Tim: A Timing Package for Two-Phase, Level-Clocked Circuitry

Marios C. Papaefthymiou Keith H. Randall
MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139
{marios,randall}@lcs.mit.edu

Abstract

Tim is a versatile and efficient tool for verifying and optimizing the timing of two-phase, level-clocked circuitry. Tim performs a variety of functions, such as timing verification, clock tuning, retiming for maximum speed of operation, retiming for minimum number of latches, and sensitivity analysis.

In this paper, we present new polynomial-time optimization algorithms for retiming and sensitivity analysis, and we describe the implementation of the new and previously reported algorithms in Tim. We also present empirical results from the application of Tim to sequential circuitry obtained from academic and industrial sources. Our experiments show that the number of latches in edge-triggered designs which have been retimed for maximum performance can be substantially reduced in corresponding two-phase, level-clocked designs that operate at the same speed.

1 Introduction

We have developed a new timing package called Tim for circuits that employ a nonoverlapping two-phase clocking scheme. Tim's features include novel optimizations for retiming and sensitivity analysis as well as more conventional operations such as timing verification and optimal tuning of clocking schemes. In this paper, we present new algorithms for retiming to minimize the number of latches and for sensitivity analysis in two-phase circuits. We also describe the implementation of Tim and present experimental results from its application on sequential circuits. These results show that two-phase clocking can maintain the performance of edge-triggered clocking while achieving substantial reductions in latch count, and therefore circuit area.

Several tools have been developed for analyzing the timing of circuits that contain level-sensitive latches [1, 3, 5, 9, 16, 20, 22]. These tools perform timing verification and enable the user to minimize the overall clock period by tuning various parameters of the clocking schemes. Our tool provides the designer with two additional features: retiming and sensitivity analysis.

Retiming relocates latches in a circuit while preserving its functionality. This transformation is employed in Tim with the objective of reducing the clock period or the number of latches in the original circuit. In both conceptual and computational terms, retiming is more difficult for level-clocked circuits than for edge-triggered circuits. Algorithms for retiming single-phase and multiphase level-clocked circuits have appeared in [21] and [8, 13], respectively. Retiming heuristics were given in [2]. We present an algorithm for retiming level-clocked circuits with a view to minimize latch count.

We describe new algorithms for sensitivity analysis that provide information about the critical paths in a circuit and about the sensitivity of a circuit's clock period to changes in its gate delays. This feature identifies the maximum increase in gate delays that does not affect the circuit's clock period. It also identifies the minimum decrease in gate delays that is necessary to remove gates from the critical paths.

All of Tim's algorithms are provably correct and run in polynomial time. The entire package has been written using the C programming language and integrated into the SIS tools from Berkeley.

Tim has been applied to sequential circuits obtained from academic and industrial sources. Our experiments show that the number of latches in these edge-triggered designs when retimed for maximum performance can be considerably reduced in corresponding level-clocked designs while maintaining performance. This reduction results in circuits with smaller area without compromising performance.

The remainder of the paper is organized as follows. Section 2 gives an overview of the system. Section 3 presents the circuit model, and Section 4 describes the algorithms implemented in Tim. In Section 5 we focus on specific implementation issues, and in Section 6 we present experimental results from the application of our timing package. Section 7 describes work in progress.

2 System overview

Tim provides the following classes of operations for circuits that employ nonoverlapping two-phase clocking schemes:

Timing verification: Given a two-phase circuit and a clocking scheme, Tim verifies whether the circuit is properly timed.

Clock tuning: Given a two-phase circuit, Tim computes a nonoverlapping two-phase clocking scheme so that the given circuit operates at maximum speed.

Retiming for speed: Given a two-phase circuit and a clocking scheme, Tim computes a retiming of the given circuit that is properly timed by the given clocking scheme. When the circuit is not bound to a specific clocking scheme, Tim can perform retiming in conjunction with simultaneous clock tuning, so that the resulting circuit operates at maximum speed. Tim is able to perform retiming and clock tuning optimally both for clocking schemes with fixed duty-ratio and for clocking schemes with unrestricted duty-ratio.

Retiming for minimum latch count: Given a two-phase circuit and a symmetric clocking scheme, Tim can compute a retimed circuit with minimum latch count that is properly timed by the given clocking scheme.

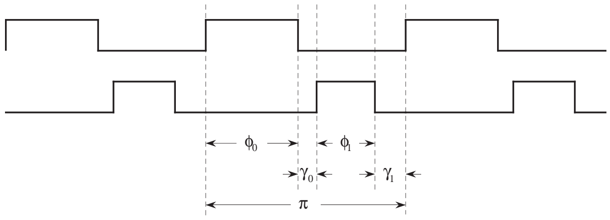
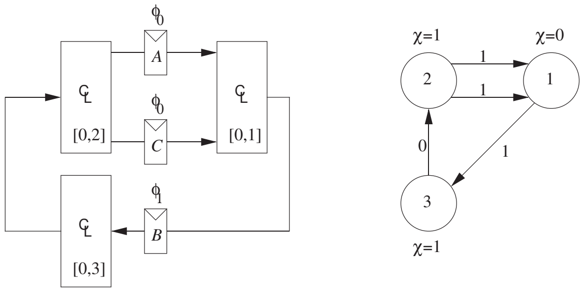


Figure 1: A nonoverlapping two-phase clocking scheme.



(a) (b)

Figure 2: Part (a) shows a two-phase circuit that consists of three blocks of combinational logic and three latches denoted by the letters A, B, and C. The minimum and the maximum propagation delays of the signals through the combinational logic are given in brackets for each block of logic. For simplicity, in this figure latches are assumed to have zero propagation delay. Part (b) illustrates the graph model of the circuit.

Sensitivity analysis: Given a two-phase circuit that is properly timed by a given clocking scheme, TIM can identify for each gate that is not on a critical path the maximum increase in its delay that will not affect proper timing of the circuit by the given clocking scheme. For each gate on a critical path, TIM can identify the minimum decrease in its delay that will take the gate off of the critical path.

3 Circuit model

TIM manipulates circuits that employ a nonoverlapping two-phase clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ with clock period $\pi = \phi_0 + \gamma_0 + \phi_1 + \gamma_1$. Figure 1 illustrates such a clocking scheme. The duty-cycles of the two phases are denoted by ϕ_0 and ϕ_1. The gaps γ_0 and γ_1 between the two phases handle engineering considerations such as setup and hold times of the latches, clock skew, and nonideal clocking waveforms.

The circuit model employed by TIM is based on the one described in [8]. A two-phase, level-clocked circuit is modeled as a vertex-weighted, edge-weighted graph $G = \langle V, E, d, w, \chi \rangle$, as shown in Figure 2. The vertices in V represent blocks of combinational logic, and the edges in E represent wires. The nonnegative, real vertex-weight $d(v)$ denotes the maximum propagation delay of the signals through the block represented by v. The minimum propagation delay (contamination delay) of every block is assumed to be zero. The nonnegative, integer edge-weight $w(e)$ denotes the number of latches on the wire represented by e. Each latch is clocked by one of the two phases of π. Whenever the clock input of a latch is asserted, the latch becomes transparent and data ripple through. Along any path in G, latches are clocked on alternate phases, and around any directed cycle in G there are at least two latches. We have extended the model in [8] to include non-ideal latches: All latches in TIM are assumed to have equal propagation delays, equal setup times and equal hold times. The function $\chi : V \to \{0, 1\}$ denotes for each vertex v the phase that clocks all latches that can reach v along a purely combinational path.

4 Algorithms

In this section we describe the algorithms implemented in TIM.

Timing verification

A general algorithm for timing verification was given in [7] and runs in $O(E^2)$ steps in the special case of two-phase circuits. We implemented in TIM an asymptotically faster algorithm from [8] that runs in $O(VE)$ steps and was designed specifically for two-phase circuits.

TIM asserts that a two-phase circuit is properly timed by a clocking scheme if at any time during the circuit's operation the latches hold the same values as they would if all propagation delays were zero. This intuitive description is quantified using the notion of *rise-to-fall* times along the paths in a circuit: For any path p from a latch A to a latch B, the rise-to-fall time $\tau(p)$ is the time it would take for the effect of a rising edge of A's phase to propagate and be stored by the falling edge of B's phase if all propagation delays in the circuit were zero. For example, the rise-to-fall time of the path $A \rightsquigarrow C$ in Figure 2 is $\phi_0 + \gamma_0 + \phi_1 + \gamma_1 + \phi_0 = \phi_0 + \pi$. In [8] it is shown that a two-phase circuit is properly timed if and only if the following two conditions are satisfied:

PC. For every simple path p in the circuit with delay $d(p)$ and rise-to-fall time $\tau(p)$, we must have $d(p) \leq \tau(p)$.

CC. For every simple cycle c in the circuit with delay $d(c)$ and $w(c)$ latches, we must have $d(c) \leq \pi w(c)/2$.

Intuitively, the first condition guarantees that signals have enough time to propagate along every *simple* path in the circuit. The second condition guarantees that signals have enough time to cycle around every *simple* closed path in the circuit. The two conditions combined guarantee that signals have enough time to propagate along *any* path in the circuit.

The timing verification algorithm in TIM operates as follows. First, from each wire $e \in E$ it eliminates latches in pairs until $w(e) \leq 3$. This elimination does not cause any violation of the timing constraints PC and CC, if the original circuit is properly timed, because a pair of latches operates as a flip-flop. The second phase of the algorithm computes a value $D(v, i)$ for each vertex $v \in V$ and for $i = 0, 1, \ldots, 3|V| - 3$, according to the following recursion:

$$D(v, i) = d(v) + \max\{D(u, i - w(e)) : u \xrightarrow{e} v \text{ and } i \geq w(e)\}.$$

Each $D(v, i)$ is the longest propagation delay among all paths in the circuit that end at v and have i latches. In order to compute the $D(v, i)$'s, the algorithm performs $3|V| - 2$ passes over the vertex-set V of G. Each pass over V considers each edge in E once, and it is performed in $O(E)$ steps. The third and final phase of the algorithm compares each of the $O(V^2)$ values $D(v, i)$ with the rise-to-fall time that corresponds to the particular v and i. At the same time, the algorithm uses

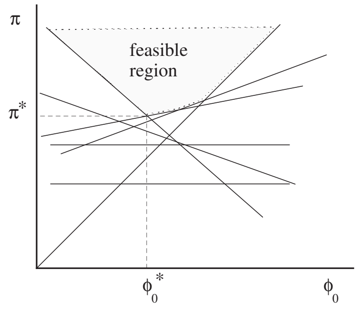


Figure 3: Clock tuning as a two-dimensional linear programming problem. The lines with negative slope are constraints on paths between latches that are clocked on ϕ_0. The lines with positive slope are constraints on paths between latches that are clocked on ϕ_1. The horizontal lines express constraints on paths between latches that are clocked on alternate phases and constraints on the simple cycles of the circuit.

the values $D(v,i)$ to check the strongest among the simple cycle constraints by extending Karp's $O(VE)$-time minimum cycle-mean algorithm to solve the problem of finding a cycle c with maximum $d(c)/w(c)$ ratio [10].

Clock tuning

When tuning the clock of a given two-phase circuit, TIM computes duty-cycles ϕ_0 and ϕ_1 such that the period of the resulting clocking scheme is minimized. The gaps γ_0 and γ_1 are supplied as inputs and remain fixed.

TIM handles clock tuning as a two-dimensional linear program in the variables ϕ_0 and π with the objective to minimize π. The rise-to-fall times in the timing constraints PC and CC are functions of ϕ_0, ϕ_1, and π. By setting $\phi_1 = \pi - \phi_0 - \gamma_0 - \gamma_1$, the constraints are viewed as half-planes in the (ϕ_0, π)-plane, as shown in Figure 3. A pair (ϕ_0^*, π^*) that minimizes the clock period π is an intersection point of these lines.

Clock tuning is performed in TIM as follows. First, from the $O(V^2)$ values $D(v,i)$ the tool generates $O(V)$ path constraints by selecting $\max_{v \in V} D(v,i)$ for each $i = 0, 1, \ldots, 3|V| - 3$. Subsequently, it computes the $O(V^2)$ intersections among these constraints and the cycle constraints CC, and it returns the pair (ϕ_0^*, π^*) with the shortest feasible period. The $D(v,i)$'s and the cycle constraints are computed in $O(VE)$ time, the $O(V)$ constraints are computed in $O(V^2)$ time, and the $O(V^2)$ intersections are computed in $O(V^2)$ time. Thus, clock tuning is performed in $O(VE)$ steps.

Retiming for speed

When retiming for speed, TIM relocates latches to minimize the length of the critical paths in the circuit. Retiming in TIM does not relocate I/O latches. In order to retime a circuit G, each vertex $v \in V$ is assigned an integer $r(v)$. The retimed circuit is obtained by removing $r(v)$ latches from each output wire of v and adding $r(v)$ latches to each input wire of v, as shown in Figure 4. In this section, we describe a *constraint graph* G_c that captures all constraints that must be satisfied so that the retimed circuit is properly timed by a given clocking scheme. We then describe how TIM uses G_c to compute a retiming.

The constraint graph $G_c = \langle V, E_c, w_c \rangle$ is an edge-weighted

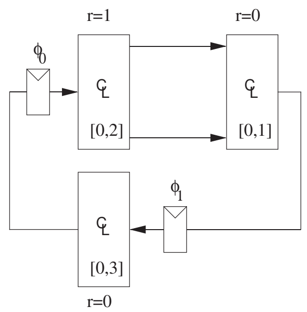


Figure 4: A retiming of the original circuit in Figure 2. Note that retiming changes the delay along the combinational paths and the number of latches in the circuit.

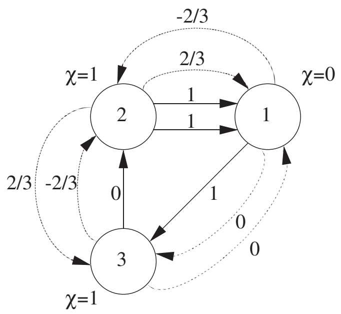


Figure 5: The constraint graph $G_c = \langle V, E_c, w_c \rangle$ that corresponds to the circuit graph in Figure 2(b) for $\pi = 6$. The solid lines are the edges in the edge-set E of the original circuit graph G, and the dotted lines are the edges in $E_c - E$ which are computed by an all-pairs shortest-paths computation.

graph that is obtained from the circuit graph G for a given clock period π. The edge-set E_c includes E as well as an edge between every pair of vertices $u, v \in V$. For $e \in E$, the edge-weight $w_c(e)$ equals $w(e)$. For $u \xrightarrow{e} v \in E_c - E$, the edge-weight $w_c(e)$ equals the length of the shortest-path from u to v in G with edge-cost $w(e') - 2d(j)/\pi$ for each edge $i \xrightarrow{e'} j \in G$. The constraint graph that corresponds to the circuit graph of Figure 2 is shown in Figure 5.

TIM computes a retiming that achieves a given clocking scheme as follows. First, it constructs the constraint graph G_c by performing an all-pairs shortest-paths computation. Then it solves a set of constraints of the form $g(r(u)) - f(r(v)) \leq w_c(e)$ for every pair of vertices $u, v \in V$ that is connected by an edge $e \in E_c$. The functions f, g are monotonically increasing and account for the changes in the rise-to-fall times along the paths $u \overset{p}{\leadsto} v$ in G due to the relocation of its latches. The functions f and g and an $O(V^3)$-time relaxation algorithm for solving the ensuing constraints are described in more detail in [8]. The constraints graph is computed in $O(VE \lg V)$ steps, and thus, the algorithm terminates in $O(V^3 + VE \lg V)$ steps.

In order to find a retiming that achieves the minimum possible clock period for clocking schemes with a given duty-ratio, TIM performs a binary search in the clock period π until the desired precision is achieved. When the duty-cycle of the clocking scheme is not fixed, TIM performs a binary search in π and a linear search in one of the two duty-cycles. More details on these approximation schemes as well as exact polynomial-time algorithms, are given in [8].

Retiming for minimum latch count

TIM can retime a given circuit in order to achieve a specified symmetric clocking scheme $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$ with the minimum number of latches. This problem is reduced to the dual of an uncapacitated minimum-cost flow problem on the constraint graph G_c [19]. The demand (supply) of each vertex v equals the difference between the outdegree and the indegree of v in the circuit graph G. For each edge $e \in E$, the cost equals $w_c(e)$. For each edge $u \xrightarrow{e} v \in E_c - E$, the cost equals $\lfloor w_c(e) + 2 - 2(d(u) + \gamma)/\pi \rfloor$. The construction of G_c is based on a modified version of the original circuit graph G so that retiming takes into account maximal sharing of latches. The constraint graph G_c is also augmented with constraints which guarantee that retiming will not relocate I/O latches. TIM computes a minimum-cost flow on G_c in $O(V^3)$ time by applying Orlin's scaling algorithm [15]. The dual variables of the minimum-cost flow are used to retime the circuit.

Sensitivity analysis

In addition to optimizations that manipulate the circuit directly, TIM provides timing information about the circuit at a higher level. This information identifies parts of the circuit that have potential for further optimization.

The current version of TIM performs two kinds of sensitivity analysis. The first kind is concerned with gates in a circuit that are not on a critical path for a given clocking scheme. Specifically, TIM solves the following problem.

Sensitivity analysis I : Let $G = \langle V, E, d, w, \chi \rangle$ be a two-phase level-clocked circuit, and let $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ be a clocking scheme such that G is properly timed by π. Let $u \in V$ be a gate that is not on a critical path in G. We wish to compute the maximum $\delta \geq 0$, such that for $d(u) \leftarrow d(u) + \delta$ the circuit G is still properly timed by π.

Based on the path constraints PC and the cycle constraints CC, this problem is solved by considering the paths with the longest propagation delay that go through u and have i latches for $i = 0, 1, \ldots, 3|V| - 3$, and the cycle c through u that minimizes $\pi w(c)/2 - d(c)$. The new value of $d(u)$ is the maximum value $d(u)$ can take without violating these constraints. For a single gate $u \in V$, these constraints are computed in $O(VE)$ steps in a way similar to the computation of the $D(v, i)$'s in the timing verification algorithm. For all gates $u \in V$, these constraints are computed in $O(VE + V^2 \lg V)$ steps by solving an all-pairs shortest-paths problem on G with edge-cost $\lfloor w(e)/2 \rfloor \pi + (w(e) \bmod 2) \left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) - d(v)$ for every edge $u \xrightarrow{e} v \in E$.

The second kind of sensitivity analysis is concerned with gates that lie on critical paths for a given clocking scheme. A gate lies on a critical path if and only if some constraint PC or some constraint CC that involves this gate is satisfied with equality. For each such gate, TIM solves the following problem.

Sensitivity analysis II : Let $G = \langle V, E, d, w, \chi \rangle$ be a two-phase level-clocked circuit, and let $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ be a clocking scheme such that G is properly timed by π. Let $u \in V$ be a gate on a critical path of G. We wish to compute the minimum $\delta \geq 0$, such that for $d(u) \leftarrow d(u) - \delta$ the gate u is not on a critical path any more.

If there exist critical paths in G that do not go through u, then the answer is straightforward: an arbitrarily small $\delta > 0$ will do. If all critical paths go through u, however, we set $d(u) = 0$, and we compute the new critical paths and optimal clocking scheme. If u still lies on some critical path, then no δ can take u off of the critical path. If u does not lie on any critical path, the new value of $d(u)$ equals the maximum value that $d(u)$ can take without violating the path and cycle constraints that go through u, with respect to the new clocking scheme. The computation of these constraints for a single gate $u \in V$ is performed in $O(VE)$ steps total.

5 Implementation issues

In this section we discuss our implementation of the shortest-paths algorithms and the data structures that we used in TIM.

Shortest-paths computations are repeatedly employed in almost all functions of TIM. We have implemented two single-source shortest-paths algorithms. The first is an algorithm by Bellman and Ford that runs in $O(VE)$ time and solves the shortest-paths problem on graphs with real edge-weights [4]. We use this algorithm before all-pairs shortest-paths computations in order to compute graphs with identical shortest-paths and nonnegative edge-weights, because shortest-paths can be computed more efficiently on such graphs.

The second shortest-paths algorithm that we have implemented is Dijkstra's algorithm for graphs with nonnegative edge-weights [4]. The running time of this algorithm depends on the implementation of the priority queue it employs. We tried three priority queues. The first one was a simple array, in which case the theoretical running time of the algorithm is $O(V^2)$. We used this implementation in the beginning until we debugged our programs. The array was soon a limiting factor in the performance of our tool, and so we implemented two additional data structures: a Fibonacci heap [6] that yields an $O(E + V \lg V)$ running time, and a binary heap that yields an $O(E \lg V)$ running time. Our binary heap implementation was faster than the Fibonacci heap implementation, most likely because circuit graphs are sparse and the overhead of a Fibonacci heap is too high.

6 Experimental results

In this section, we present the results of TIM's application to circuits obtained from academic and industrial sources. These results show that clocking with symmetric phases can often be as fast as clocking with tuned clocking schemes. They also show that two-phase level-sensitive clocking can lead to substantial savings in the number of latches in a circuit as compared to edge-triggered clocking, without any performance degradation.

Our collection of two-phase circuits was limited. In contrast, we had access to several circuits that were designed entirely using edge-triggered flip-flops. These circuits were MCNC benchmark circuits, AT&T communication circuits and custom circuitry designed for MIT's Alewife machine. We generated two-phase circuits by replacing each edge-triggered flip-flop by a pair of level-sensitive latches that are clocked on alternate phases of a two-phase clocking scheme. In fact, it is a common practice in VLSI to implement edge-triggered flip-flops by two back-to-back level-sensitive latches [26]. The statistics of our original example circuits are shown in Figure 6.

Circuit	i/p	o/p	latches	gates	clock period
mult16a	17	1	32	147	66.987
s344	9	11	30	160	28.579
s349	9	11	30	161	28.579
s382	3	6	42	158	28.962
s386	7	7	12	159	22.999
s400	3	6	42	162	19.272
s510	19	7	12	211	20.369
s641	35	23	38	379	95.464
s820	19	19	10	289	31.993
ampseq1	15	2	72	135	17.041
ampseq3	4	70	144	213	12.041
ampseq4	21	32	64	216	23.087
DRAM-ctl	19	22	32	339	8.794

Figure 6: Statistics of our example circuits. The table shows the number of primary inputs, the number of primary outputs, the number of latches, where each edge-triggered flip-flop is counted as two level-sensitive latches, the number of gates, and the original clock period.

Using TIM we performed the following three sets of experiments. In the first set of experiments, we retimed each circuit for maximum speed with symmetric clocking schemes. In the second set of experiments, we retimed each circuit for maximum speed with unrestricted duty-ratio. Retiming with unrestricted duty-ratio produced circuits that were no more than 1.5% faster than the circuits generated by retiming with symmetric clocking schemes. A more detailed description of these experiments can be found in [19].

In the third set of experiments, we applied the following procedure for each example circuit. We retimed the original edge-triggered circuit to achieve a specific clock period with the minimum number of flip-flops. We also retimed the corresponding two-phase, level-clocked circuit to achieve the same clock period with minimum latch count, and then we compared the latch counts in the two optimal designs. Our experiment shows that two-phase clocking can lead to substantial savings in the number of storage elements in aggressive edge-triggered designs. The table of Figure 7 gives the minimum latch count for each edge-triggered circuit and its corresponding two-phase circuit when both circuits operate at the minimum clock period that can be achieved by retiming the original edge-triggered circuit. In this case, two-phase clocking leads to savings up to 38%. For half of the circuits reductions of greater than 18% are achieved. The results in the table of Figure 8 are different, however. This table gives the minimum latch count for each edge-triggered circuit and its corresponding two-phase circuit when both circuits operate at the clock period of the original edge-triggered circuit. In this case, two-phase clocking does not save any storage elements over edge-triggering, except in one case.

We ran TIM on a Sun SPARCstation IPX with 16MB of main memory. For the circuit DRAM-ctl, timing verification, clock tuning, and sensitivity analysis required approximately 1 minute. Retiming for speed with fixed duty-ratio required approximately 3 minutes. Retiming for speed with unrestricted duty-ratio required approximately 5 minutes. Retiming for minimum latch count required approximately 20 minutes.

Circuit	clock period	latches (e-t)	latches (l-c)	reduction (%)
mult16a	15.465	62	51	18
s344	20.946	46	34	26
s349	20.945	46	34	26
s382	11.567	68	42	38
s386	22.999	12	12	0
s400	11.759	64	42	34
s510	18.740	16	14	12
s641	95.464	38	38	0
s820	31.993	10	10	0
ampseq1	11.773	86	74	14
ampseq3	9.700	174	152	13
ampseq4	15.004	98	76	22
DRAM-ctl	8.692	52	32	38

Figure 7: Reduction in latch count achieved by two-phase clocking over edge-triggering. In this table, each edge-triggered circuit and its corresponding two-phase circuit operate at the minimum clock period achievable by retiming the original edge-triggered circuit. For each circuit, the table gives its operating period, the minimum number of latches in the edge-triggered circuit after retiming, the minimum number of latches in the two-phase, level-clocked circuit after retiming, and the reduction in the number of latches achieved by two-phase clocking over edge-triggering. The number of latches in each edge-triggered circuit is twice its number of flip-flops.

Circuit	clock period	latches (e-t)	latches (l-c)
mult16a	66.987	32	32
s344	28.579	30	30
s349	28.579	30	30
s382	28.962	32	32
s386	22.999	12	12
s400	19.272	32	32
s510	20.369	12	12
s641	95.464	38	38
s820	31.993	10	10
ampseq1	17.041	70	70
ampseq3	12.041	144	141
ampseq4	23.087	64	64
DRAM-ctl	8.794	32	32

Figure 8: When each edge-triggered circuit and its corresponding two-phase circuit are clocked at the original clock period specification of the edge-triggered circuit the latch count decreases only for ampseq3. For each circuit, the table gives its operating period, the minimum number of latches in the edge-triggered circuit after retiming, and the minimum number of latches in the two-phase, level-clocked circuit after retiming. The number of latches in each edge-triggered circuit is twice its number of flip-flops.

7 Work in progress

We are currently addressing the limitations of our timing package. We are extending TIM to handle precharged gates, overlapping phases, nonzero contamination delays, and gated clocks. Handling gated clocks seems to be the most challenging among these extensions. We are also trying to lift the restriction that latches must have equal propagation delays, equal setup times, and equal hold times. This restriction is not necessary for the algorithms that perform timing verification, clock tuning, and sensitivity analysis. The proof for the running time of the retiming algorithms, however, relies on it.

We are also working to improve the efficiency of TIM's current implementation. We are investigating ways to reduce the number of edges in the constraint graph G_c by eliminating in a systematic way constraints that are subsumed by others. Based on our experimental evidence that retiming with symmetric clocking schemes often leads to designs that operate just as fast as designs obtained by retiming with unrestricted duty-ratio, we are implementing an asymptotically faster algorithm for retiming with symmetric clocking schemes that runs in $O(VE + V^2 \lg V)$ time [8]. We are also working on faster implementations of the minimum-cost flow algorithm and the shortest-paths algorithms.

Acknowledgments

We would like to thank Charles Leiserson of MIT for his suggestions and ideas during the course of this work. Thanks also to Alex Ishii of NEC for many fruitful discussions and pointers to edge-triggered circuits. Srinivas Devadas of MIT provided us with edge-triggered circuits and comments on drafts of this paper. John Kubiatowicz of MIT provided us with circuitry from MIT's Alewife machine. Marios Papaefthymiou was supported in part by the Defense Advanced Research Projects Agency under Grant N00014-91-J-1698. Keith Randall was supported in part through the Undergraduate Research Opportunities Program at MIT.

References

[1] V. D. Agrawal. Synchronous path analysis in MOS circuit simulator. In *Proc. 19th Design Automation Conference*, pages 629–635, June 1982.

[2] T. Burks, K. Sakallah, and T. Mudge. Multiphase retiming using $minT_c$. *τ92 ACM Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, March 1992.

[3] J. J. Cherry. Pearl: a CMOS timing analyzer. In *Proc. 25th ACM/IEEE Design Automation Conference*, pages 148–153, June 1988.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, MIT Press, 1990.

[5] M. R. Dagenais and N. C. Rumin. Automatic determination of optimal clocking parameters in synchronous MOS VLSI circuits. In *Advanced Research in VLSI: Proc. of the 5th MIT Conference*, pages 19–33, 1988.

[6] M. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization problems. *Proc. of the 25th Annual Symposium on Foundations of Computer Science*, pages 338–346, October 1984.

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

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

[9] N. P. Jouppi. Timing analysis for NMOS VLSI. In *Proc. 20th Design Automation Conference*, pages 411–418, June 1983.

[10] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[11] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing synchronous circuitry by retiming. *3rd Caltech Conference on VLSI*, 1983. R. Bryant, ed., pp. 87-116.

[12] C. E. Leiserson and J. B. Saxe. Optimizing synchronous systems. *Journal of VLSI and Computer Systems*, 1(1):41–67, 1983.

[13] B. Lockyear and C. Ebeling. Optimal retiming of multi-phase, level-clocked circuits. In *Advanced Research in VLSI and Parallel Systems: Proc. of the 1992 Brown/MIT Conference*. MIT Press, March 1992.

[14] N. Megiddo. Partitioning with two lines in the plane. *Journal of Algorithms*, 6:430 – 433, 1985.

[15] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Proc. of the 20th Annual ACM Symposium on Theory of Computing*, pages 377–387, May 1988.

[16] J. K. Ousterhout. Switch-level timing verifier for digital MOS VLSI. *IEEE Trans. Computer-Aided Design*, CAD-4:336–349, July 1985.

[17] M. C. Papaefthymiou. Sensitivity analysis of synchronous circuitry. Unpublished manuscript, August 1992.

[18] M. C. Papaefthymiou and K. H. Randall. TIM: an interactive timing optimization tool for level-clocked circuits. July 1992. User's Guide and Reference Manual, under preparation.

[19] M. C. Papaefthymiou and K. H. Randall. Edge-triggering vs. two-phase level-clocking. In *Research on Integrated Systems: Proceedings of the 1993 Symposium*, March 1993.

[20] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun. $checkT_c$ and $minT_c$: Timing verification and optimal clocking of synchronous digital circuits. In *Digest of Technical Papers of the 1990 IEEE International Conference on CAD*, pages 552–555, November 1990.

[21] N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Retiming of circuits with single phase level-sensitive latches. In *International Conference on Computer Design*, October 1991.

[22] T. G. Szymanski. LEADOUT: A static timing analyzer for MOS circuits. In *Digest of Technical Papers of the 1986 IEEE International Conference on CAD*, pages 130–133, November 1986.

[23] T. G. Szymanski. Computing optimal clock schedules. In *Proc. 29th ACM/IEEE Design Automation Conference*, pages 399–404, June 1992.

[24] T. G. Szymanski and N. Shenoy. Verifying clock schedules. In *Digest of Technical Papers of the 1992 IEEE/ACM International Conference on CAD*, November 1992.

[25] S. H. Unger and C. J. Tan. Clocking schemes for high speed digital systems. *IEEE Transactions on Computers*, C-35(10):880–895, October 1986.

[26] S. A. Ward and R. H. Halstead, Jr. *Computation Structures*. McGraw-Hill, MIT Press, 1990.