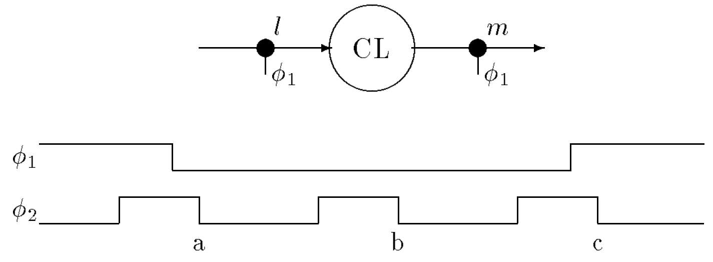

Figure 8: *An illustration of a circuit for which it is not clear what the maximum computational time available for logic block CL is.*

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 simplified 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 phase are *enabled* during its active interval and *disabled* during its passive interval. The transitions *into* and *out of* the active interval are called the *enabling* and *latching* edges respectively. We refer to the clock phase controlling latch l by $P(l)$.

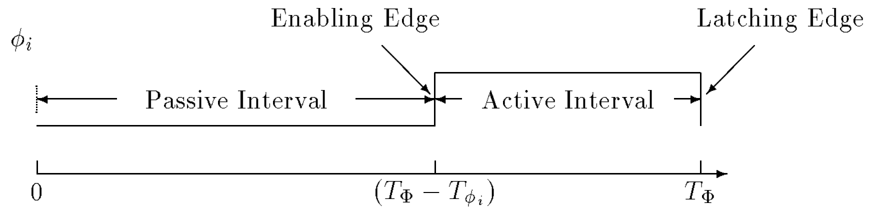

Figure 6: *Diagram from Sakallah et. al. showing a clock phase ϕ_i and its local time zone.*

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_Φ. 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 ϕ_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 ϕ_i in the clock set is referred to as ϕ_{i+1} with phase $\phi_{k+1} \equiv \phi_1$ and $\phi_{1-1} \equiv \phi_k$.

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 ϕ_i, E_{ij} changes the frame of reference to the *next* local time zone of ϕ_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_Φ:

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

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

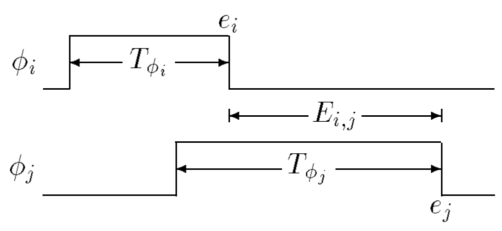

Figure 7: *The phase shift operator provides the relative difference between times in the local time zones of different phases.*

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

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

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 $w_r(e) = w(e) + r(u) - r(v)$, 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 $r(v_h) = 0$.

The positive edge weight constraints prevent 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 [10, 11]. 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 [15] 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 ϕ_i is referred to as *phase* i of the clock Φ. All ϕ_i have a common cycle time T_Φ. Each phase divides the clock cycle into two intervals as shown in Figure 6: An *active* interval of duration T_{ϕ_i} and a *passive* interval of duration $(T_\Phi - T_{\phi_i})$. The latches controlled by a clock

[2] In some applications negative edge weights can be useful as an intermediate step [12].

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 retime 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 $u \overset{p}{\twoheadrightarrow} v$ and $u \overset{q}{\twoheadrightarrow} v$. 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 [10, 11] defines a 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:

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

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

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

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

1.1 Overview of the Paper

We first review the work of Leiserson et. al. [10, 11] 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 [15]. 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. [10, 11] and extend it to handle level-sensitive latches. We then review the basic retiming results of their paper. The reader is encouraged to read [10, 11] for full details.

A circuit is modeled as a directed multigraph $G = \langle V, E, w, d, s \rangle$ 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 $u \overset{p}{\twoheadrightarrow} v$. 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 $l \overset{p}{\twoheadrightarrow} m$.

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 $l \overset{p}{\twoheadrightarrow} m$ 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 $w(l \overset{p}{\twoheadrightarrow} l)$ 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.

replacing each register with a pair of ϕ_1, ϕ_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 10.345. 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:* In 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 not 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.

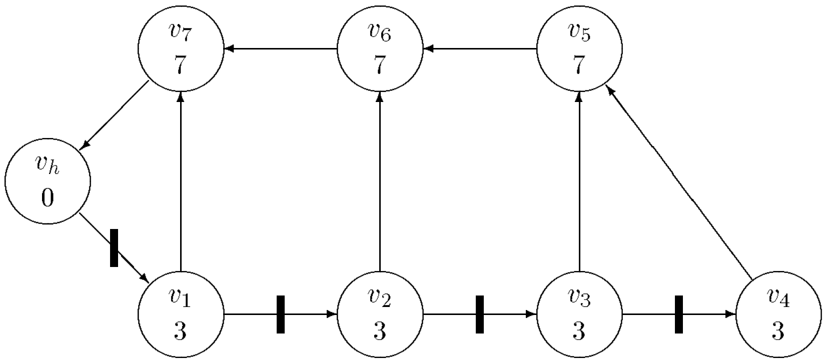

Figure 3: *The correlator circuit from Leiserson et. al. in its initial configuration with registers shown as solid bars.*

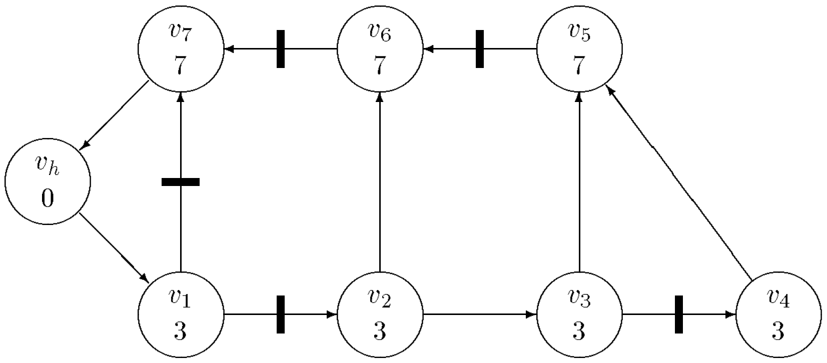

Figure 4: *The edge-clocked correlator circuit optimally retimed to a clock period* $T_\Phi = 13$.

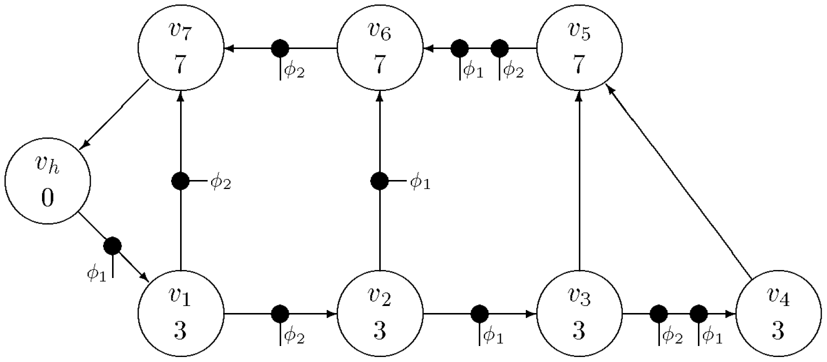

Figure 5: *The correlator circuit optimally retimed using a two-equal-phase clock. Latches are represented by solid circles and marked with controlling clock phase. The resulting clock cycle time is 10 units.*

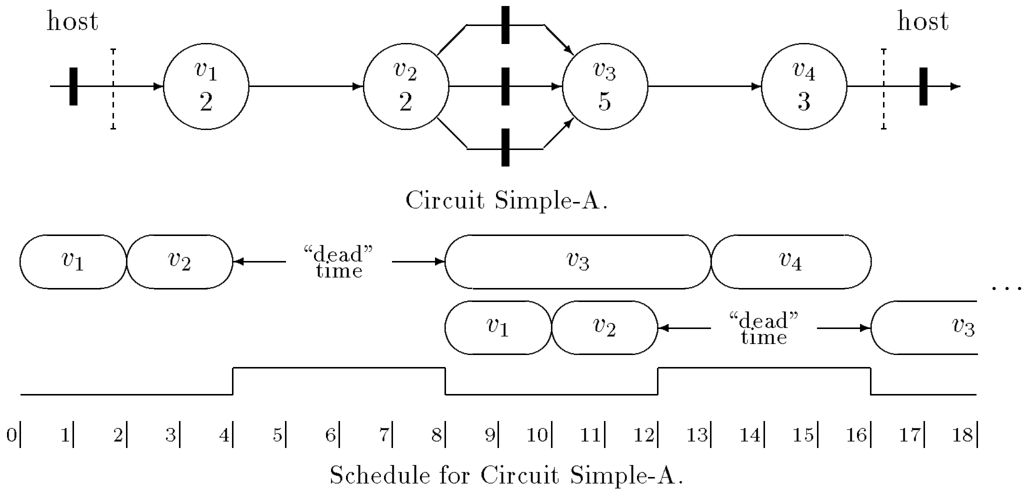

Figure 1: *A simple circuit optimally timed using edge-triggered registers and the resulting clock schedule.*

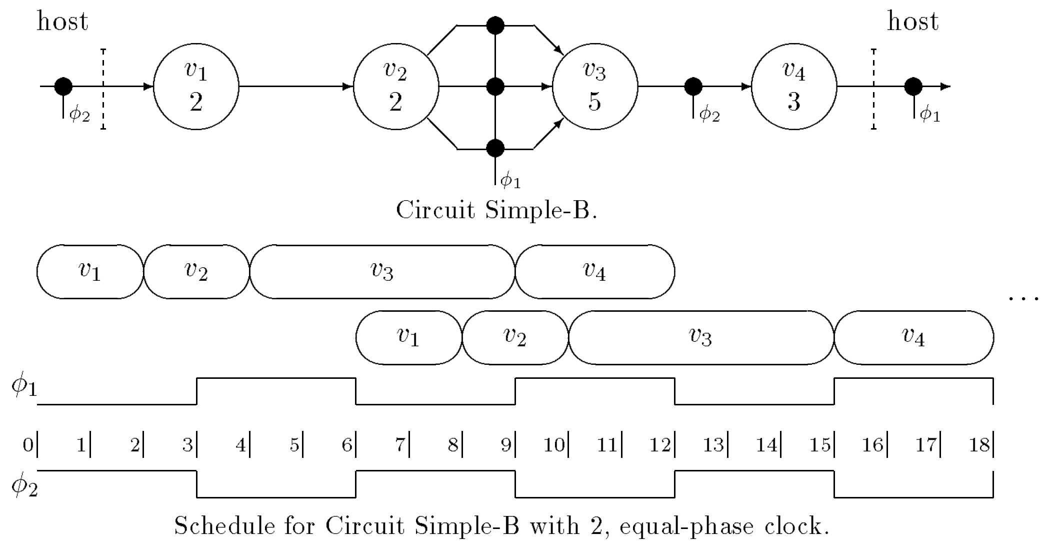

Figure 2: *The simple circuit now timed using level-sensitive latches.*

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, if the circuit in Figure 2 is to be retimed with $T_\Phi = 7$, the latches on the edges 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

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 [10, 11].

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 ϕ_1, ϕ_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

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.