An Experimental Comparison of Edge-Triggering and Level-Clocking

by

Keith H. Randall

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1993

Author . . . .
Department of Electrical Engineering and Computer Science
May 4, 1993

Certified by . . . .
Charles E. Leiserson
Thesis Supervisor

Accepted by . . . .
Leonard A. Gould
Chairman, Departmental Committee on Undergraduate Theses

An Experimental Comparison of Edge-Triggering and Level-Clocking

by

Keith H. Randall

Submitted to the Department of Electrical Engineering and Computer Science
on May 4, 1993, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Computer Science and Engineering

Abstract

Circuits implemented with two-phase level-clocked latches have the theoretical potential to operate faster and require less state than equivalent circuits implemented with edge-triggered latches. We investigated to what extent one can achieve this theoretical potential with real circuits. We found that level-clocked circuits are no faster than edge-triggered circuits except when the delay between any two latches is approximately equal to the maximum gate delay. On the other hand, level-clocked circuits can often be implemented with significantly less state than equivalent edge-triggered circuits clocked at the same speed. Over one-third of the circuits tested had a reduction in state of at least 25 percent.

These tests were performed in TIM, a computer-aided design tool for verification and optimization of two-phase, level-clocked circuitry. TIM consists of several efficient polynomial-time algorithms that can check circuit timing and modify circuit layout in order to meet various timing criteria. TIM was implemented in C on top of the widely available SIS circuit design environment so that TIM can operate on circuits in many formats.

TIM is an implementation of a wide variety of algorithms taken from the literature on circuit optimization. Two novel algorithms were developed for TIM that improve upon existing algorithms. One is a $\Theta(V)$-space version of Karp's minimum mean-weight cycle algorithm, which improves upon the previous space bound of $\Theta(V^2)$. The other is a latch minimization algorithm which uses Orlin's minimum-cost flow algorithm and latch sharing to achieve a $O(\log V)$ reduction in running time over the conventional latch minimization algorithm.

Some of this work represents joint research with Marios Papaefthymiou.

Thesis Supervisor: Charles E. Leiserson

Title: Professor of Computer Science and Engineering

Contents

Chapter 1

Introduction

1.1 Edge-Triggering and Level-Clocking

Level-clocking is becoming an increasingly popular alternative to edge-triggering as a clocking methodology for high-performance designs. In level-clocked circuitry, clocked storage elements are implemented using level-sensitive latches, which become transparent and transmit their inputs unimpeded to their outputs when their clock signals are asserted. Proponents of level-clocking argue that level-clocked circuitry can provide more flexibility in meeting a specific clock period and that it has the theoretical potential to operate faster than the more conventional edge-triggered circuitry. These arguments are based on the fact that in level-clocked circuitry, the delay of a gate can be split among two or more clock periods, in contrast to edge-triggered circuitry in which the delay of a gate must be assigned to a single clock period. Advocates of edge-triggering, on the other hand, present simplicity and implementation ease as major advantages of edge-triggering, since edge-triggered flip-flops directly support the abstraction of a storage element that is synchronized by the ticking of a clock. They also refer to the existence of powerful design tools as another major incentive for designing circuitry with edge-triggered flip-flops.

These arguments in support of edge-triggering and level-clocking are either theoretical or nonquantifiable. We wanted to make an empirical study of the two clocking methodologies. For this purpose, we have run experiments that compare edge-triggered implementations of synchronous circuitry and corresponding level-clocked implementations that employ a two-phase, nonoverlapping clocking scheme. Our empirical comparison focused on two specific quantitative measures: speed and number of storage elements. We ran our tests using Tim, a timing optimization tool currently under development at MIT [14].

1.2 Theoretical Results

The first question to ask is why edge-triggered and level-clocked circuits should have different speed and latch count properties at all. Delay in a circuit is caused by propagation delay in the circuit's combinational logic, not in its storage elements. Then why should the choice of storage elements have a bearing on overall circuit speed and latch count?

The reason storage elements matter is because they determine how chains of combinational logic are combined to determine the clock period. In edge-triggered circuits, the largest combinational delay between any two latches in the circuit determines the clock period. For instance, in the edge-triggered circuit in figure 1-1 the path from logic element A to logic element B determines the minimum clock period because it is one of the largest combinational paths in the circuit. For other placements of storage elements, different combinational paths would determine the minimum clock period. Thus, we see that placement of storage elements is a key factor in determining the speed of a circuit.

Level-clocked latches are more flexible than edge-triggered latches because they can split up combinational paths in a circuit in ways that edge-triggered latches cannot. For instance, in the circuit in figure 1-1 the minimum clock period for any placement of the edge-triggered latches (with the same degree of pipelining) is 2. By switching to level-clocked latches, we can decrease the minimum clock period to 1.5 while keeping the latch count constant, as shown in figure 1-2. This demonstration shows that the choice of storage element may have a large impact on the speed and latch count of certain circuits. We investigated how much of this potential advantage

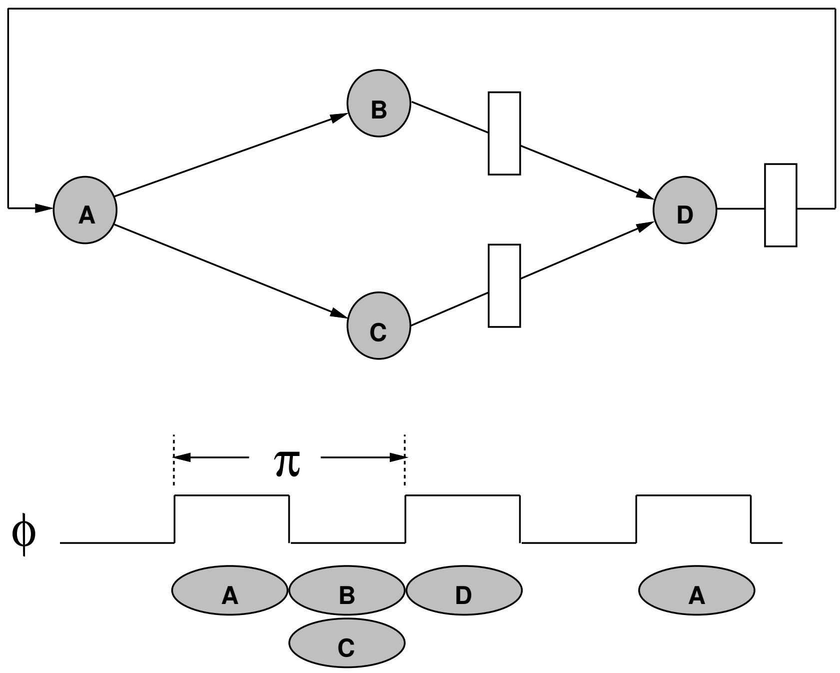

Figure 1-1: A sample edge-triggered circuit. The circles represent combinational logic elements of unit propagation delay. The rectangles represent edge-triggered latches clocked by a common global clock. The lines represent wires connecting these elements together.

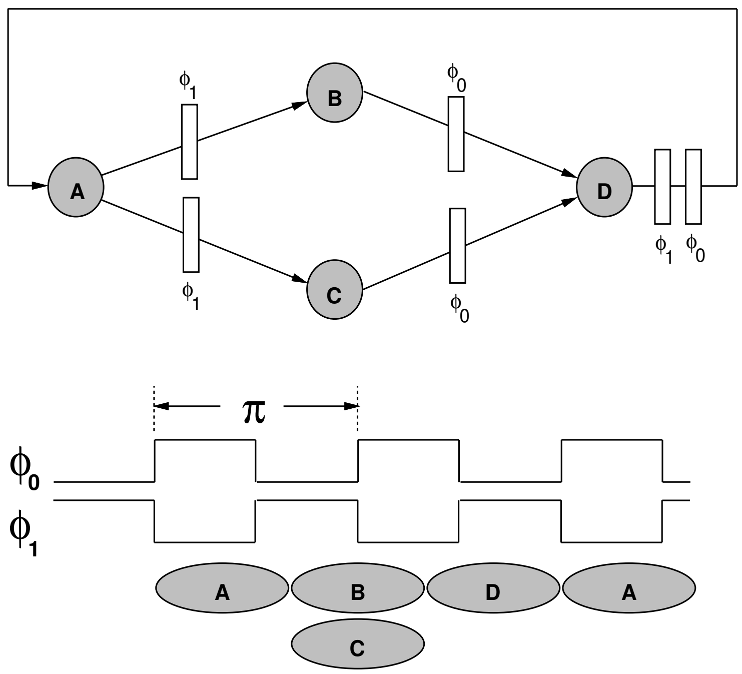

Figure 1-2: A sample level-clocked circuit. This circuit is equivalent to the circuit in the previous figure but its clock period is shorter.

of level-clocked circuits can actually be realized with real circuits.

1.3 Synopsis of Results

Our speedup experiments show that edge-triggered circuitry often operates just as fast as two-phase circuitry, despite the theoretical advantage of two-phase clocking, and that the speed potential of two-phase clocking is generally obtained only when the combinational delay between any two consecutive latches is roughly uniform and close to the maximum gate delay. Our experiments also show that two-phase clocking leads to greater speedups when all gate delays in the circuit are roughly equal. Our experimentation with clock tuning suggests that asymmetric clocking schemes provide little or no speedup over optimal clocking schemes.

With respect to the number of storage elements, however, our experiments demonstrate that two-phase clocking can lead to substantial latch-count reductions in aggressive edge-triggered designs that operate at maximum speed. For two of our test circuits, we obtained reductions of 38%; for more than one third of our circuits, we obtained reductions of 25%; and for more than half of our circuits, reductions exceeded 18%. For low-performance designs that operate below their speed potential, however, our experiments show that two-phase clocking does not reduce the number of storage elements.

1.4 Methodology of Comparison

How can edge-triggering and two-phase clocking be fairly compared on an empirical basis? First, a fair experiment should compare competing circuit implementations that have the same functionality. It would be meaningless to compare two circuits that compute different functions. Second, a fair experiment should compare competing circuit implementations based solely on differences due to their storage elements. It would be unfair to compare two circuits which differ in their combinational elements, for example, because such a comparison would not depend only on the clocking methodology employed.

We did not have the resources to embark on designing pairs of competing circuits for various applications. We settled, therefore, on the strategy of taking edge-triggered circuits and using our timing tool TIM to produce equivalent two-phase circuits. We could have done the reverse, converting two-phase circuits to edge-triggered ones, but since edge-triggered designs are more popular, we were able to obtain several interesting edge-triggered circuits.

We produced a two-phase circuit from an edge-triggered one by following a two-step procedure. The first step of this procedure was to replace each edge-triggered flip-flop by a pair of back-to-back level-sensitive latches that are clocked by a two-phase, nonoverlapping clocking scheme, as shown in figure 1-3. (In fact, it is common in VLSI to implement edge-triggered flip-flops by a pair of back-to-back level-sensitive latches [3, 20].) The two-phase circuit produced by this conversion has the same clock period and the same number of storage elements as the original edge-triggered circuit, under the reasonable assumption that each edge-triggered flip-flop counts as two level-sensitive latches. Moreover, the placement of its latches is dictated by the original edge-triggered design, and the potential of two-phase clocking due to alternate placements of the latches in the circuit is not revealed. Thus, we needed a method to relocate storage elements and explore the space of possible placements in the circuit without changing its functionality or its I/O specification.

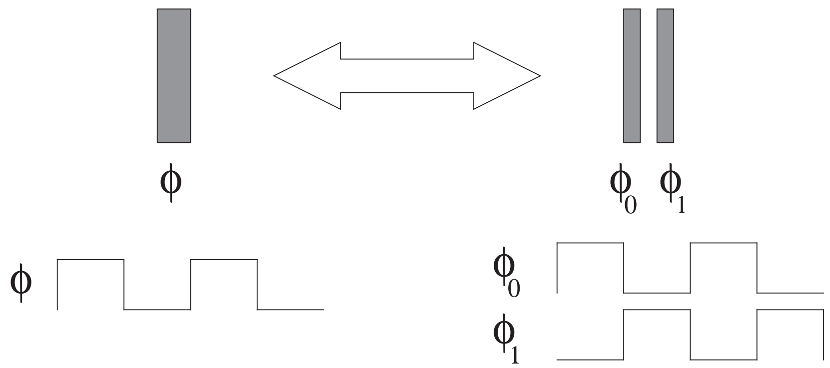

Figure 1-3: Replacement of an edge-triggered flip-flop by a pair of level-sensitive latches. The flip-flop is clocked by a single clock ϕ, and the two level-sensitive latches are clocked on the phases ϕ_0, ϕ_1 of a two-phase, nonoverlapping clocking scheme.

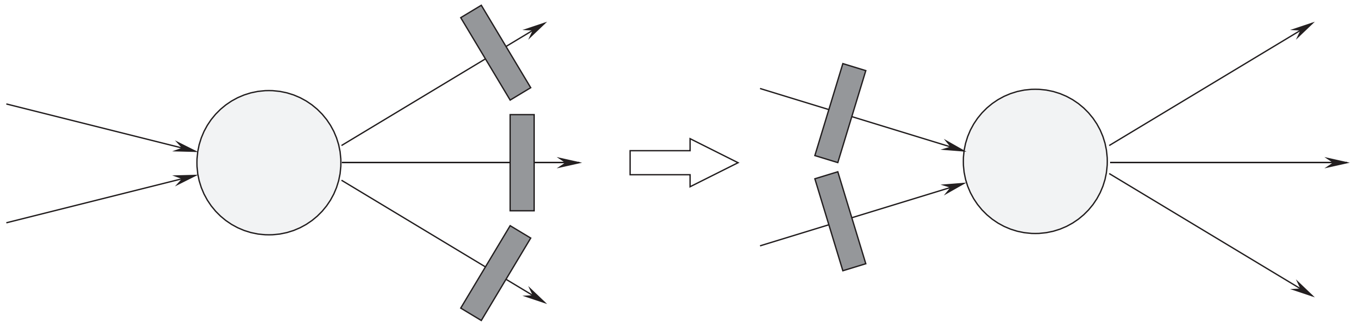

Figure 1-4: This figure illustrates retiming of a gate with lag 1. In this case, one storage element is removed from each output wire of the gate, and one storage element is inserted at each input wire of the gate. The total number of storage elements is reduced by 1. The critical paths in the circuit may also change as a result of the relocation of the storage elements.

The second step of the procedure was to use the "retiming" transformation to relocate the storage elements of the two-phase circuit that resulted from the first step. Retiming relocates storage elements in both edge-triggered and level-clocked circuitry without changing its functionality [7, 8, 10]. In addition, retiming is a "universal" transformation for speeding up circuits, in the sense that any other functionality-preserving transformation that did better than retiming would depend on the functionality of the gates in the circuit [8]. Figure 1-4 illustrates the retiming operation for a gate in a circuit. Observe that retiming can change the clock period as well as the number of storage elements in a circuit. We insured, however, that the retiming we used to relocate latches did not alter the input and output timing constraints of the original circuit.

Our experimental procedure compared an optimal edge-triggered implementation that we obtained from an original edge-triggered circuit with an optimal two-phase implementation of the corresponding two-phase circuit. The use of an optimal edge-triggered implementation as a reference point was essential to ensure that we did not penalize edge-triggering due to suboptimalities in the original edge-triggered circuit that depended on the placement of the storage elements by the circuit designer and were not intrinsic to edge-triggering.

We performed two kinds of experiments. The first kind of experiments compared edge-triggered and two-phase circuits with respect to speed. Our basic experimental approach was the following. First, we retimed a given edge-triggered implementation for maximum speed. Then, by using retiming in conjunction with tuning of the clocking schemes, we obtained the fastest possible implementation of the corresponding two-phase circuit, and we compared the speed of the two optimal implementations. The second kind of experiments compared edge-triggered and two-phase circuits with respect to their number of storage elements when operating at some specified clock period. We first retimed a given edge-triggered implementation without changing its I/O specification, in order to achieve the specified clock period with the minimum number of flip-flops. We then retimed the corresponding two-phase circuit without changing its I/O specification, in order to achieve the same clock period with the minimum number of latches. We compared the number of storage elements in the two optimal implementations, under the reasonable assumption that each edge-triggered flip-flop is counted as two level-sensitive latches.

1.5 Implementation of TIM

TIM is a tool designed and implemented to run the tests described above. In particular, TIM can perform tasks such as converting edge-triggered circuits to level-clocked circuits, verifying correct operation of edge-triggered and level-clocked circuits, and performing retimings of these circuits. Also, TIM has the ability to optimize clocking schemes and retimings to achieve fast clock periods and small latch counts. TIM uses polynomial-time algorithms from the literature to perform these tasks, and as part of this thesis we have developed two new algorithms specifically to improve the running time of TIM.

TIM also interfaces with SIS, a widely available circuit design environment from Berkeley. SIS performs operations such as circuit format conversion, technology mapping, and delay modeling. With this interface, circuit designers can use TIM to optimize their circuits along with other operations they might want to perform. This interface also makes TIM easier to write and use because TIM can work on a variety of circuit formats and delay models with no additional effort.

1.6 Previous Work

Timing verification and optimization of synchronous circuitry has been the subject of extensive study [1, 4, 5, 6, 7, 10, 11, 15, 16, 18, 19]. In [9], Leiserson et al. present a good summary of algorithms for retiming edge-triggered circuits for speed and latch count. The concept of replacing each edge-triggered flip-flop by a pair of back-to-back level-sensitive latches, and then using retiming for speed optimization has been explored in [1, 5, 11]. The potential of level-clocking for reducing the number of storage elements has been mentioned in [11]. The idea of using latches instead of flip-flops has been also used in [17] in the context of multi-phase clocks. Retiming for speed has been studied in the context of single-phase level-clocked circuits in [16]. Despite the large amount of work in this area, our contribution is (we believe) the first attempt to quantify empirically the performance differences of edge-triggering and two-phase clocking.

1.7 Outline of Thesis

The remainder of this thesis has four sections. Chapter 2 describes our experimental methodology and reports our results on the relative speed of the two implementation approaches. In Chapter 3 we present our experimental results on latch-minimization. In Chapter 4 we examine the tools used to perform these experiments and some issues in their implementation, and Chapter 5 presents conclusions and further research questions.

Chapter 2

Speedup experiments

In this chapter, we present our investigation of edge-triggering and two-phase clocking with respect to speed. First, we briefly refer to our tools and test circuits. We move on to describe and motivate our experimental methodology, and then we discuss our results.

Our experiments were performed using TIM, a timing optimization package currently under development at MIT [14]. TIM performs a variety of functions on two-phase circuitry, such as timing verification, retiming, clock tuning, and sensitivity analysis. It also performs timing verification, retiming, and sensitivity analysis on edge-triggered circuitry. TIM has been implemented in C, and most of its functions have been integrated in the SIS tools from Berkeley. Our test circuits were MCNC benchmark circuits and AT&T communication circuits, all of which were originally designed with edge-triggered flip-flops. The largest among these circuits had 290 gates, and the current version of TIM performed retiming and simultaneous tuning on that circuit in less than 2 minutes on a 16MB SPARCstation.

2.1 Experimental procedure

In our speedup experiments we employed the following three optimizations:

OP1 Retiming of edge-triggered circuitry for maximum speed of operation (minimum clock period).

OP2 Retiming of two-phase circuitry for maximum speed of operation with a symmetric clocking scheme.

OP3 Retiming and simultaneous clock tuning of two-phase circuitry for maximum speed of operation.

Using these three optimizations, we initially performed experiments *SP1* and *SP2*.

SP1 We compared the speed of each original edge-triggered circuit that was optimized using *OP1* with the speed of the corresponding two-phase circuit that was optimized using *OP2*.

SP2 We compared the speed of each original edge-triggered circuit that was optimized using *OP1* with the speed of the corresponding two-phase circuit that was optimized using *OP3*.

The goal of our experimentation was not only to investigate whether two-phase clocking could speed-up the particular edge-triggered circuits in our test suite. We also wanted to determine specific design characteristics that may lead to faster two-phase circuits. To that effect, we performed experiments *SP3* and *SP4* on altered versions of our original test circuits that were obtained by modifying them in two ways.

The first modification changed the number of storage elements in the original circuits by pipelining, a transformation that increases the latency of a computation without decreasing its throughput. A degree-P pipelining of each circuit was obtained by multiplying the number of storage elements on each wire of the circuit by the integer P. The purpose of this transformation was to investigate which of the two implementation approaches is favored more when the number of storage elements is increased.

The second modification changed the gate delays of the original circuits to test whether more uniform gate delays led to higher speedups. For each circuit G, we created four additional circuits G_i for $i = 0.8, 0.6, 0.4, 0.2$. Each G_i was topologically identical to G, but its gate delays d_i were modified. For each circuit G_i, each gate delay $d_i(v)$ was set equal to $d^i(v)$, where $d(v)$ was the delay assigned to v using the library corresponding to the circuit G. Thus, for smaller values of the exponent i, the gate delays in the circuits G_i became increasingly uniform. The objective of this modification was to see how uniformity of gate delays affects the speed of the two implementations.

Using the three optimizations on the modified circuits we performed experiments *SP3* and *SP4*.

SP3 On each circuit G_i for $i = 1.0, 0.8, 0.6, 0.4, 0.2$, we applied the following procedure for $P = 1, 2, \ldots, 6$. We optimized the edge-triggered circuit using *OP1*, and we compared its speed with its corresponding two-phase circuit that was optimized using *OP2*.

SP4 On each circuit G_i for $i = 1.0, 0.8, 0.6, 0.4, 0.2$, we applied the following procedure for $P = 1, 2, \ldots, 6$. We optimized the edge-triggered circuit using *OP1*, and we compared its speed with its corresponding two-phase circuit that was optimized using *OP3*.

Note that for $i = 1.0$ and $P = 1$, experiments *SP3* and *SP4* were identical to experiments *SP1* and *SP2*, respectively.

2.2 Experimental results

Remarkably, our initial experiments *SP1* and *SP2* indicated that two-phase clocking was no better than edge-triggering for any of our test circuits. The application of the three optimizations *OP1*, *OP2*, and *OP3* on the original circuits, with gate delays assigned by their corresponding libraries, showed no speedup by switching to two-phase clocking. Although this result was surprising and unexpected, it could not have been a mere coincidence. Our subsequent empirical investigation with experiments *SP3* and *SP4* led us to the conclusion that there are two important circuit characteristics that determine the relative speed of the two implementation approaches: the maximum gate delay $d_{\max}$ and the *critical ratio* R, which is defined as the maximum ratio of total delay over total number of storage elements around the cycles in the edge-triggered circuit.

Our experimental results for *SP3* are illustrated in the plots of figure 2-1. Each plot gives data for an original test circuit $G_{1.0}$ and its four delay-modified versions G_i for $i = 0.8, 0.6, 0.4, 0.2$. For each of the five delay configurations of a test circuit and for degrees of pipelining up to 6, each plot gives two numbers: the speedup f_{lc}/f_{et} achieved by two-phase clocking over edge-triggering, and the ratio $d_{\max}/R$, where $d_{\max}$ is the maximum gate delay and R is the critical ratio of the circuit. For each circuit G_i, the value of the ratio $d_{\max}/R$ closest to 1 is boldfaced.

As is apparent from the graphs, for almost every delay configuration, the maximum speedup is achieved when the ratio $d_{\max}/R$ is closest to 1. In the five configurations of `mult16a`, for example, as the ratio $d_{\max}/R$ increases from small values and approaches 1 from below, the speedup constantly increases. When the ratio exceeds 1, the speedup soon drops down to 1. A similar pattern is revealed for almost all of our test circuits. This phenomenon can be justified as follows. The critical ratio R is a lower bound on the clock period of both the edge-triggered and the level-clocked circuit [5, 13]. Consequently, the longest combinational delay in the circuits is at least R under any transformation that does not change the number of storage elements around the cycles in the circuit. Retiming distributes the storage elements, however, so that combinational path delays are roughly equal across the circuit and close to the critical ratio R. When R becomes comparable to the maximum gate delay $d_{\max}$, then the longest combinational delay also tends to approach $d_{\max}$, and then the potential of two-phase clocking becomes apparent. Intuitively, when R approaches $d_{\max}$, level-clocking evens out differences among path delays more effectively than edge-triggering by letting the computations ripple through the transparent latches.

Let us examine more closely some characteristic graphs in figure 2-1. Our initially surprising results from experiments *SP1* and *SP2* can be explained by looking at the ratios $d_{\max}/R$ of the original circuits, which correspond to $P = 1$ and $i = 1.0$. For every such circuit, the ratio $d_{\max}/R$ is smaller than 0.67. The only exceptions are `mult16b`, `ampseq2`, and `ampseq1`. `mult16b` has a ratio greater than 1, and conse-

Figure 2-1: Results of experiment *SP3* on the MCNC benchmark and the AT&T circuits. Each plot corresponds to a test circuit. The first row of the horizontal axis gives the pipelining degree P. Each of the next five rows corresponds to a circuit G_i for $i = 0.2, 0.4, 0.6, 0.8, 1.0$, and it gives the ratio $d_{\max}/R$ for each pipelining degree. In each row, the ratio $d_{\max}/R$ closest to 1 is boldfaced. The vertical axis gives the speedup f_{lc}/f_{et} obtained for a specific i and a specific P. The clock frequency f_{et} was obtained by applying *OP1*. The clock frequency f_{lc} was obtained by applying *OP2*. For almost every test circuit, maximum speedups are achieved when the ratio $d_{\max}/R$ is closest to 1, or equivalently, when R is closest to $d_{\max}$. Greater peak speedups are achieved as we move from $G_{1.0}$ to $G_{0.2}$, that is, as the gate delays become roughly equal across the entire circuit. The results of experiment *SP4* have no significant differences from the results in this figure.

quently, it is already heavily pipelined. For higher degrees of pipelining, `mult16b` does not become any faster, which leads us to the conclusion that the original design of `mult16b` takes full advantage of any existing speed potential. The situation with `ampseq2` is similar. The original design has already no margin for improvement, and for higher degrees of pipelining there are sufficiently many storage elements for edge-triggering to be as fast as two-phase clocking. The situation with `ampseq1` is somewhat different. The ratio $d_{\max}/R$ is close to 1, but there is still room for improvement, since without any pipelining, that is, for $P = 1$, all versions of `ampseq1` become faster by level-clocking.

Another conclusion that we can draw from the plots in figure 2-1 is that two-phase clocking leads to greater speedups when the gate delays are more uniform. For every test circuit, peak speedups increase as the exponent i decreases, that is, as the gate delays become more uniform. This observation suggests that standard-cell designs in which gate delays are roughly equal are likely to benefit from two-phase clocking.

The data shown in figure 2-1 are the results of experiment *SP3*, in which the two-phase circuits were clocked by symmetric clocking schemes. We also performed experiment *SP4* that combines retiming with tuning of the clocking schemes. In all cases, however, *OP3* did not provide any speedup greater than 2% over *OP2*. Thus, our experiments suggest that clocking with asymmetrical schemes often does not provide any speed advantage over symmetric schemes.

Chapter 3

Latch-minimization experiments

In this chapter we present our experimental comparison of edge-triggering and two-phase clocking in terms of the number of storage elements required by each implementation approach. We first describe our methodology, and then we present and discuss our experimental results.

3.1 Experimental procedure

In our experiments, we employed retiming in order to minimize the number of storage elements in the circuits. We retimed both the original edge-triggered circuits and their corresponding two-phase circuits in order to achieve a given clock period with the minimum number of storage elements. In both cases, the retiming transformation was applied without relocating the I/O storage elements of the circuits, and thus the I/O specification remained unchanged.

We compared the two implementations of each circuit by performing experiments *LM1* and *LM2*.

LM1 We retimed the original edge-triggered circuit in order to achieve the minimum period possible with the minimum number of flip-flops. Then, we retimed the corresponding two-phase circuit in order to achieve the same period with the minimum number of latches, and we compared the number of storage elements in the two circuits.

LM2 We retimed the original edge-triggered circuit, in order to achieve its original clock period specification with the minimum number of flip-flops. Then, we retimed the corresponding two-phase circuit, in order to achieve the same period with the minimum number of latches, and we compared the number of storage elements in the two circuits.

The motivation behind these two experiments was to investigate the impact of two-phase clocking on the number of storage elements under different conditions of operation. Experiment *LM1* was aimed at the typical situation, where speed is the primary concern, and edge-triggered circuits are configured to operate at the maximum of their potential. It is often the case, however, that the clock period is dictated by external system considerations and cannot be changed easily. To that effect, we also performed experiment *LM2*, which compares the number of storage elements in the two implementations when the clock period equals that of the original edge-triggered circuit.

We performed our experiments using TIM. The latch-minimization algorithms in TIM run in polynomial time and take into account maximal sharing of storage elements. We ran our tests on MCNC benchmark circuits, AT&T communication circuits and custom circuitry designed for MIT's Alewife machine. The largest among these circuits had 340 gates, and the current version of TIM retimed that circuit to minimize its number of latches in less than 20 minutes on a 16MB SPARCstation.

3.2 Experimental results

Our experimental results for the two sets of experiments are shown in figure 3-1 and 3-2. There is a striking difference between these two sets of results. When the operating period is the minimum clock period that can be achieved by retiming the original edge-triggered circuit, then two-phase clocking leads to substantial reductions in the number of storage elements. When the operating period is that specified for the original circuit, however, there are almost no gains in the number of storage elements when we switch from edge-triggering to two-phase clocking.

Circuit	clock period	e-t latch count	l-c latch count	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 3-1: Results of experiment *LM1*. In this experiment, each edge-triggered circuit and its corresponding two-phase circuit operate at the minimum clock period that can be achieved 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 implementation after retiming, the minimum number of latches in the two-phase implementation after retiming, and the reduction in the number of storage elements with respect to edge-triggering. The number of latches in each edge-triggered circuit is twice its number of flip-flops.

In the experimental results of figure 3-1 the greatest reductions were achieved for two controller circuits. The number of latches in both `s382` and the DRAM controller `DRAM-ctl` of the Alewife machine was reduced by 38%. Substantial reductions were also achieved for the multiplier circuits `mult16a`, `s344`, `s349`, for the controller circuit `s400`, as well as for the communication circuits `ampseq1`, `ampseq3`, and `ampseq4`. The two circuits `s641` and `s820` for which the number of storage elements did not decrease by two-phase clocking were PLD's.

Figure 3-2 shows that for all circuits except `ampseq3`, there was no reduction in the number of storage elements when the circuits were operating at the clock period specification of the original circuit. This seemingly negative result can be explained by comparing the clock periods of the original and the optimally retimed designs. In most cases, the original circuits operate substantially slower than the optimally retimed circuits. Most notably, the original `mult16a` is almost four times slower than its minimum-period retimed version. When the original clock period specification is

Circuit	clock period	e-t latch count	l-c latch count
`mult16a`	66.987	32	32
`s344`	28.579	30	30
`s349`	28.579	30	30
`s382`	18.962	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 3-2: Results of experiment *LM2*. In this experiment, each edge-triggered circuit and its corresponding two-phase circuit are clocked at the original clock period specification of the edge-triggered circuit. For each circuit, the table gives its operating period, the minimum number of latches in the edge-triggered implementation after retiming, and the minimum number of latches in the two-phase implementation after retiming. The number of latches in each edge-triggered circuit is twice its number of flip-flops. Note that the level-clocked latch count decreases only for `ampseq3`.

so far from the minimum achievable, the placement of the storage elements in the edge-triggered circuit is as flexible as in the two-phase implementation, and thus no additional reductions are achieved by two-phase clocking. In the optimally retimed edge-triggered circuits, however, the minimum number of storage elements increases substantially, as it can be verified by comparing the columns that give the edge-triggered latch counts in figure 3-1 and 3-2. Two-phase clocking can decrease this number without degrading circuit performance. In fact, as it is evident from the columns that give the level-clocked latch counts in figure 3-1 and 3-2, the number of latches in more than half of the aggressive level-clocked implementations is not more than 15% higher than the number of latches in the low-performance implementations.

Experiments *LM1* and *LM2* explore the two extremes of possible clock periods for a circuit. We would also like to explore what happens to the latch count comparison at intermediate clock periods. In figure 3-3 we graph the minimum latch counts for circuit `s382` at a range of clock periods between the original clock period of 18.962 and the minimum clock period of 11.567. We notice that the closer the clock period is to the minimum clock period, the more latches are saved by level-clocking. We found that the same qualitative pattern happens for all circuits we tested in which level-clocking reqires less state than edge-triggering at the minimum clock period.

Figure 3-3: Minimum latch count as a function of clock period for the circuit `s382`. Circles mark the edge-triggered minimum and squares mark the level-clocked minimum.

Chapter 4

Implementation

4.1 Implementation of TIM

The tests described in the previous chapters were performed using a timing package called TIM developed as part of this thesis. TIM's purpose was to provide a unified platform from which a whole host of verification and optimization programs could be run. TIM interfaces with a circuit design tool called SIS, developed at Berkeley, which performs low-level circuit operations such as circuit format conversion, technology mapping, and delay modeling. Circuit designers who are familiar with the SIS tool will find it easy to call TIM operations just as they can other SIS operations. Furthermore, SIS makes it easy to import and export circuits in various formats, change technology mappings, and modify delay models according to what circuit designers will use their circuits for.

A list of TIM's functions can be found in table 4.1. TIM formulates all of the optimization problems in this table as graph problems. In order to do this, TIM must convert an edge-triggered or level-clocked circuit into a graph. This conversion is performed as follows. Combinational logic elements are represented as the vertices of a graph and connections between combinational logic elements are represented as the edges of a graph. Vertices have weights that represent their delay, and edges have weights that represent the number of latches on the wire between the two elements that the edge connects. For example, the circuit in figure 1-1 can be represented as

Command	Description
Tim-edge-verify	Inputs: e-t circuit G, clock period π. This command checks to make sure that the clock period π correctly clocks G.
Tim-level-verify	Inputs: l-c circuit G, clocking scheme $\pi = (\phi_0, \gamma_0, \phi_1, \gamma_1)$. This command checks to make sure that the clocking scheme π correctly clocks G.
Tim-edge-tune	Inputs: e-t circuit G. This command finds the fastest clock period π that correctly clocks G.
Tim-level-tune	Inputs: l-c circuit G, gaps γ_0 and γ_1, duty ratio ρ. This command finds the fastest clocking scheme $\pi = (\phi_0, \gamma_0, \phi_1, \gamma_1)$ that correctly clocks G. If ρ is given, the optimization will be subject to the condition $\phi_1 = \rho\phi_0$.
Tim-edge-retime	Inputs: e-t circuit G, clock period π. This command relocates edge-triggered latches in G in order to achieve the clock period π. If no π is given, it will find the minimum π possible.
Tim-level-retime	Inputs: l-c circuit G, clocking scheme $\pi = (\phi_0, \gamma_0, \phi_1, \gamma_1)$, duty ratio ρ. This command relocates level-clocked latches in G in order to achieve the clocking scheme specified. If no π is given, it will find the minimum π possible. If ρ is given, the optimization will be subject to the condition $\phi_1 = \rho\phi_0$.
Tim-edge-min	Inputs: e-t circuit G, clock period π. This command relocates edge-triggered latches in G in order to achieve the clock period π with the minimum number of latches.
Tim-level-min	Inputs: l-c circuit G, symmetric clocking scheme $\pi = (\phi_0, \gamma_0, \phi_0, \gamma_0)$. This command relocates level-clocked latches in G in order to achieve the clocking scheme π with the minimum number of latches.

Figure 4-1: A list of Tim's commands. Each command has a brief description of its functionality.

the graph in figure 4-2. This transformation is not information preserving because we lose all knowledge about what kind of function the combinational logic is computing. This loss is not important, however, because TIM performs optimizations that depend only on the total delay of combinational logic, not on the specific function that an element calculates.

TIM can then use standard graph algorithms, such as shortest path algorithms, to determine features of a circuit. All of the commands listed in figure 4.1 are implemented as a combination of various graph algorithms. The detailed algorithms for solving these optimization problems can be found in [10] and [5].

A real problem with TIM is the fact that many of its algorithms take a significant amount of time to run on large circuits. In view of this, much effort was put into improving the algorithms that TIM uses to verify and optimize circuits. We present here two improvements to existing algorithms that allow TIM to run faster and on larger circuits.

4.2 A Space Efficient Algorithm

This section presents a new, more space efficient algorithm for determining the minimum mean-weight cycle of a graph. This problem is important because it comes up as a subproblem in many of the verification and optimization algorithms in TIM. Especially for the verification algorithms, this problem represents a substantial portion

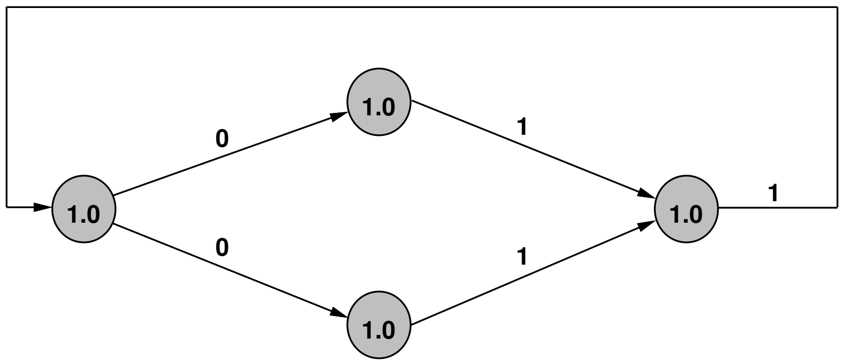

Figure 4-2: Conversion to a graph of the circuit in figure 1-1. Delays are written on nodes of the graph and latch counts are written next to edges of the graph.

of the computation and is thus a good target for optimization. We will describe why this problem comes up in TIM's algorithms, describe Karp's algorithm to solve it, and then give our new algorithm.

Given a graph $G = (V, E)$ with edge weights $w : e \to R$, we define the mean weight m_C of a cycle C as the ratio of the sum of the weights w around the cycle divided by the length of the cycle, i.e.

$$m_C = \frac{\sum_{e \in C} w(e)}{|C|} .$$

The minimum mean-weight cycle of a graph G is defined as the cycle C whose mean weight m_C is minimum over all cycles in the graph. In practice, we only need the mean weight of the minimum mean-weight cycle, which means that we do not need to explicitly identify the minimum cycle. We need only calculate

$$M = \min_{C \in \text{cycles}(G)} m_C ,$$

where M is the mean weight of the minimum mean-weight cycle. The value M is important in establishing a lower bound on the clock period of any circuit because around any cycle C in the graph, the total time it takes to propagate a value around the cycle is $\sum_{v \in C} d(v)$ and the total time the circuit has to propagate that value is $\pi \sum_{e \in C} w(e)$ in the edge-triggered case (or $\frac{\pi}{2} \sum_{e \in C} w(e)$ in the level-clocked case). Furthermore, this number is independent of any retiming of the circuit because the number of latches in any cycle is unaffected by retiming any node. Therefore, if we set $w'(e) = \pi w(e) - d(v)$ for edge-triggered latches (or $w'(e) = \frac{\pi}{2} w(e) - d(v)$ for level-clocked latches) then we must have $M \geq 0$ with respect to the w' weights for the circuit to function correctly.

Karp's minimum mean-weight cycle algorithm is a $\Theta(VE)$-time, $\Theta(V^2)$-space algorithm to compute M (see [2]). Although the time bound is a good one (or at least a reasonable one), the space required can be prohibitive for large circuits because memory-access times increase dramatically with the size of memory, especially for virtual memory systems where memory latency can be in the millisecond range. A better space bound for Karp's minimum mean-weight cycle algorithm translates to a faster running time because of this memory latency effect.

Karp's original algorithm is very space inefficient. With $\Theta(VE)$ time and $\Theta(V^2)$ space, each memory location is accessed on average only a constant number of times for sparse graphs. Consequently, a more space-efficient algorithm should be easy to develop. As part of this thesis, a new $\Theta(V)$ space version of Karp's algorithm was developed and used as part of TIM. In order to describe this new algorithm, we first look at Karp's minimum mean-weight cycle algorithm.

Karp's algorithm works as follows. First, a set of numbers $F(k,v)$ are computed, where $F(k,v)$ equals the minimum distance from a source vertex to v using k edges. These values can be computed using the recurrence relations

$$\begin{aligned} F(0,v) &= 0 \ , \\ F(k,v) &= \min_{u \in V} F(k-1,u) + w(u,v) \ . \end{aligned}$$

Then, we compute M as

$$M = \min_{v \in V} \max_{0 \le k \le n-1} \frac{F(n,v) - F(k,v)}{n-k} \ ,$$

where $n = |V|$.

Karp's algorithm works in two stages: first compute F, and then compute M. By interleaving these two stages, we can improve on the space bound. The $\Theta(V)$-space version of Karp's algorithm that we developed proceeds in three stages.

- Compute $F(n,v)$ for each v. Because we can compute $F(i,v)$ from $F(i-1,v)$, we need only keep two "rows" of F at any one time. Consequently, this calculation takes $\Theta(VE)$ time and only $\Theta(V)$ space.

- Compute
$$R(v) = \max_{0 \le k \le n-1} \frac{F(n,v) - F(k,v)}{n-k}$$
for each v. We can do this in $\Theta(V)$ space by recomputing all of the $F(k,v)$ just

as we did in the previous stage and update our maximum R for each k in turn.

- Compute $M = \min_{v \in V} R(v)$.

The total space required—two rows of F and the vector R—is $\Theta(V)$ space total. Furthermore, the constants are small, which makes this algorithm very practical. The only drawback is that $F(k, v)$ must be computed twice as compared to Karp's original algorithm which only computes $F(k, v)$ once. This cost is easily offset by the increased speed due to the small amount of space used.

4.3 Latch Sharing

The most time consuming function that TIM performs is latch minimization. We find that by implementing a latch sharing version of a typical latch minimization algorithm, we can save a $O(\log V)$ factor in the running time. We will describe the latch minimization problem and an algorithm to solve it, and then demonstrate that a $O(\log V)$ factor can be saved by implementing latch sharing.

The standard latch minimization problem is to find a retiming of the circuit graph that minimizes the number of latches in the circuit while still satisfying all timing constraints. A retiming is an assignment of *lags* to each vertex that represent the number of latches moved from output wires to input wires. Thus, the retimed weight (the resulting number of latches) of any edge is $w_r(u, v) = w(u, v) + r(u) - r(v)$, where r is the lag function. The number of latches in the circuit is then

$$\begin{aligned}
\sum_{e \in E} w_r(e) &= \sum_{e \in E} w(e) + r(v) - r(u) \\
&= \sum_{e \in E} w(e) + \sum_{e \in E} r(v) - r(u) \\
&= \sum_{e \in E} w(e) + \sum_{v \in V} (\text{indegree}(v) - \text{outdegree}(v)) r(v)
\end{aligned}$$

and the timing constraints (see [5]) are of the form

$$r(u) - r(v) \leq x(u, v) \; .$$

Thus, an equivalent statement of the latch minimization problem is to minimize $\sum_{v \in V} y(v)r(v)$ subject to the constraints $r(u) - r(v) \leq x(u, v)$. Expressed in this way, the problem is a linear programming problem which is the dual to a capacitated minimum-cost flow problem, with the capacities equal to the x variables and the demand/supply equal to the y variables. This problem can be solved in $O(V^3 \log V)$ time using Orlin's version of the Edmonds-Karp RHS scaling algorithm ([12]).

When circuit designers actually build circuits, however, the number of latches they use to build them is *not* the one computed above. The reason is because designers can employ a technique called *latch sharing*. This technique allows latches to be "shared" among several wires in the circuit. For example, the circuit graph in figure 4-3 has three latches in it, but by using one real latch to implement both latches leaving gate A we need only two latches to implement it.

Latch sharing implies that the real measure of latches required to implement a circuit should not be $\sum_{e \in E} w_r(e)$ but rather $\sum_{u \in V} \max_{e:u \to v} w_r(e)$, i.e. the number of latches needed to implement the fanout of any gate, no matter how large, is simply the largest latch count on any fanout wire.

Is this new minimization problem also dual to some capacitated minimum-cost flow problem? The answer is yes, and even more surprising, it is dual to a special case in which the demand/supply variables are all ± 1. This means that Orlin's RHS scaling algorithm will only take $O(V^3)$ time because the $\log V$ factor that comes from scaling the demand/supply variables can be eliminated.

The reduction from a minimization problem of the type with latch sharing to one without latch sharing can be found in [10]. Basically, the procedure is to associate with each gate in the graph a "mirror" gate which has no outputs, no delay, and receives one input from each of its pair's outputs (see figure 4-4). The weights on the edges into the mirror gate are n minus the weight on the corresponding edge out of the original gate, where $n = max_{e:u \to v} w(e)$. Finally, every edge out of a gate u and into its mirror u' is scaled with a factor of $1/\text{degree}(u)$. If we then run the normal latch minimization algorithm on this new circuit, we ensure three things about the resulting circuit:

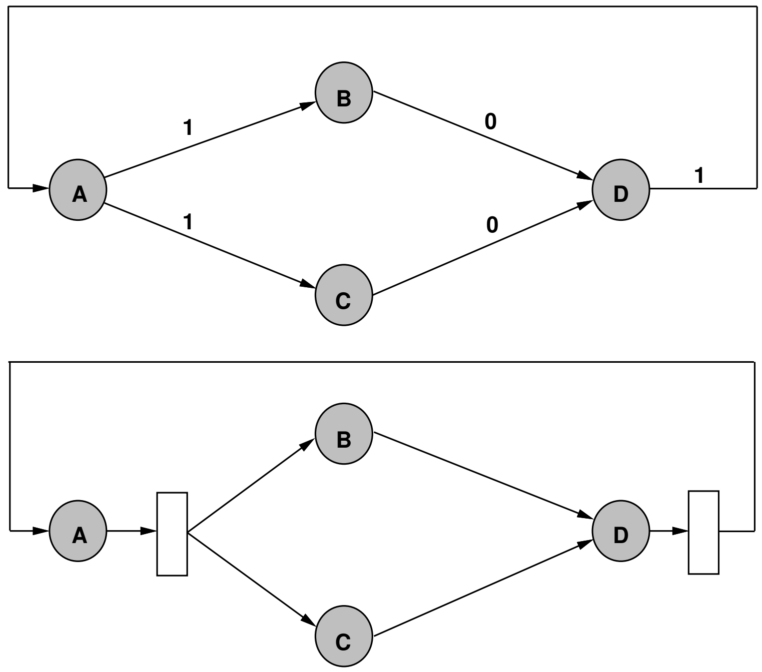

Figure 4-3: The circuit graph above which appears to require three latches to implement can in fact be implemented with only two latches as shown in the circuit below.

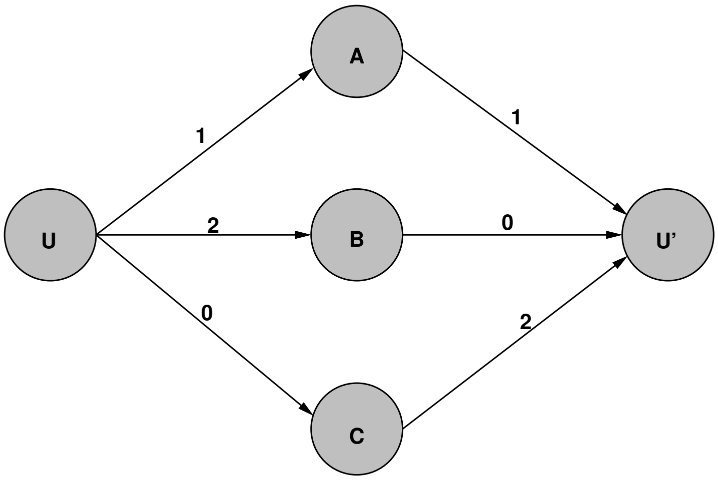

Figure 4-4: For every gate u in the circuit, a mirror gate u' is added and weighted edges are added from u's fanout gates to u' such that the weight along any path from u to u' is constant. Also, each edge is scaled by the fraction $1/\text{degree}(u)$ (not shown).

1. The total weight on a path from a gate u to its mirror gate u' is constant and equal to $n + r(u') - r(u)$, independent of the path chosen.

2. Since the mirror gate has no outputs, it will be retimed by the latch minimization algorithm until the weight on one of its input wires is zero. Therefore, the weight on one of u's output wires is $n + r(u') - r(u)$.

3. The total cost of this configuration is $n - r(u') - r(u)$ because there are degree(u) paths from u to u' of equal latch count and each one is scaled by a 1/degree(u) factor.

Since the latch minimization algorithm minimizes total cost, and total cost is equal to the maximum weight edge out of u, we have just minimized $\max_{e:u \to v} w_r(e)$ for each vertex u, which was our goal.

Also, notice that because we have scaled the edges leaving u, and any edge entering u has a new edge leaving u of the same weight, the value of indegree(v) − outdegree(v) is -1 for all real gates and $+1$ for all mirror gates. Thus the demand/supply variables in the dual problem are ± 1 and thus Orlin's RHS scaling algorithm runs in $O(V^3)$ time, saving a $O(\log V)$ factor over the traditional latch minimization techniques. This same savings will manifest itself in any min-cost flow algorithm that scales demand/supply variables because these variables are all of order unity. Furthermore, the latch minimization problem that is solved gives a more realistic count of the number of latches that a circuit designer would actually use to implement a given circuit.

Chapter 5

Conclusion

We compared edge-triggering and two-phase clocking in terms of speed and number of storage elements. Our methodology was independent of the functionality of the circuit and compared the two design approaches based on the effects of the storage elements in each one of them. In our speedup experiments, edge-triggering was often as fast as two-phase clocking, except when the average delay between any two consecutive latches was roughly uniform over the entire circuit and equal to the maximum gate delay, in which case the potential of two-phase clocking was generally obtained. Our experimental results suggest that circuits designed with standard cells of uniform delay benefit more from two-phase clocking. Moreover, symmetric clocking schemes seem to perform as well as tuned clocking schemes.

In terms of number of storage elements, two-phase clocking led to substantial reductions when the target clock period was set aggressively to the minimum that could be achieved by retiming the original edge-triggered circuit.

A software package called TIM was implemented to perform these experiments. It can perform verifications, speed optimizations, and state minimizations, all in polynomial time. TIM interfaced with SIS, a popular circuit design tool that allows TIM to inherit features such as circuit format conversion, technology mapping, and delay modeling. All of these features made TIM an easy-to-use, productive design tool for circuit designers.

Two new algorithms were discovered as part of the implementation of TIM. One was a $\Theta(V)$-space algorithm for the minimum mean-weight cycle problem, improving the previous space bound of $\Theta(V^2)$ from Karp's algorithm. The other was a method for saving $O(\log V)$ time in the latch minimization algorithm by taking advantage of latch sharing. Both of these new algorithms were practical improvements over existing algorithms that make TIM a faster and more powerful design tool.

We are currently in the process of collecting experimental data from more circuits. We plan to perform experiments with circuits that were originally designed as two-phase circuits. We also plan to perform experiments with circuits in which the nonuniformity in gate delays is amplified, in order to check if, in this case, tuning of the clocking schemes provides any speedups. Another interesting question that remains open, and we plan to address, is whether asymmetric clocking schemes can decrease the number of storage elements further than symmetric ones.

Bibliography

[1] 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.

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

[3] L. A. Glasser and D. W. Dobberpuhl. *The Design and Analysis of VLSI Circuits*. Addison-Wesley, Reading, Massachusetts, 1985.

[4] 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.

[5] 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.

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

[7] 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.

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

[9] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. Technical Report TM-372, MIT Laboratory for Computer Science, October 1988.

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

[11] 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.

[12] J. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Sloan Working Papers*, number 3060-89, August 1989.

[13] M. C. Papaefthymiou. Understanding retiming through maximum average-weight cycles. *3rd ACM Symposium on Parallel Algorithms and Architectures*, July 1991.

[14] M. C. Papaefthymiou and K. H. Randall. TIM: a timing package for two-phase, level-clocked circuitry. In *Proceedings of the 30th ACM/IEEE Design Automation Conference*, June 1993. Also available as an MIT VLSI Memo 92–693, October 1992.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

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