A Timing Analysis and Optimization System for Level-Clocked Circuitry

by

Marios Christos C. Papaefthymiou

B.S., Electrical Engineering
California Institute of Technology (1988)

S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology (1990)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1993

Author . . . .
Department of Electrical Engineering and Computer Science
August 28, 1993

Certified by . . . .
Charles E. Leiserson
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by . . . .
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

A Timing Analysis and Optimization System for Level-Clocked Circuitry

by
Marios Christos C. Papaefthymiou

Submitted to the Department of Electrical Engineering and Computer Science
on August 28, 1993, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis investigates timing analysis and optimization issues in synchronous circuitry. The major thrust of our work is a collection of provably correct and efficient algorithms that perform a variety of architectural-level operations on level-clocked circuitry, that is, circuitry that employs a multiphase clocking scheme and level-clocked storage elements. We implemented several of these algorithms in TIM, a timing package for two-phase, level-clocked circuitry. Using TIM we empirically compared the performance and the storage element requirements of edge-triggered and equivalent level-clocked implementations of synchronous circuitry. Our research contributes towards a better understanding of the complex timing issues involved in level-clocking and provides the enabling technology for bringing level-clocking into the mainstream of circuit design.

We begin by describing algorithms for optimizing edge-triggered circuitry in Chapter 1. This kind of circuitry is particularly popular among designers, because of its intuitive operation and ease of implementation. Our work in this area focuses on optimization by *retiming*, an architectural transformation that speeds up circuits by relocating their storage elements. A highlight of our research is an $O(V^{1/2}E \lg V)$-time algorithm for retiming unit-delay circuitry for maximum speed. This is the asymptotically fastest algorithm known to date for the problem.

In Chapter 2 we move on to investigate timing in a general class of level-clocked circuitry. The operation of this circuitry is much more complex and difficult to understand than that of edge-triggered circuitry. We first describe polynomial-time algorithms that analyze the timing of level-clocked circuitry. Specifically, we give algorithms that verify the proper timing of a circuit by a given clocking scheme and analyze the sensitivity of its timing to changes in the propagation delays of its components. We also describe a polynomial-time algorithm for *clock tuning*, an optimization that speeds up level-clocked circuits by adjusting the parameters of their clocking schemes. We extend retiming to encompass level-clocked circuitry, and we describe polynomial-time algorithms that perform retiming or simultaneous retiming and tuning. We also present a polynomial-time retiming algorithm that minimizes the number of storage elements in level-clocked circuitry without degrading its performance. Major results of our research in this area are an $O(VE + V^2 \lg V)$-time algorithm for retiming with symmetric clocking schemes and an $O(VE + V^2 \lg V)$-time algorithm for analyzing the sensitivity of a circuit's timing.

Chapter 3 describes TIM, a versatile and efficient design automation tool for two-phase, level-clocked circuitry that is based on our timing analysis and optimization algorithms. TIM has been implemented using the C programming language and has been integrated into the SIS tools from Berkeley. Our software runs on a workstation under the UNIX environment, and it is available over the Internet by ftp.

We employed TIM to empirically compare edge-triggered and functionally equivalent level-clocked implementations of synchronous circuitry in terms of speed and number of storage elements. Our results, which are presented in Chapter 4, show that although two-phase, level-clocked circuitry has the theoretical potential to operate faster than conventional edge-triggered circuitry, edge-triggered circuitry can often perform just as well. These empirical results indicate the special circumstances, however, in which level clocking has an advantage. Moreover, our empirical results also indicate another advantage of optimized level-clocked designs: they contain substantially fewer storage elements than edge-triggered designs that operate at the same speed.

Keywords: timing analysis, timing optimization, retiming, computer-aided design, VLSI design, digital systems, synchronous circuitry, multiphase circuitry, level-clocked circuitry, graph algorithms, combinatorial optimization.

Thesis Supervisor: Charles E. Leiserson
Title: Professor of Computer Science and Engineering

Acknowledgments

I am grateful to my advisor Charles Leiserson for guiding me throughout my career as a graduate student at MIT. Charles offered to me much more than his excellent advice on technical issues. He was my mentor and at the same time my collaborator. Two things impressed me with Charles. First, the personal, enthusiastic, and dedicated way in which he involves himself with his research and his students. Second, his continuous endeavor to be fair and objective. I wish I become as inspiring a scholar and a professional as he is.

I was fortunate to have Alex Ishii and Keith Randall as collaborators. We spent several hours with Alex discussing about timing. His witty perspective of every issue in life always offered a refreshing break after many hours of proving theorems on the whiteboard. Keith demonstrated the professionalism and sense of responsibility that I would never expect to find in an undergraduate student. I will be fortunate if I have graduate students like him.

Anant Agarwal and Srinivas Devadas, my thesis readers, were always quick with excellent ideas. Srinivas taught me a lot about computer-aided design, and Anant gave me the opportunity to expand my research to areas outside the main focus of my doctoral work.

I am thankful to Mike Dertouzos and Elias Gyftopoulos for always being happy to offer me their advice and share their experiences with me. I am also thankful to Ron Rivest for supporting me during the first year of my graduate studies.

The Theory of Computation Group initially, and subsequently the Computation Structures Group, the VLSI and Parallel Systems Group and the Supercomputing Technologies Group at MIT's Laboratory for Computer Science provided an unsurpassed environment for my research. The laboratories of Lee Gehrke and Helmut Zarbl offered me a quiet place to think and work after hours.

My officemates and groupmates were all wonderful and too many to enumerate without running the risk to forget someone. This is also true for all my friends outside the lab. They know who they are, and I thank them all.

Last, but not least, I want to thank those who stood by my side all these years. From my first year in Boston, I realized how special Maria was for me. She allowed me to see the best and the worst in me, and she always strove to help me in her best possible way. Back in Greece, my parents and my brother always believed in me and offered me the best. If I could only give to all of them, here and in Greece, more happiness in the years to come than they have given to me in any single day.

Contents

Introduction

For several years now, VLSI designers have been routinely implementing synchronous digital systems with clocked storage elements that are synchronized by the falling edge of a single clocking waveform. The operation of this *edge-triggered circuitry* is quite simple and intuitive, and it can be described in just two sentences. When the clocking waveform falls, each storage element instantaneously samples its input and asserts that value at its output for the remainder of the clock cycle. The circuit operates correctly if the time between every two consecutive falling edges is long enough to allow for all signals to propagate along every combinational path in the circuit. Due to its simplicity, edge-triggered circuitry has become particularly popular among both circuit designers and CAD tool designers.

An alternative method to implement synchronous systems is to employ a multiphase clocking scheme and to use level-clocked latches as storage elements. In this so-called *level-clocked circuitry*, a latch operates like a traffic light. While the clocking waveform is high, the latch is transparent and allows data to flow through unimpeded. When the clocking waveform falls, however, the latch samples its input signal and asserts it at its output for the remainder of the clock cycle. The signals that flow through a latch during its transparent phase can initiate the computation of the next combinational stage before the beginning of the next clock cycle, a phenomenon known as *cycle stealing*. Thus, level-clocked circuitry has the theoretical potential to operate faster than edge-triggered circuitry, and it is often employed in high-performance designs. Unfortunately, cycle stealing perplexes the operation of level-clocked circuitry, because data can ripple through several stages of storage elements before their propagation is complete. As a result, the design of level-clocked circuitry is notoriously difficult, and there are almost no automation tools available to facilitate this task.

In this thesis we present the enabling technology that will bring level-clocking into the mainstream of circuit design. Specifically, we describe a collection of provably correct and efficient algorithms for analyzing and optimizing the timing of level-clocked circuitry. We implemented several of our algorithms in a software package called TIM, and we used our tool to empirically compare edge-triggered and functionally equivalent level-clocked designs. We believe that the results of our research provide VLSI designers with the tools and the intuition required to design level-clocked circuitry with the same degree of confidence, ease, and efficiency that is customary for edge-triggered designs.

Motivation and Related Work

Since the early years of integrated digital systems, timing analysis and timing optimization had been identified as two of the most important problems in the design and implementation of synchronous circuitry. The development of sophisticated timing analysis tools for edge-triggered systems had already started in the early 70s, several years before the advent of VLSI design [17]. One of the best known programs for analyzing circuitry that employed edge-triggered latches was SCALD [34].

Level-clocked latches and multiphase clocks became commonplace with the arrival of VLSI. The first timing analysis programs that accounted for level-clocked latches were TV and CRYSTAL which appeared in the early 80s [23, 40]. These systems, however, were hampered by the fact that they could not handle signals that cross phases. Agrawal made an important contribution with his timing analysis program which accounted for signals that cross phases but not for signals that cross clock cycles [1]. In the mid 80s, Szymanski presented the timing analysis program LEADOUT which could handle signals that cross phases as well as signals that cross clock cycles [54]. A few years later, Ishii and Leiserson presented a formal timing analysis for a general class of level-clocked circuitry, and they proved that their algorithms run in polynomial time [20]. In their analysis, each block of combinational logic was assumed to have a minimum propagation delay equal to zero and a maximum propagation delay that was independent of the block's functionality.

The first procedures for optimizing the timing of level-clocked circuitry by selecting appropriate parameters for their clocking waveforms appeared in the late 80s [6, 57]. A mathematical framework for timing analysis and optimal selection of clocking parameters was presented by Sakallah, Mudge and Olukotun [51]. In this work, each block of combinational logic was assumed to have a nonzero minimum propagation delay and a nonzero maximum propagation delay that were independent of the block's functionality. The retiming optimization, a circuit transformation that has been extensively studied for edge-triggered circuitry [28, 29, 31, 45], was investigated in the context of single-phase, level-clocked circuitry by Shenoy, Brayton and Sangiovanni-Vincentelli [52]. In all these papers, the authors described iterative approximation or successive relaxation schemes to solve the problems. The proposed schemes, however, were not guaranteed to run in polynomial time.

In this thesis we present a rigorous investigation of timing in level-clocked circuitry. Assuming the same delay model as the one in [20], we describe provably correct and provably efficient algorithms for analyzing and optimizing the timing of a general class of level-clocked circuitry that employs multiphase clocking schemes. We believe that our work will transform the design of level-clocked circuitry into a substantially easier and more reliable task. Our timing analysis algorithms extend beyond simple timing verification by providing information about the timing slack of the combinational logic blocks in the circuit. These are the first polynomial-time algorithms presented for analyzing the timing sensitivity of level-clocked circuitry. Moreover, our timing optimization algorithms do not only select the optimal parameters for the clocking waveforms of a level-clocked circuit, but they also improve its performance by retiming, that is, by relocating its latches without changing its functionality. Retiming has been already investigated in edge-triggered circuitry [28, 29, 31, 45] and in single-phase, level-clocked cicuitry [52]. Our algorithms are the first, however, to extend retiming to level-clocked circuitry with multiple clocking waveforms.

Our work extends beyond the borders of theory. We implemented several of our algorithms in TIM, a timing package for two-phase, level-clocked circuitry. We have used TIM to empirically compare edge-triggered and functionally equivalent level-clocked designs. Our experiments demonstrate specific circuit characteristics that allow level-clocking to achieve its speed potential. They also show that level-clocking leads to circuits with fewer storage elements in high-performance designs. Despite the large amount of work in the area, our contribution is the first attempt to empirically quantify the performance differences of edge-triggering and two-phase clocking. We believe that our results will prove particularly useful to designers of custom integrated circuitry.

Independently of our work, Lockyear and Ebeling have presented retiming algorithms for multiphase, level-clocked circuitry in [32]. The general retiming algorithm in that paper has the same computational complexity as the one we present in this thesis. We present an asymptotically more efficient algorithm, however, when the clocking waveforms are symmetric. Some retiming heuristics were also presented in [3], but their correctness and computational complexity was not analyzed.

Recently, Szymanski and Shenoy presented a provably correct algorithm for timing verification that has the same computational complexity as the one we present in this thesis and assumes a more general delay model in which the blocks of combinational logic have nonzero minimum propagation delays [55]. An efficient algorithm for selecting the parameters of the optimal clocking waveforms in this more general model appeared in [56]. An efficient retiming algorithm for this model, however, is not known yet.

Overview of the Thesis

This thesis is organized in four chapters. Parts of the material in each chapter have appeared in conferences during the last three years. Other parts have been submitted for publication.

Optimizing Edge-Triggered Circuitry

In Chapter 1 we investigate algorithms for optimizing edge-triggered circuitry. Early versions of this work appeared in [42, 43].

An edge-triggered circuit comprises blocks of combinational logic that perform functions and edge-triggered latches (also called registers) that implement storage elements. Such a circuit is a simple implementation of the semisystolic model of computation that can be used to design parallel algorithms. In this chapter, we give tight bounds on the minimum clock period that can be achieved by retiming an edge-triggered circuit. Our bounds are independent of the size of the circuit; they are expressed in terms of the maximum ratio of the total delay over the total register count around any cycle in the circuit graph and in terms of the maximum propagation delay $d_{\max}$ of the combinational logic blocks. Moreover, our bounds characterize exactly the minimum clock period that can be achieved by retiming a *unit-delay* circuit, that is, a circuit in which all combinational logic blocks have propagation delay of one unit.

We also present more efficient algorithms for several important problems related to retiming. Specifically, we give an $O(V^{1/2}E \lg V)$-time algorithm for retiming unit-delay circuits with the minimum possible clock period. This is the asymptotically fastest algorithm known to date for this problem, and its efficiency stems from our exact characterization of the minimum clock period for unit-delay circuits. For the general case, in which circuits also include combinational logic blocks with non-unit delays, we describe an $O(VE \lg d_{\max})$-time algorithm for retiming with minimum clock period. We also describe an $O(V^{1/2} E \lg^2(V d_{\max}))$-time algorithm for retiming with a clock period that does not exceed the optimal by more than an additive factor of $d_{\max}-1$. Finally, we give an $O(E \lg d_{\max})$-time algorithm for pipelining combinational circuits with the minimum possible clock period.

Analyzing and Optimizing Level-Clocked Circuitry

In Chapter 2 we present our algorithms for analyzing and optimizing the timing of level-clocked circuitry. Earlier versions of this work appeared in [21, 22, 44, 46, 47, 48].

A level-clocked circuit comprises blocks of combinational logic that perform functions and level-clocked latches that implement storage elements. In this chapter we describe algorithms that verify whether a circuit is properly timed by a given clocking scheme and analyze the sensitivity of the circuit's timing to changes in the propagation delays of its components. We also investigate two strategies for reducing the clock period of a level-clocked circuit: *clock tuning*, which adjusts the waveforms that clock the circuit, and *retiming*, which relocates circuit latches. These methods can be used to convert a circuit with edge-triggered latches into a faster level-clocked one.

The algorithms in this chapter are presented in terms of two-phase, level-clocked circuitry. At the end of the chapter we extend our algorithms to encompass a general class of level-clocked circuitry with multiple phases. We model a two-phase circuit as a graph $G = \langle V, E \rangle$ whose vertex set V is a collection of combinational logic blocks, and whose edge set E is a set of interconnections. Each interconnection passes through zero or more latches, where each latch is clocked by one of two periodic, nonoverlapping waveforms, or *phases*.

We give efficient polynomial-time algorithms for problems involving the timing analysis and optimization of two-phase circuitry. Included are algorithms for

- verification of proper timing: $O(VE)$ time.
- noncritical sensitivity analysis for a single combinational block: $O(VE)$ time.
- noncritical sensitivity analysis for all combinational blocks: $O(VE + V^2 \lg V)$ time.
- critical sensitivity analysis for a single combinational block: $O(VE)$ time.
- minimization of clock period by clock tuning: $O(VE)$ time.
- retiming to achieve a given clock period when the phases are symmetric: $O(VE + V^2 \lg V)$ time.
- retiming to achieve a given clock period when either the duty cycle (high time) of one phase or the ratio of the phases' duty cycles is fixed: $O(V^3)$ time.

By characterizing the set of possible clock periods under any retiming of the circuit, we are able to obtain polynomial-time algorithms for clock period minimization by:

- retiming and tuning when the duty cycles of the two phases are required to be equal: $O(V^2 E)$ time.

- retiming and tuning when either the duty cycle of one phase is fixed or the ratio of the phases' duty cycles is fixed: $O(V^2E + V^3 \lg V)$ time.
- simultaneous retiming and clock tuning with no conditions on the duty cycles of the two phases: $O(V^{11})$ time.

Unfortunately, this last algorithm is not practical. For these problems, however, we present fully polynomial-time approximation schemes for clock period minimization within any given relative error $\epsilon > 0$. Specifically, we give an

- $O((VE + V^2 \lg V)\lg(V/\epsilon))$-time algorithm for retiming and tuning when the duty cycles of the two phases are required to be equal.
- $O(V^3 \lg(V/\epsilon))$-time algorithm for retiming and tuning when either the duty cycle of one phase is fixed or the ratio of the phases' duty cycles is fixed.
- $O(V^3(1/\epsilon)\lg(1/\epsilon) + (VE + V^2 \lg V)\lg(V/\epsilon))$-time algorithm for simultaneous retiming and clock tuning with no conditions on the duty cycles of the two phases.

The first two of these approximation algorithms can be used to obtain the optimum clock period in the special case where all propagation delays are integers.

At the end of this chapter, we generalize most of the results for two-phase clocking schemes to simple multiphase clocking disciplines, including ones with overlapping phases. Typically, the algorithms to verify and optimize the timing of k-phase circuitry are at most a factor of k slower than the corresponding algorithms for two-phase circuitry. Sensitivity analysis for all combinational blocks and retiming with symmetric phases, however, can still be performed in asymptotically the same number of steps as for two-phase circuitry.

TIM: A Timing Package for Level-Clocked Circuitry

In Chapter 3 we describe TIM, a versatile and efficient tool for verifying and optimizing the timing of two-phase, level-clocked circuitry. An earlier version of this work appeared in [48].

TIM is based on the algorithms that we present in Chapter 2 and performs a wide variety of functions such as timing verification, sensitivity analysis, clock tuning, retiming and clock tuning for maximum speed of operation, and retiming for minimum number of latches. In TIM we have extended our algorithms to handle nonideal latches. All latches are assumed to have equal propagation delays, equal setup times and equal hold times. Moreover, the implementation of our retiming algorithms does not relocate the input/output latches, thus preserving the input/output phases and the total latency of the circuit.

The entire software package has been developed using the C programming language in a UNIX environment. The system has been integrated in the SIS tools from Berkeley, and it is available over the Internet by ftp [1]. On a SPARCstation 2 with 64MB of main memory, each of TIM's timing analysis functions requires a couple of minutes for a circuit with approximately 1,500 gates. The retiming functions are slower, however, and they require approximately 35 minutes for a circuit of the same size. TIM's retiming algorithms operate on a dense graph representation of the problem. Almost half of the time required

[1] Copies of TIM can be obtained by sending a request to `marios@lcs.mit.edu`.

by its retiming operations is spent on constructing this graph. We believe that the practical performance of some of the retiming algorithms will be substantially improved by adjusting them to operate on a sparse graph representation.

Edge-Triggering vs. Level-Clocking

In Chapter 4 we present an empirical comparison of edge-triggered and two-phase, level-clocked circuitry in terms of speed and storage elements requirements. An earlier version of this work appeared in [47].

Level-clocked circuitry that employs a two-phase, nonoverlapping clocking scheme has the theoretical potential to operate up to twice as fast as edge-triggered circuitry. Using Tim, we have run experiments that demonstrate, however, that edge-triggering is often just as fast as two-phase clocking, and that the speed potential of two-phase clocking is generally not obtained except when the delay between any two consecutive latches is roughly uniform and close to the maximum combinational block delay. Moreover, our experiments show that asymmetrical clocking of a two-phase circuit often does not provide any speedup over optimal symmetric clocking schemes.

Level-clocking can lead to substantial reductions in the number of storage elements in a circuit, however. Our experiments show that for edge-triggered circuitry that has been retimed to operate with the minimum possible clock period, by replacing each edge-triggered latch by a pair of level-clocked latches and subsequently retiming the resulting two-phase circuit, the number of storage elements can be reduced by up to 38% without increasing the clock period of the final design or affecting its input/output specification. Reductions of greater than 25% were achieved for more than one third of the circuits we tested.

We ran our tests on MCNC benchmark circuits, AT&T communication circuits, and custom circuitry designed for MIT's Alewife machine.

Chapter 1

Optimizing Edge-Triggered Circuitry

1.1 Introduction

This chapter describes algorithms for optimizing edge-triggered circuitry, that is, synchronous circuits built of *functional elements* and globally clocked *registers*. Retiming, which was introduced in [27, 28, 29] and treated in [31], is a well-known design automation technique which transforms a given edge-triggered circuit into a faster circuit, that is, one with a shorter clock period, by relocating the registers of the given circuit while preserving its functionality. In this chapter we further investigate retiming and provide results of theoretical as well as practical interest. Specifically, we give tight bounds on the minimum clock period that can be achieved by retiming a circuit in terms of the maximum delay-to-register ratio of the cycles in the circuit graph and the maximum propagation delay of the combinational logic blocks in the circuit. These bounds do not depend on the size of the circuit and characterize exactly the minimum clock period that can be achieved by retiming a unit-delay circuit. We exploit these bounds to obtain asymptotically improved algorithms for several important problems related to retiming. A highlight of the research presented in this chapter is the asymptotically fastest algorithm known to date for retiming unit-delay circuitry.

We model a synchronous circuit according to [27, 28, 29] by a *circuit graph* $G = \langle V, E, d, w \rangle$. A vertex $v \in V$ corresponds to a functional element of the circuit, and an edge $u \xrightarrow{e} v \in E$ corresponds to a wire between the functional elements u and v. The integer edge-weight $w(e)$ is the number of registers on the directed edge e. The vertex-weight $d(v)$ is the propagation delay of the signals through the functional element v. For simplicity, and without any loss of generality, we assume that vertex-weights are also integers. This assumption is justified by the fact that digital computers represent data with only a finite number of bits. Figure 1-1(a) illustrates a synchronous circuit with unit-delay functional elements.

Intuitively, the circuit operates as follows. Between any two consecutive clock ticks, signals propagate along wires and ripple concurrently through the functional elements. By the end of a clock period all signals must have settled in the registers of the circuit. Although

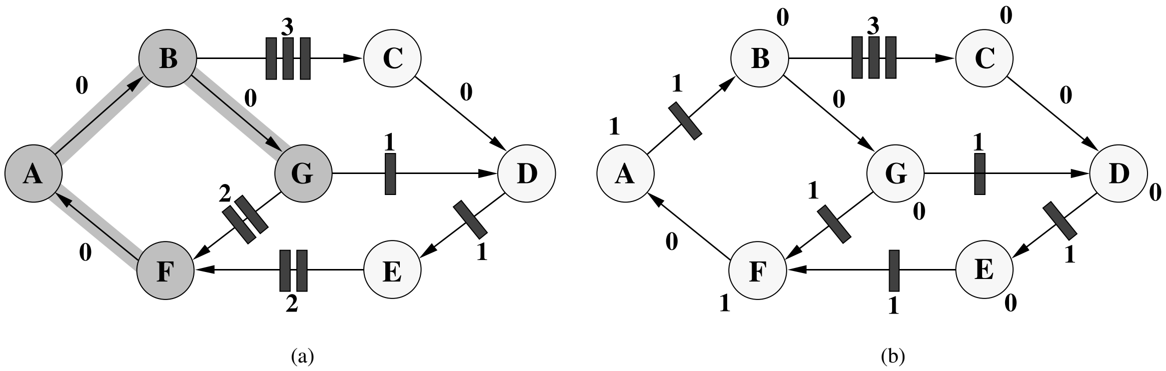


Figure 1-1: (a) A synchronous circuit G with unit-delay functional elements. The vertices represent functional elements, the edges represent wires, and the rectangles represent registers. The integers next to the edges indicate number of registers. The clock period of the circuit is 4 units of time (path FABG). (b) A retiming of G. The integer assignment is indicated next to the vertices. The clock period of this circuit is 2 units of time.

the functional elements of the circuit operate in parallel, some signals may require more time to settle than others, because they experience longer propagation delays along their paths. The *clock period* $\Phi(G)$ of the system is defined naturally as the propagation delay of the longest register-free path in the circuit, which is well-defined for *synchronous circuits* in which every directed cycle contains at least one register. For example, the clock period of the unit-delay circuit in Figure 1-1(a) is 4 units of time (path FABG).

A *retiming* r of G is an assignment $r : V \to \mathbf{Z}$, such that $w(e) + r(u) - r(v) \geq 0$. Given r, we transform the circuit by removing $r(v)$ registers from every edge coming into v, and adding $r(v)$ registers to every edge going out of v. This results to a retimed circuit $G_r = \langle V, E, d, w_r \rangle$, with $w_r(e) = w(e) + r(u) - r(v) \geq 0$ for every edge $u \xrightarrow{e} v \in E$. In Figure 1-1(b) we have retimed the circuit of Figure 1-1(a) so that the resulting circuit has clock period 2 units of time. Note that the total number of registers around any cycle in the circuit remains invariant after retiming.

The *delay-to-register* ratio of a directed cycle in a circuit is defined as the ratio of the total propagation delay around the cycle over the total number of registers in the cycle. For example, the delay-to-register ratio of the directed cycle ABCDEF in Figure 1-1 is $6/6 = 1$. Observe that the delay-to-register ratio of any cycle is the same in the original circuit G and in the retimed circuit G_r, since both the total delay and the number of registers around any cycle remain invariant after retiming. This observation suggests a relation between the minimum clock period that we can achieve by retiming G and the maximum delay-to-register ratio in G. Let us illustrate this relation by means of our example circuit in Figure 1-1. Consider the cycle ABGF with delay-to-register ratio $4/2 = 2$, which is the *maximum* among the three directed cycles in G. It is not possible to distribute the registers around ABGF in a way that achieves a clock period shorter than the average delay per register in ABGF, since the delay-to-register ratio around any cycle in G_r remains invariant. Therefore, G cannot be retimed to achieve a clock period smaller than 2, the delay-to-register ratio of ABGF, and the circuit in Figure 1-1(b) has achieved its minimum possible clock period.

In this chapter we prove that rounding up the maximum delay-to-register ratio in any circuit yields a lower bound on the minimum clock period that we can achieve by retiming the circuit. Moreover, we show that there always exists a retiming that comes within an additive factor of $d_{\max} - 1$ away from the lower bound, where $d_{\max}$ denotes the maximum among the propagation delays of the functional elements in the circuit. As a special case of these general bounds we have that the maximum delay-to-register ratio in any *unit-delay circuit* characterizes exactly the minimum clock period achievable by retiming. (The result for the special case of unit-delay circuitry has been claimed independently in [8].) Our tight bounds yield asymptotically more efficient algorithms for several important problems related to retiming, such as minimum clock-period retiming, retiming for approximately minimum clock-period, and minimum clock-period pipelining.

The remainder of this chapter is structured as follows. In Section 1.2 we give some background material on retiming and its relation with the single-source shortest-paths problem. In Section 1.3 we state and prove the tight bounds on the minimum clock period $\Phi_{\min}(G)$ that can be achieved by retiming a circuit G.

In Section 1.4, we give an $O(V^{1/2}E \lg V)$-time algorithm that retimes any unit-delay circuit to achieve the minimum possible clock period $\Phi_{\min}(G)$. This result improves the $O(VE \lg V)$ bound from [31]. Our algorithm is based on the exact characterization of $\Phi_{\min}(G)$ as well as on scaling algorithms for finding single-source shortest-paths and the minimum cycle-mean in a graph [12, 39].

In Section 1.5, we present algorithms for the general case, in which circuits include combinational logic blocks of non-unit delay. Specifically, Subsection 1.5.1 describes an $O(VE \lg d_{\max})$-time algorithm for minimum clock period retiming. This algorithm performs a preprocessing step for computing the maximum delay-to-register ratio in the circuit, followed by a binary search of $d_{\max}$ possible clock periods. Assuming that the maximum propagation delay $d_{\max}$ of the circuit components grows subpolynomially with the size of the circuit, our algorithm is asymptotically more efficient than the previously known schemes [31]. Subsection 1.5.2 presents an $O(V^{1/2}E \lg^2(V d_{\max}))$-time procedure for retiming a circuit with a clock period that does not exceed the minimum by more than an additive factor of $d_{\max} - 1$.

In Section 1.6 we extend our characterization of the minimum clock period to encompass combinational circuits, that is, circuits with no directed cycles in their graphs. We show how to pipeline any combinational circuit in $O(E \lg d_{\max})$ steps in order to achieve a specified latency with the minimum possible clock period. This result improves the $O(VE \lg V)$ bound that can be obtained by applying the general algorithms from [31], and it is optimal within a constant multiplicative factor for circuitry with unit-delay functional elements.

1.2 Retiming and Shortest Paths

In this section we define some notation and terminology needed in this chapter. We formulate retiming according to [31] as a set of difference constraints, and we introduce the notion of the constraint graph. Finally, we exhibit the relation between retiming and the existence of single-source shortest-paths in the constraint graph.

Given a circuit graph $G = \langle V, E, d, w \rangle$, we define the *path weight* $w(p)$ for any path

$p = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{k-1}} v_k$ in the circuit as the sum of the edge-weights of the path:

$$w(p) = \sum_{i=0}^{k-1} w(e_i).$$

We also define the *path delay* $d(p)$ as the sum of the delays of the vertices in the path:

$$d(p) = \sum_{i=0}^{k} d(v_i).$$

A *retiming* of a circuit G with *gain* r is an integer-valued vertex-labeling $r : V \to \mathbf{Z}$. The retiming r specifies a transformation of the original circuit in which registers are added and removed so as to change the original circuit G into a new circuit $G_r = \langle V, E, d, w_r \rangle$ with clock period $\Phi(G_r)$. The edge-weighting w_r is defined for an edge $u \xrightarrow{e} v$ of G_r by the equation

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

The following theorem characterizes the conditions under which we can find a retiming that produces a circuit with clock period no greater than a given constant.

Theorem 1 ([31]) *Let $G = \langle V, E, d, w \rangle$ be a synchronous circuit, let c be an arbitrary positive real number, and let r be a function from V to the integers. Then, r is a legal retiming of G such that $\Phi(G_r) \leq c$ if and only if*

$$r(v) - r(u) \leq w(e) \tag{1.1}$$

for every edge $u \xrightarrow{e} v$ in G, and

$$r(v) - r(u) \leq W(u, v) - 1 \tag{1.2}$$

for all vertices $u, v \in V$ such that $D(u, v) > c$, where

$$\begin{aligned} W(u,v) &= \min\left\{w(p) : u \overset{p}{\rightsquigarrow} v\right\}, \\ D(u,v) &= \max\left\{d(p) : u \overset{p}{\rightsquigarrow} v \text{ and } w(p) = W(u,v)\right\}. \end{aligned}$$

□

Inequality (1.1) guarantees that the number of registers on every edge $u \xrightarrow{e} v$ of the retimed circuit G_r is nonnegative. Inequality (1.2) enforces the clock period constraint: every simple path $u \xrightarrow{p} v$ with delay $d(p) > c$ will have at least one register in the retimed circuit G_r. There are potentially $O(V^2)$ inequalities of the form (1.2), one for each pair of vertices in G, and they can be computed in $O(VE + V^2 \lg V)$ steps [31].

The constraints (1.1) and (1.2) in Theorem 1 are linear inequalities involving only differences of the unknowns $r(v)$. Therefore, the retiming problem can be expressed in the following general form.

Problem DC *(Difference Constraints) Let L be a set of m linear constraints of the form*

$$x(v) - x(u) \leq t(u, v) \tag{L}$$

on the n unknowns $x(1), x(2), \ldots, x(n)$*, where* $t(u,v)$ *are given integer constants. Determine a set of integer feasible values for the unknowns* $x(u)$ *or determine that no such set exists.* □

The following theorem is classic in the field of combinatorial optimization [25, 41], and provides a method for solving Problem DC.

Theorem 2 *Let* L *be a set of difference constraints, and let* $\hat{G} = (\hat{V}, \hat{E}, \hat{t})$ *be the directed, edge-weighted, constraint graph constructed from* L *in the following way:*

$$\begin{aligned}
\hat{V} &= \{u : x(u) \textit{ is an unknown of } L\}; \\
\hat{E} &= \{u \to v : x(v) - x(u) \leq t(u,v) \textit{ is a constraint in } L\}; \\
\hat{t}(u,v) &= t(u,v), \textit{ for every edge } u \to v \textit{ in } \hat{E}.
\end{aligned}$$

Then, Problem DC is feasible if and only if there exists no directed cycle $C \in \hat{G}$ *with edge-weight* $\hat{t}(C) < 0$*. Moreover, let every vertex* $v \in \hat{V}$ *be reachable from a vertex* $s \in \hat{V}$ *by a path* $s \leadsto v$ *in* $\hat{G}$*. If there exists a solution* r *to the shortest-paths problem in* $\hat{G}$ *from the source* s*, that is, if* $r(v) = \min\{\hat{t}(p) : s \xrightarrow{p} v \in \hat{G}\}$ *for every vertex* $v \in \hat{V}$*, then* r *is also a solution for the constraint set* L*, such that* $x(v) - x(s)$ *is maximized for every vertex* $v \in \hat{V}$*.*

We denote by $G_c = \langle V, E_c, w_c \rangle$ the constraint graph that corresponds to Inequalities (1.1) and (1.2) for a given c. Theorem 2 implies that a retimed circuit with clock period no greater than c can be computed in $O(V^3)$ steps by applying the $O(VE)$-time shortest-paths algorithm by Bellman and Ford [25, page 74] on the dense constraint graph G_c. An asymptotically faster algorithm which runs in $O(VE)$ time appears in [31].

1.3 Characterization of Minimum Clock Period $\Phi_{\min}(G)$

1.3.1 Bounds on $\Phi_{\min}(G)$

In this section we characterize the minimum clock period $\Phi_{\min}(G)$ that can be obtained by retiming a given circuit $G = \langle V, E, d, w \rangle$ in terms of the maximum delay-to-register ratio of the cycles in the circuit graph G and the maximum propagation delay of the circuit components.

First, we give some definitions that will allow us to state and prove our results formally. We define the *delay-to-register* ratio $R(C)$ of a cycle $C = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{k-2}} v_{k-1} \xrightarrow{e_{k-1}} v_0$ in the circuit G as follows:

$$R(C) = \frac{\sum_{v \in C} d(v)}{\sum_{e \in C} w(e)}.$$

We denote by $C^*(G)$ the directed cycle in G with maximum delay-to-register ratio. By definition, $R(C^*(G)) \geq R(C)$ for every cycle $C \in G$. Finally, we denote by $\Phi_{\min}(G)$ the smallest possible clock period that we can achieve by retiming G:

$$\Phi_{\min}(G) = \min\{\Phi(G_r) : r \text{ is a retiming of } G\}.$$

Our first theorem bounds the range of the minimum clock period of a circuit.

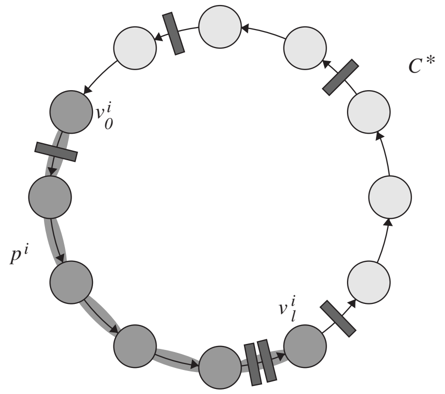


Figure 1-2: Path p^i used in the proof of the lower bound. Note that only the first and the last edge in the path have non-zero register count.

Theorem 3 *Let $G = \langle V, E, d, w\rangle$ be a synchronous circuit with maximum delay-to-register ratio $R(C^*(G))$, and let $\Phi_{\min}(G)$ be the minimum clock period we can obtain by retiming G. Then*

$$\lceil R(C^*(G))\rceil \le \Phi_{\min}(G) \le \lceil R(C^*(G))\rceil + d_{\max} - 1,$$

where $d_{\max} = \max\{d(v) : v \in V\}$.

The proofs of the lower and the upper bound are given in Sections 1.3.2 and 1.3.3 respectively. Observe that both the upper and the lower bound are independent of the number of vertices and the number of edges in the circuit.

For unit-delay circuits, the bounds in Theorem 3 yield an *exact* characterization of the minimum clock period.

Corollary 4 *Let $G = \langle V, E, 1, w\rangle$ be a unit-delay synchronous circuit with maximum delay-to-register ratio $R(C^*(G))$, and let $\Phi_{\min}(G)$ be the minimum clock period we can obtain by retiming G. Then*

$$\lceil R(C^*(G))\rceil = \Phi_{\min}(G).$$

Proof. Follows directly from Theorem 3 for $d_{\max} = 1$. □

As we shall see in Section 1.4, this property of unit-delay circuits allows us to derive asymptotically more efficient schemes for their optimization.

1.3.2 Lower Bound

In this section we prove the lower bound of Theorem 3. Specifically, we prove the following lemma.

Lemma 5 *Let $G = \langle V, E, d, w\rangle$ be a synchronous circuit with maximum delay-to-register ratio $R(C^*(G))$, and let $\Phi_{\min}(G)$ be the minimum clock period we can obtain by retiming G. Then*

$$\lceil R(C^*(G))\rceil \le \Phi_{\min}(G).$$

Proof. Assume that we have retimed G in such a way that it achieves the minimum possible clock period $\Phi_{\min}(G)$. Let C^* be the cycle with maximum delay-to-register ratio in G (see Figure 1-2), and let $N = \{e \in C^* : w_r(e) > 0\}$. Now, consider a path $p^i = v_0^i \xrightarrow{e_0^i} v_1^i \xrightarrow{e_1^i} \ldots \xrightarrow{e_{l-2}^i} v_{l-1}^i \xrightarrow{e_{l-1}^i} v_l^i$ in C^*, such that $e_0^i, e_{l-1}^i \in N$ and $e_j^i \notin N$, for $j = 1, 2, \ldots, l-2$. Observe that only the first and the last edge in the path p^i have registers on them. Now, by definition of the clock period, the register-free part of p^i satisfies

$$\sum_{j=1}^{l-1} d(v_j^i) \leq \Phi_{\min}(G).$$

There are $|N|$ paths around C^* that have the form of p^i. By summing up the $|N|$ corresponding inequalities for $\Phi_{\min}(G)$, we obtain

$$\begin{aligned}\sum_{v \in C^*} d(v) &\leq \Phi_{\min}(G) \cdot |N| \\ &\leq \Phi_{\min}(G) \cdot \left(\sum_{e \in C^*} w_r(e) \right) \\ &\leq \Phi_{\min}(G) \cdot \left(\sum_{e \in C^*} w(e) \right),\end{aligned}$$

since $w_r(e) = w(e) + r(u) - r(v)$ for every edge $u \xrightarrow{e} v$, and the sum $\sum_{e \in C^*} w_r(e)$ telescopes. Since the propagation delays are integers, $\Phi_{\min}(G)$ must also be an integer, and therefore

$$\left\lceil \frac{\sum_{v \in C^*} d(v)}{\sum_{e \in C^*} w(e)} \right\rceil \leq \Phi_{\min}(G).$$

The lemma follows directly from the definition of $R(C^*(G))$. □

1.3.3 Upper Bound

In this section we prove the upper bound of Theorem 3. Our proof uses properties of the *constraint graph* G_c that was introduced in Section 1.2, and Lemma 6, whose correctness follows directly from Theorem 2 and the definition of the constraint graph.

Lemma 6 *Given a circuit $G = \langle V, E, d, w \rangle$ and a real number c, there exists a retiming r of G such that $\Phi(G_r) \leq c$ if and only if the constraint graph G_c has no negative edge-weight cycles.* □

We proceed with the proof of the upper bound.

Lemma 7 *Let $G = \langle V, E, d, w \rangle$ be a synchronous circuit with maximum delay-to-register ratio $R(C^*(G))$, and let $\Phi_{\min}(G)$ be the minimum clock period we can obtain by retiming G. Then*

$$\Phi_{\min}(G) \leq \lceil R(C^*(G)) \rceil + d_{\max} - 1.$$

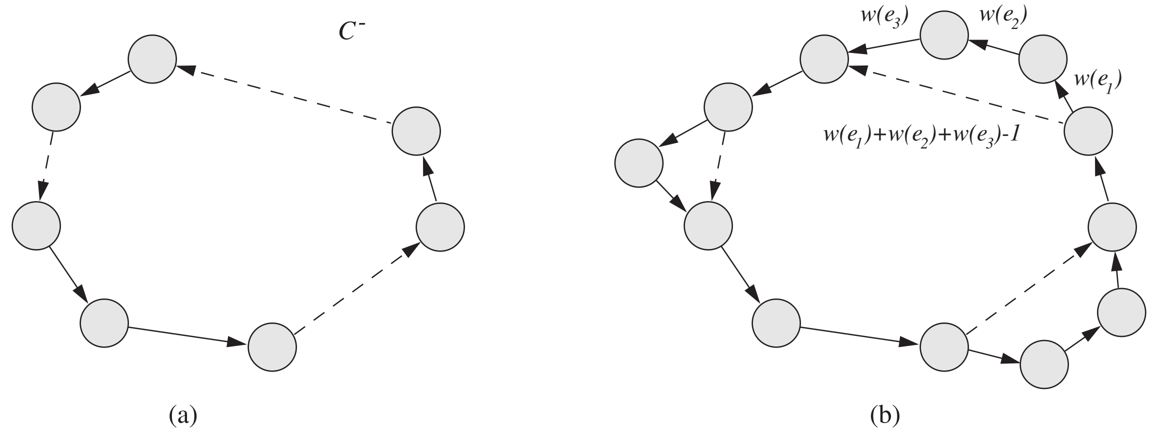


Figure 1-3: (a) The cycle C^- in G_c. The solid lines indicate edges in S. For simplicity, we have not indicated the registers on these edges. The broken lines indicate edges in S'. (b) After the introduction of the edges in S'' that correspond to the edges in S', the solid cycle has edges exclusively in G and its delay-to-register ratio is greater than $R(C^*(G))$.

Proof. The proof is by contradiction of the fact that $R(C^*(G))$ is the maximum delay-to-register ratio in G. Let us assume that there does not exist a retiming r, such that $\Phi(G_r) \leq \lceil R(C^*(G)) \rceil + d_{\max} - 1$. Equivalently, according to Lemma 6, we assume that for $c = \lceil R(C^*(G)) \rceil + d_{\max} - 1$ there exists a negative edge-weight cycle C^- in the constraint graph $G_c = (V, E \cup E', w_c)$ (see Figure 1-3(a)), where E' denotes the edges of G_c introduced due to Inequality (1.2). We can partition the edges of C^- into $C^- = S \cup S'$, where $S \subseteq E$ and $S' \subseteq E'$. Since the edge-weights $w_c(e)$ are integral, we have

$$\sum_{e \in S} w_c(e) + \sum_{e \in S'} w_c(e) \leq -1. \tag{1.3}$$

Now, Inequality (1.2) implies that every edge $u \xrightarrow{e} v$ in E' with weight $w_c(e) = W(u,v) - 1$ corresponds to a path $u \xrightarrow{p} v$ in E with weight $w_c(p) = W(u,v)$ and delay $d(p) > c$. Let $S'' = \{v_1 \rightarrow v_2 : u \rightarrow v \in S' \text{ corresponds to path } p \in G, v_1 \rightarrow v_2 \in p\}$ (see Figure 1-3(b)). Then, using Inequality (1.3) we obtain

$$\begin{aligned}
\sum_{e \in S} w(e) + \sum_{e \in S''} w(e) &= \sum_{e \in S} w(e) + \sum_{e \in S''} w(e) - |S'| + |S'| \\
&= \sum_{e \in S} w(e) + \sum_{e \in S'} w_c(e) + |S'| \\
&= \sum_{e \in S} w_c(e) + \sum_{e \in S'} w_c(e) + |S'| \\
&\leq |S'| - 1.
\end{aligned}$$

Note that $|S'| \geq 2$, because otherwise the cycle $S \cup S''$ would have no registers, which contradicts the fact that G is synchronous. Now, for the delay-to-register ratio of the cycle

```
UD-RETIMING(G)
for each edge e ∈ E
    do w'(e) ← min{w(e), |V|}
Φ_min(G') ← ⌈ max_{C∈G'} |C| / Σ_{e∈C} w'(e) ⌉        ▷ Φ_min(G') = ⌈R(C*(G'))⌉
for each vertex v ∈ V
    do r(v) ← ⌈length of single-source shortest-path s ⇝ v in G' − 1/Φ_min(G')⌉
for each edge u →e v ∈ E
    do w_r(e) ← w(e) + r(u) − r(v)
```

Figure 1-4: Algorithm UD-RETIMING for optimal retiming of unit-delay circuitry. Given a unit-delay circuit $G = \langle V, E, 1, w\rangle$, this algorithm determines a retimed circuit G_r with minimum clock period.

$S \cup S''$ in G we have:

$$\begin{aligned}
\frac{\sum_{u \xrightarrow{e} v \in S} d(u) \ + \sum_{u \xrightarrow{e} v \in S''} d(u)}{\sum_{u \xrightarrow{e} v \in S} w(e) \ + \sum_{u \xrightarrow{e} v \in S''} w(e)} &\geq \frac{\sum_{u \xrightarrow{e} v \in S} d(u) \ + \sum_{u \xrightarrow{e} v \in S''} d(u)}{|S'| - 1} \\
&\geq \frac{\sum_{u \xrightarrow{e} v \in S''} d(u)}{|S'| - 1} \\
&> \frac{|S'|(c + 1 - d_{\max})}{|S'| - 1} \\
&\geq \frac{|S'|}{|S'| - 1} R(C^*(G)).
\end{aligned}$$

Since $|S'|/(|S'|-1) > 1$, we conclude that there exists a cycle in G with delay-to-register ratio greater than the maximum delay-to-register ratio $R(C^*(G))$, which is a contradiction. Therefore, G_c has no negative edge-weight cycles and, according to Lemma 6, there exists a retiming r of G such that $\Phi(G_r) \leq \lceil R(C^*(G)) \rceil + d_{\max} - 1$. Consequently $\Phi_{\min}(G) \leq \lceil R(C^*(G)) \rceil + d_{\max} - 1$. □

1.4 Optimal Retiming of Unit-Delay Circuitry

In this section we describe an efficient algorithm for optimal retiming of edge-triggered circuitry in the special case where all combinational logic blocks have unit propagation delays. Specifically, we give an $O(V^{1/2} E \lg V)$-time procedure for the following problem: Given a unit-delay edge-triggered circuit $G = \langle V, E, 1, w\rangle$, determine a retiming r such that $\Phi(G_r) = \Phi_{\min}(G)$.

Our Algorithm UD-RETIMING for optimal retiming of unit-delay circuitry is illustrated in Figure 1-4. The basic idea behind it is the construction of a circuit $G' = \langle V, E, 1, w'\rangle$ with small edge-weights $w'(e)$, such that r is a retiming of G' with clock period $\Phi(G'_r)$ if and only if r is a retiming of G with clock period $\Phi(G_r) = \Phi(G'_r)$. Optimal retimings of G

can be computed more efficiently on G' using procedures that scale the small edge-weights $w'(e)$.

In order to prove the correctness of Algorithm UD-RETIMING, we must show that the optimal clock period $\Phi_{\min}(G)$ equals the optimal clock period $\Phi_{\min}(G')$ computed in Step 2 of the algorithm. This equality is a special case of the following general theorem.

Theorem 8 *Let $G = \langle V, E, d, w\rangle$ be an edge-triggered circuit, and let $G' = \langle V, E, d, w'\rangle$ be the edge-triggered circuit with $w'(e) = \min\{w(e), |V|d_{\max}\}$ for every edge $u \xrightarrow{e} v \in E$. Let $R(C^*(G))$ and $R(C^*(G'))$ be the maximum delay-to-register ratios of G and G' respectively. Then*

$$\lceil R(C^*(G))\rceil = \lceil R(C^*(G'))\rceil .$$

Proof. In order to prove the theorem, we first argue that the maximum $R(C(G))$ is obtained for a *simple* cycle in G. Indeed, if for any non-simple cycle $C = C_1 \cup C_2$ with $R(C_1) \geq R(C_2)$ we have $R(C) > R(C_1)$, then a straightforward calculation shows that $R(C_1) < R(C_2)$ which contradicts our assumption about $R(C_1)$ and $R(C_2)$.

Now, we show that every simple cycle $C \in G$ must satisfy

$$\left\lceil \frac{\sum_{v\in C} d(v)}{\sum_{e\in C} w(e)} \right\rceil = \left\lceil \frac{\sum_{v\in C} d(v)}{\sum_{e\in C} w'(e)} \right\rceil . \tag{1.4}$$

If $w(e) \leq |V|d_{\max}$ for every $e \in C$, then $w(e) = w'(e)$ for every $e \in C$ and equation (1.4) holds. If $w(e) > |V|d_{\max}$ for some edge $e \in C$, then $w'(e) = |V|d_{\max}$. Since $|C| \leq |V|$, we have

$$\left\lceil \frac{\sum_{v\in C} d(v)}{\sum_{e\in C} w(e)} \right\rceil = \left\lceil \frac{\sum_{v\in C} d(v)}{\sum_{e\in C} w'(e)} \right\rceil = 1.$$

Therefore, equation (1.4) holds again. □

For unit-delay circuitry, it follows that $\Phi_{\min}(G)$ equals $\Phi_{\min}(G')$.

Corollary 9 *Let $G = \langle V, E, 1, w\rangle$ be a unit-delay circuit, and let $G' = \langle V, E, 1, w'\rangle$ be the unit-delay circuit with $w'(e) = \min\{w(e), |V|\}$ for every edge $u \xrightarrow{e} v \in E$. Let $\Phi_{\min}(G)$ and $\Phi_{\min}(G')$ be the minimum clock periods that can be achieved by retiming G and G' respectively. Then*

$$\Phi_{\min}(G) = \Phi_{\min}(G').$$

Proof. Follows directly from Corollary 4 and Theorem 8 for $d_{\max} = 1$. □

Step 3 of Algorithm UD-RETIMING computes an optimal retiming of G', since a retiming that achieves a given clock period c can be computed for any unit-delay circuit G by rounding-up the shortest-paths lengths in the graph $G - 1/c = (V, E, w - 1/c)$ with edge-weight $w(e) - 1/c$ for each edge $e \in E$ [31]. Now, since $w'(e) \leq w(e)$ for every edge $e \in E$, a retiming of G' with clock period $\Phi(G'_r)$ is also a retiming of G with clock period $\Phi(G_r) = \Phi(G'_r)$. Therefore, Step 4 correctly computes an optimally retimed G_r.

OPT-RETIMING(G)
1 **for** each edge $u \xrightarrow{e} v \in E$
  do $w'(e) \leftarrow \min\{w(e), |V|d_{\max}\}$
2 Binary search $[1, |V|d_{\max}]$ for smallest integer n, $\triangleright$ $n = \lceil R(C^*(G')) \rceil$
  such that $G' - d/n$ has no negative edge-weight cycles.
3 Binary search $[n, n + d_{\max} - 1]$ for smallest integer period
  that can be achieved by a retiming r of G.
4 **for** each edge $u \xrightarrow{e} v \in E$
  do $w_r(e) \leftarrow w(e) + r(u) - r(v)$

Figure 1-5: Algorithm OPT-RETIMING for optimal retiming of edge-triggered circuitry. Given an edge-triggered circuit $G = \langle V, E, d, w \rangle$, this algorithm determines a retimed circuit G_r with minimum clock period.

Algorithm UD-Retiming terminates in $O(V^{1/2} E \lg V)$ time. Steps 1 and 4 can be computed in $O(E)$ time. In Step 2, since every functional element in G' has unit delays, we have $R(C^*(G')) = 1/\mathrm{mcm}(G')$, where

$$\mathrm{mcm}(G') = \min_{C \in G'} \frac{\displaystyle\sum_{C \in G'} w'(e)}{|C|}$$

is known as the minimum cycle-mean of G'. Ahuja and Orlin [39] have presented an algorithm for computing the minimum cycle-mean of a graph in $O(V^{1/2} E \lg(VW))$ steps, where W is the maximum edge-weight in the graph.[1] Since $w'(e) \le |V|$ for every edge $e \in E$, we can use this algorithm in Step 2 to compute $\Phi_{\min}(G')$ in $O(V^{1/2} E \lg V)$ time. The shortest-paths lengths in Step 3 can also be computed in $O(V^{1/2} E \lg V)$ time, using an $O(V^{1/2} E \lg(VW))$-time algorithm for shortest-paths by Gabow and Tarjan [12]. Thus, the total running time of Algorithm UD-RETIMING is $O(E) + O(V^{1/2} E \lg V) + O(V^{1/2} E \lg V) + O(E) = O(V^{1/2} E \lg V)$.

1.5 Retiming of Circuitry that Includes Non-Unit Delays

1.5.1 Optimal Retiming

In this section we describe an $O(VE \lg d_{\max})$-time algorithm for the optimal retiming problem in its general form: Given an edge-triggered circuit $G = \langle V, E, d, w \rangle$, determine a retiming r such that $\Phi(G_r) = \Phi_{\min}(G)$. An $O(VE \lg V)$-time algorithm that binary searches the $O(V^2)$-size set of all possible clock periods for the minimum feasible one appears in [31]. Our procedure binary searches an interval with only $d_{\max}$ possible clock periods, and it is more efficient than that in [31], assuming that $d_{\max}$ grows subpolynomially with respect to the number of functional elements in the circuit.

[1] This W should not be confused with $W(u, v)$ in Theorem 1.

```
APPROX-RETIMING(G)
1  for each edge u -e-> v ∈ E
       do w'(e) ← min{w(e), |V|d_max}
2  Binary search [1, |V|d_max] for smallest integer n,        ▷ n = ⌈R(C*(G'))⌉
   such that G' − d/n has no negative edge-weight cycles.
3  for each vertex v ∈ V
       do r(v) ← ⌈length of single-source shortest-path s ⇝ v in G' − d/n⌉
4  for each edge u -e-> v ∈ E
       do w_r(e) ← w(e) + r(u) − r(v)
```

Figure 1-6: Algorithm APPROX-RETIMING for retiming with approximately minimum clock period. Given an edge-triggered circuit $G = \langle V, E, d, w\rangle$, this algorithm determines a retimed circuit G_r with clock period $\Phi(G_r) \leq \Phi_{\min}(G) + d_{\max} - 1$.

Our Algorithm OPT-RETIMING is illustrated in Figure 1-5. In Steps 1 and 2, the algorithm computes the maximum delay-to-registers ratio $\lceil R(C^*(G'))\rceil$ of the circuit $G' = \langle V, E, d, w'\rangle$ with $w'(e) \leq |V|d_{\max}$. This ratio equals the smallest integer n in the interval $[1, |V|d_{\max}]$ of possible ratios that does not induce negative edge-weight cycles in the graph $G' - d/n = (V, E, w' - d/n)$ with edge-weight $w'(e) - d(v)/n$ for each edge $u \xrightarrow{e} v \in E$ [25]. Step 3 of the algorithm binary searches the integers in the interval $[\lceil R(C^*(G'))\rceil, \lceil R(C^*(G'))\rceil + d_{\max} - 1]$ for the minimum achievable clock period $\Phi_{\min}(G)$. Theorems 3 and 8 and the integrality of the propagation delays guarantee that $\Phi_{\min}(G)$ is an integer in this interval. The retiming r that corresponds to $\Phi_{\min}(G)$ is used in Step 4 to compute an optimally retimed G_r.

Algorithm OPT-RETIMING runs in $O(VE \lg d_{\max})$ time. Step 1 completes in $O(E)$ time. Negative-weight cycles in Step 2 can be detected by solving a single-source shortest-paths problem on the edge-weighted graph $G' - d/n$ [25]. Gabow and Tarjan have given an $O(V^{1/2}E \lg^2(VW))$-time algorithm for the single-source shortest-paths problem, where W is the maximum edge-weight in the graph [12]. Thus, the binary search in Step 2 can be performed in $O(V^{1/2}E \lg^2(V d_{\max}))$ time, since $w'(e) \leq |V|d_{\max}$ for every edge $e \in E$. Step 3 utilizes the $O(VE)$ retiming algorithm by Leiserson and Saxe [31] to test whether a potential clock period is feasible. Thus, a retiming that achieves $\Phi_{\min}(G)$ is computed in $O(VE \lg d_{\max})$ time, and the optimally retimed circuit is computed in Step 4 in $O(E)$ time. The overall running time is $O(E) + O(V^{1/2}E \lg^2(V d_{\max})) + O(VE \lg d_{\max}) + O(E) = O(VE \lg d_{\max})$.

1.5.2 Approximately Optimal Retiming

In this section we give an $O(V^{1/2}E \lg^2(V d_{\max}))$-time algorithm for retiming of a circuit so that its clock period is approximately minimized. Specifically, we consider the following problem: Given an edge-triggered circuit $G = \langle V, E, d, w\rangle$ determine a retiming r such that $\Phi(G_r) \leq \Phi_{\min}(G) + d_{\max} - 1$.

Our Algorithm APPROX-RETIMING for retiming with approximately minimum clock period is illustrated in Figure 1-6. The first two steps of the algorithm are the same as those in Algorithm OPT-RETIMING. In Step 3, a retiming r is obtained simply by rounding-up the shortest-paths lengths in the graph $G' - d/n = (V, E, w' - d/n)$ with edge-weights $w'(e) - d(v)/n$ for each edge $u \xrightarrow{e} v \in E$, where $n = \lceil R(C^*(G')) \rceil$. The following theorem shows that the period $\Phi(G_r)$ of the retimed circuit G_r does not exceed $\Phi_{\min}(G)$ by more than $d_{\max} - 1$.

Theorem 10 *Let $G = \langle V, E, d, w \rangle$ be a circuit graph with maximum delay-to-register ratio $R(C^*(G))$ and let $\Phi_{\min}(G)$ be the minimum clock period we can obtain by retiming G. Let $G' = \langle V, E, d, w' \rangle$ be the circuit with $w'(e) = \min\{w(e), |V| d_{\max}\}$ for every edge $u \xrightarrow{e} v \in E$. Moreover, let $n = \lceil R(C^*(G')) \rceil$, and let l be the solution of a single-source shortest-paths problem on the graph $G' - d/n = (V, E, w' - d/n)$ with edge-weight $w'(e) - d(v)/n$ for each edge $u \xrightarrow{e} v \in E$. Then, the assignment $r(v) = \lceil l(v) \rceil$ for each vertex $v \in V$ is a retiming of G such that*

$$\Phi(G_r) \leq \Phi_{\min}(G) + d_{\max} - 1.$$

Proof. Before we proceed with the proof, we note that $G' - d/n$ has no negative edge-weight cycles, since $n = \lceil R(C^*(G')) \rceil$. Therefore, the shortest-paths lengths $l(v)$ in $G' - d/n$ are well-defined. Now, in order to prove that $r(v) = \lceil l(v) \rceil$ is a legal retiming of G with clock period $\Phi(G_r) \leq \Phi_{\min}(G) + d_{\max} - 1$, we must show that $w_r(e) = w(e) + r(u) - r(v) \geq 0$ for every edge $u \xrightarrow{e} v \in G_r$, and that every path $p \in G_r$ with delay $d(p) > \Phi_{\min}(G) + d_{\max} - 1$ has at least one register.

First, we prove that the assignment $r(v) = \lceil l(v) \rceil$ for each vertex $v \in V$ satisfies $w_r(e) \geq 0$ for every edge e in the retimed circuit G_r. Since l is a single-source shortest-paths solution on $(V, E, w' - d/n)$, we have $l(v) \leq l(u) + w'(e) - d(v)/n$ for every edge $u \xrightarrow{e} v$ in E. Therefore

$$\begin{aligned}
\lceil l(v) \rceil - \lceil l(u) \rceil &\leq \lceil l(v) - l(u) \rceil \\
&\leq \lceil w'(e) - d(v)/n \rceil \\
&\leq w'(e) \\
&\leq w(e),
\end{aligned}$$

since $\lceil x \rceil - \lceil y \rceil \leq \lceil x - y \rceil$ for every real x, y, and since $w(e)$ is an integer. It follows that $w_r(e) = w(e) + \lceil l(u) \rceil - \lceil l(v) \rceil \geq 0$.

Now, we show that the assignment $r(v) = \lceil l(v) \rceil$ for each vertex $v \in V$ satisfies the clock period constraint. Consider any path $p = u_0 \xrightarrow{e_0} u_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{k-2}} u_{k-1} \xrightarrow{e_{k-1}} u_k$ in the retimed circuit G_r with delay $\sum_{i=0}^{k} d(u_i) > \Phi_{\min}(G) + d_{\max} - 1$. For this path we have

$$\begin{aligned}
l(u_k) - l(u_0) &\leq \sum_{i=0}^{k-1} \left(w'(e_i) - \frac{d(u_{i+1})}{n} \right) \\
&= \left(\sum_{i=0}^{k-1} w'(e_i) \right) - \left(\sum_{i=0}^{k} \frac{d(u_i)}{n} \right) + \frac{d(u_0)}{n} \\
&\leq \left(\sum_{i=0}^{k-1} w'(e_i) \right) - \frac{\Phi_{\min}(G) + d_{\max}}{n} + \frac{d(u_0)}{n} \\
&\leq \left(\sum_{i=0}^{k-1} w'(e_i) \right) - \frac{\Phi_{\min}(G)}{n} - \frac{d_{\max} - d(u_0)}{n}
\end{aligned}$$

$$\leq \left(\sum_{i=0}^{k-1} w'(e_i)\right) - 1,$$

since $\Phi_{\min}(G) \geq \lceil R(C^*(G')) \rceil$ from Theorems 3 and 8, and since $d_{\max} \geq d(u_0)$ by definition. Therefore, the number $w_r(p)$ of registers along the path p in the retimed circuit satisfies

$$\begin{aligned}
\sum_{i=0}^{k-1} w_r(e_i) &= \sum_{i=0}^{k-1} (w(e_i) + r(u_0) - r(u_k)) \\
&= \left(\sum_{i=0}^{k-1} w(e_i)\right) + \lceil l(u_0) \rceil - \lceil l(u_k) \rceil \\
&\geq \left(\sum_{i=0}^{k-1} w(e_i)\right) - \lceil l(u_k) - l(u_0) \rceil \\
&\geq \left(\sum_{i=0}^{k-1} w(e_i)\right) - \left\lceil \left(\sum_{i=0}^{k-1} w'(e_i)\right) - 1 \right\rceil \\
&\geq \left(\sum_{i=0}^{k-1} w(e_i)\right) - \left\lceil \left(\sum_{i=0}^{k-1} w(e_i)\right) - 1 \right\rceil \\
&= 1.
\end{aligned}$$

The last inequality implies that there exists at least one register along p, and therefore the clock period constraint is met. □

Algorithm Approx-Retiming terminates in $O(V^{1/2} E \lg^2(V d_{\max}))$ time. Step 1 completes in $O(E)$ time. Steps 2 and 3 complete in $O(V^{1/2} E \lg^2(V d_{\max}))$ time, since negative edge-weight cycles can be detected in $O(V^{1/2} E \lg(V d_{\max}))$ time using the single-source shortest-paths algorithm by Gabow and Tarjan [12]. Step 4 terminates in $O(E)$ time, and thus the total running time is $O(V^{1/2} E \lg^2(V d_{\max}))$.

1.6 Optimal Pipelining of Combinational Circuitry

In this section, we describe an $O(E \lg d_{\max})$ algorithm for the problem of pipelining combinational circuitry with the minimum possible clock period. Our result improves the $O(VE \lg V)$ bound that can be obtained by applying the general retiming algorithm from [31], and it is optimal within a constant multiplicative factor for unit-delay circuitry. In contrast to the algorithm in [31] that computes all possible clock periods, our algorithm exploits the special structure of a combinational circuit to identify an interval of $d_{\max}$ integers that contains the optimal period.

In a combinational circuit the graph is acyclic and has an input interface v_{in} and an output interface v_{out}. Initially, the circuit is assumed to have no registers. By retiming a combinational circuit G we add registers to the circuit in such a way that the retimed circuit G_r achieves a shorter clock period at the cost of introducing a *latency* of $r(v_{in}) - r(v_{out})$ clock ticks for the signals to propagate from the input interface v_{in} to the output interface v_{out}. The problem of *minimum clock period pipelining* is defined as follows: Given a combinational circuit $G = \langle V, E, d, 0 \rangle$ and a positive integer l, determine a retiming r such that G_r is a

MPP(G, l)
1 Compute the delay Δ of the longest path p_Δ in G from v_{in} to v_{out}.
2 Use Algorithm MLP to binary search $\left[\left\lceil \frac{\Delta}{l+1} \right\rceil, \left\lceil \frac{\Delta}{l+1} \right\rceil + d_{\max}\right]$ for the minimum integer period achievable with latency at most l.

Figure 1-7: Algorithm MPP for minimum period pipelining. Given a combinational circuit $G = \langle V, E, d, 0\rangle$ with input interface v_{in} and output interface v_{out}, and a positive integer l, this algorithm determines a retiming r such that G_r is a pipelined combinational circuit with latency l and minimum clock period.

pipelined combinational circuit with latency at most l and with minimum clock period.

Our Algorithm MPP for minimum clock period pipelining of combinational circuitry is illustrated in Figure 1-7. Step 1 of the algorithm computes the delay Δ of the longest path $v_{in} \rightsquigarrow v_{out}$ in $O(E)$ steps by traversing the vertices of G in topological sort order [5]. Step 2 binary searches an interval of $d_{\max} + 1$ integers for the minimum achievable clock period $\Phi_{\min}(G)$. In each iteration of the search, the $O(E)$-time Algorithm MLP generates a pipelined circuit that achieves the clock period under consideration with the minimum possible latency. The search ends when the shortest period that can be achieved with latency at most l has been identified. Thus, Algorithm MPP terminates in $O(E \lg d_{\max})$ steps.

The correctness of Algorithm MPP follows directly from Theorem 11 that bounds the minimum achievable clock period $\Phi_{\min}(G)$ and the correctness of Algorithm MLP for minimum latency pipelining. First, we prove the bounds on $\Phi_{\min}(G)$. The proof relies on the constraint graph G_c, which in this case has been augmented by an edge $v_{out} \to v_{in}$ of weight l in order to account for the desired latency of the circuit.

Theorem 11 *Let $G = \langle V, E, d, 0\rangle$ be a combinational circuit with input interface v_{in} and output interface v_{out}. Let Δ be the delay of the path $p_\Delta = \stackrel{*}{\rightarrow} v_{in} v_{out}$ in G with the longest propagation delay, and let l be a positive integer. Then the minimum clock period $\Phi_{\min}(G)$ for any pipelined version of G with latency l satisfies:*

$$\left\lceil \frac{\Delta}{l+1} \right\rceil \le \Phi_{\min}(G) \le \left\lceil \frac{\Delta}{l+1} \right\rceil + d_{\max},$$

where $d_{\max} = \max\{d(v) : v \in V\}$.

Proof. According to Theorem 1, every retiming r that yields a pipelined version of the original circuit with clock-period at most c must satisfy

$$r(v) - r(u) \le 0 \tag{1.5}$$

for every edge $u \to v$ in E, and

$$r(v) - r(u) \le -1 \tag{1.6}$$

for all vertices $u, v \in V$ connected by a path $p \in G$ with delay $d(p) > c$. In addition, it must satisfy a latency constraint

$$r(v_{in}) - r(v_{out}) \leq l, \tag{1.7}$$

where l is an upper bound on the latency of the pipelined circuit. Inequalities (1.5), (1.6), and (1.7) induce a constraint graph G_c that is described in the statement of Theorem 2. Note that Inequality (1.7)) introduces an edge $v_{out} \xrightarrow{e_l} v_{in}$ in G_c with weight $w_c(e_l) = l$, and that e_l is included in every cycle in G_c.

First, we derive the lower bound on $\Phi_{\min}(G)$. Let r be a retiming of the circuit with latency l and clock period $\Phi_{\min}(G)$. Adding up the delays of the $l+1$ register-free parts along the path p_Δ yields $\Delta \leq \Phi_{\min}(G)(l+1)$. It follows that $\Phi_{\min}(G) \geq \lceil \Delta/(l+1) \rceil$, since the propagation delay $d(v)$ is an integer for every vertex $v \in V$.

We establish the upper bound on $\Phi_{\min}(G)$ by proving that $\lceil \Delta/(l+1)\rceil + d_{\max}$ is a feasible clock period. According to Lemma 6, it suffices to show that G_c has no negative edge-weight cycles for $c = \lceil \Delta/(l+1)\rceil + d_{\max}$. The only negative-weight edges of G_c have weight -1. The maximum number of such edges in any path from v_{in} to v_{out} is

$$\begin{aligned}
\left\lfloor \frac{\Delta - 1}{(\lceil \Delta/(l+1)\rceil + d_{\max}) - d_{\max}} \right\rfloor &= \left\lfloor \frac{\Delta - 1}{\lceil \Delta/(l+1)\rceil} \right\rfloor \\
&\leq \left\lfloor \frac{\Delta - 1}{\Delta/(l+1)} \right\rfloor \\
&= \left\lfloor l + 1 - \frac{l+1}{\Delta} \right\rfloor \\
&\leq l,
\end{aligned}$$

since $l + 1 > 0$. Every cycle in G_c must use e_l with $w_c(e_l) = l$, and therefore the last inequality implies that G_c has no negative edge-weight cycles. □

Algorithm MPP employs a subroutine for pipelining a combinational circuit to achieve a specified period c using the minimum possible number of stages in the pipeline. In mathematical terms, this subroutine computes a solution r for Inequalities (1.5) and (1.6), such that the latency $r(v_{in}) - r(v_{out})$ is minimized. According to Theorem 2, a solution r is given by the lengths of the single-source shortest-paths in the possibly dense constraint graph $G_c = \langle V, E_c, w_c \rangle$ that is induced by Inequalities (1.5) and (1.6).

Algorithm MLP, which is described in Figure 1-8, finds a minimum latency pipelining of a combinational circuit by a single sweep over its circuit graph G. The intuitive idea behind the algorithm is to visit the vertices of G while keeping track of the longest propagation delay $\delta(v)$ along any combinational path $u \overset{p}{\rightsquigarrow} v$ in G up to the currently visited vertex v. New registers are inserted greedily: whenever $\delta(v)$ exceeds the desired clock period c, a pipeline stage is introduced. Algorithm MLP terminates in $O(E)$ steps, since each edge is examined twice, and as we prove in the following lemma it returns a correct answer.

Lemma 12 *The assignment r computed by Algorithm* MLP *is a solution to the single-source shortest-paths problem on the constraint graph G_c defined by Inequalities (1.5) and (1.6).*

Proof. We prove that $r(v)$ gives the length of the shortest path $v_{in} \rightsquigarrow v$ in the constraint graph $G_c = (V, E_c, w_c)$, and that $\delta(v)$ gives the maximum propagation delay among all

MLP(G, c)
1 **for** each vertex $v \in V$
2   **do** $r(v) \leftarrow 0$
3         $\delta(v) \leftarrow d(v)$
4 **for** each vertex $v \in V$ in topological sort order
5   **do for** each edge $u \rightarrow v \in E$
6   **do if** $\delta(u) + d(v) > c$
7   **then** $r(v) \leftarrow \min\{r(v), r(u) - 1\}$
8   **else** $r(v) \leftarrow \min\{r(v), r(u)\}$
9   **for** each edge $u \rightarrow v \in E$
10   **do if** $r(v) = r(u)$
11   **then** $\delta(v) \leftarrow \max\{\delta(v), \delta(u) + d(v)\}$

Figure 1-8: Algorithm MLP for minimum latency pipelining. Given a combinational circuit $G = \langle V, E, d, 0\rangle$ and a desired clock period c, this algorithm determines a pipelined combinational circuit G_r with clock period $\Phi(G_r) \le c$ and minimum latency.

paths $u \rightsquigarrow v$ in G with $r(u) = r(v)$. The proof is by induction on the number of vertices that have been visited by the algorithm.

For the basis, the input interface v_{in} is visited, and we have $r(v_{in}) = 0$ and $\delta(v_{in}) = d(v)$ after initialization. Therefore, the lemma holds.

For the inductive step, vertex v is visited after all preceding vertices u have been visited. We assume that

$$r(u) = \min\{w_c(p) : v_{in} \overset{p}{\rightsquigarrow} u \in G_c\},$$

and

$$\delta(u) = \max\{d(p) : u' \in V, u' \overset{p}{\rightsquigarrow} u \in G, r(u') = r(u)\}$$

for every vertex u preceding v in the topological order. Lines 5-8 perform a relaxation over all edges $u \rightarrow v$ in E and set

$$\begin{aligned} r(v) &= \min\{\{r(u) - 1 : \delta(u) + d(v) > c, u \rightarrow v \in E\} \cup \{r(u) : \delta(u) + d(v) \le c, u \rightarrow v \in E\}\} \\ &= \min\{w_c(p) : v_{in} \overset{p}{\rightsquigarrow} v \in G_c\}, \end{aligned}$$

since $\delta(u) + d(v) > c$ implies that there exists an edge $u' \overset{e}{\rightarrow} v \in E_c$ such that $r(u') = r(u)$ and $w_c(e) = -1$. Lines 9-11 set

$$\begin{aligned} \delta(v) &= d(v) + \max\{\delta(u) : u \rightarrow v \in E, r(u) = r(v)\} \\ &= \max\{d(p) : u' \in V, u' \overset{p}{\rightsquigarrow} v \in G, r(u') = r(v)\}, \end{aligned}$$

and therefore the lemma holds. □

1.7 Conclusion

In this chapter we presented tight bounds on the minimum clock period that can be achieved by retiming an edge-triggered circuit. We expressed these bounds in terms of the maximum delay-to-register ratio around the cycles in the circuit and the maximum propagation delay $d_{\max}$ of the combinational logic blocks in the circuit. Using these bounds, we characterized exactly the minimum clock period that can be achieved by retiming unit-delay circuits, and we designed improved algorithms for several problems related to retiming. Specifically, we presented an $O(V^{1/2}E \lg V)$-time algorithm for optimal retiming of unit-delay circuits. This is the asymptotically fastest algorithm known to date for this problem. For circuits that also include combinational logic blocks with non-unit delays, we gave an $O(VE \lg d_{\max})$-time algorithm for optimal retiming and an $O(V^{1/2}E \lg^2(Vd_{\max}))$-time algorithm for retiming with a clock period that does not exceed the optimal by more than $d_{\max} - 1$. Finally, we gave an $O(E \lg d_{\max})$-time algorithm for optimal pipelining of combinational circuitry.

An interesting open problem is the design of an algorithm for optimal retiming of circuits that include non-unit delay combinational logic blocks whose running time matches that of Algorithm UD-Retiming for optimal retiming of unit-delay circuits.

Chapter 2

Analyzing and Optimizing Level-Clocked Circuitry

2.1 Introduction

A VLSI designer often has the choice of whether to use level-clocked latches or the more conventional edge-triggered latches to implement clocked storage elements in his circuit. An edge-triggered latch directly supports the abstraction of a storage element that is synchronized by the ticking of a clock. When the clocking waveform rises (*i.e.*, the clock ticks), an edge-triggered latch instantaneously samples its input and asserts that value on its output.

A level-clocked latch operates somewhat differently. While the clock input to a level-clocked latch is low, the latch output maintains its value from the most recent time that the clock was high. While the clock is high, however, the input flows unimpeded to the output, unsynchronized with either edge of the clock. In order to avoid problems with race conditions, it is common for level-clocked circuits to adopt clocking disciplines which involve multiple clock waveforms, or "phases".

In a *two-phase clocking scheme* [35], two clocking waveforms, or *phases*, denoted ϕ_0 and ϕ_1, are employed, as is shown in Figure 2-1. Formally, we denote a two-phase clocking scheme by a 4-tuple $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ of strictly positive real numbers. In this context, ϕ_0 denotes the *duty cycle* of the first phase, *i.e.*, the length of time during which the phase is high, while γ_0 denotes the *gap* of the first phase, *i.e.*, the amount of time between a falling edge of the first phase and the next rising edge of the second phase, which generally must be long enough to overcome various engineering constraints, such as setup and hold times, the nonzero durations required for clock signals to rise and fall, and clock skew [14, 58]. The duty cycle and gap of the second phase are, similarly, denoted by ϕ_1 and γ_1, respectively. The ratio $\rho = \phi_1/\phi_0$ is the *duty ratio* of the clocking scheme. We overload the symbol π to denote the sum

$$\pi = \phi_0 + \gamma_0 + \phi_1 + \gamma_1, \tag{2.1}$$

Parts of this chapter represent joint research with Alex Ishii and Charles Leiserson.

which is the *period* of the clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$. We also overload ϕ_i to denote both phase i and its corresponding duty cycle, and γ_i to denote both gap i and its corresponding duration. Observe, that since the values γ_0 and γ_1 are strictly greater than 0, phases ϕ_0 and ϕ_1 are nonoverlapping (not simultaneously high).

In this chapter we give an efficient algorithm to verify the proper timing of circuits that employ two-phase clocking schemes, and we present several other algorithms for optimizing their clock periods. Since an edge-triggered latch can be implemented by two back-to-back level-clocked latches [14] our algorithms also provide an automatic way to take edge-triggered circuits and transform them into faster level-clocked ones. At the end of this chapter, we generalize most of our algorithms to level-clocked circuits that employ more than two clock phases.

We model a circuit as an edge-weighted, vertex-weighted multigraph $G = \langle V, E \rangle$ in which V is a collection of combinational logic elements with associated propagation delays and E is the set of interconnections, each of which passes through zero or more latches. Each latch is clocked either by ϕ_0 or ϕ_1. A general framework for the timing verification of level-clocked circuits appears in [19, 20].

An example of a two-phase, level-clocked circuit is shown in Figure 2-2(a). The integers in the vertices signify propagation delays. For simplicity, let us assume that in the figure, we have $\gamma_0 = \gamma_1 = 0$. (In our mathematical development, we shall assume that the gaps γ_0 and γ_1 are strictly positive, as is consistent with engineering situations, and because the assumption of 0 gaps can raise some subtle, but tedious and largely irrelevant, difficulties.) If the two phases ϕ_0 and ϕ_1 have equal duty cycles, then the circuit cannot be clocked with a clock period shorter than 36 units of time, since the path DEA has propagation delay 54, and intuitively, a datum has at most 3/2 clock periods to propagate from the latch preceding D to the latch succeeding A.

The first problem that we consider in this chapter is the *timing verification problem* for two-phase circuits. We give an $O(VE)$-time algorithm that verifies whether a level-clocked circuit is properly timed by a given two-phase clocking scheme. This result improves the $O(E^2)$ bound obtained when the general algorithm from [20] is applied to the special case of two-phase, level-clocked circuits. (The bound in [20] is also $O(VE)$, but the circuit model used in that paper represented both functional elements and latches as vertices, and interconnections between them as edges. Translating to the model presented here yields the $O(E^2)$ bound. The algorithm in [20] applies to more general circuits and timing

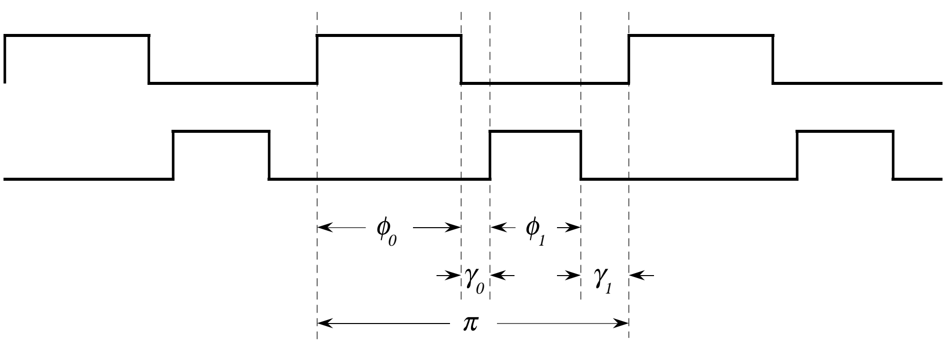


Figure 2-1: A two-phase, nonoverlapping clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$.

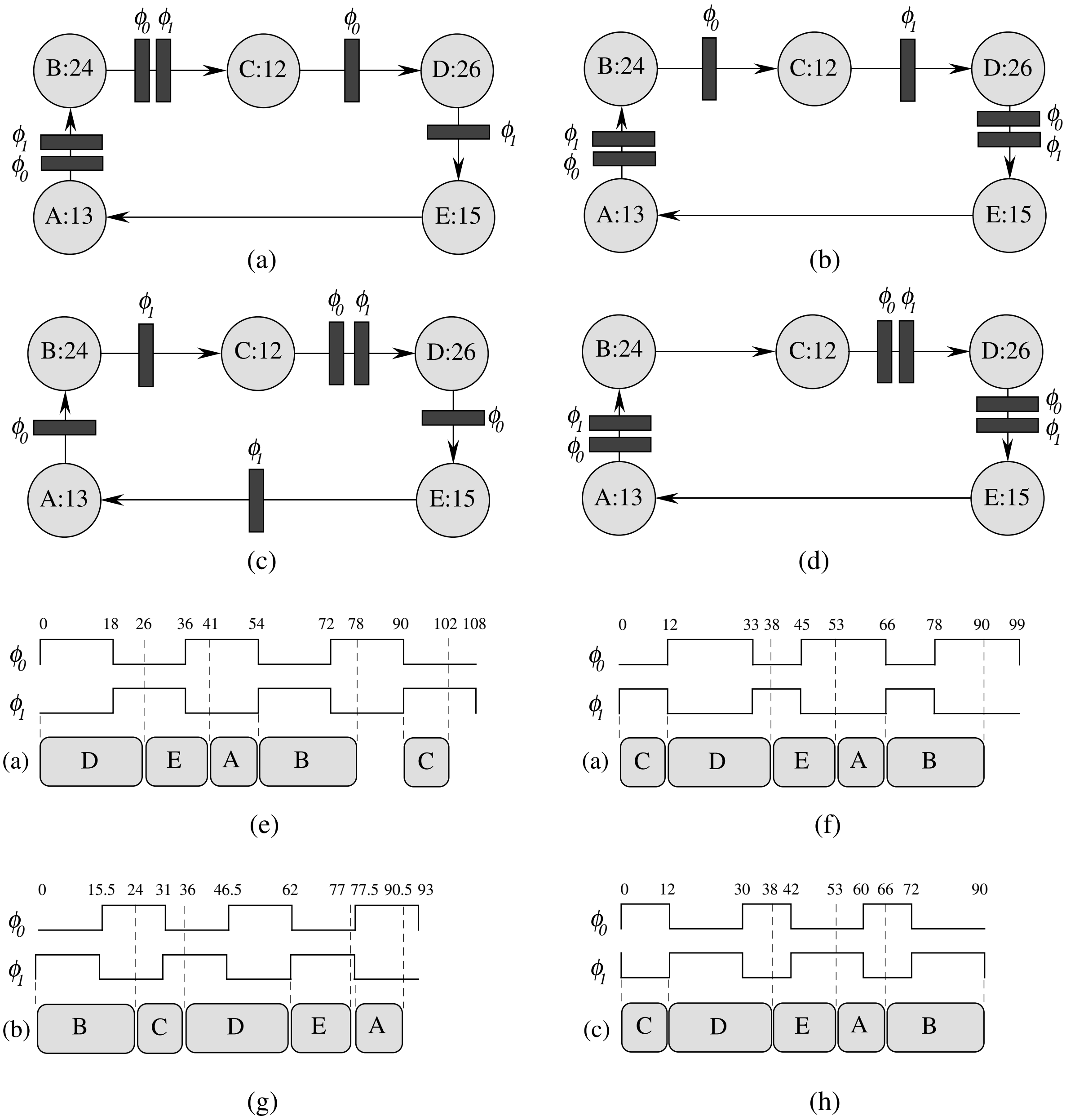


Figure 2-2: An illustration of the various techniques for optimizing two-phase, level-clocked circuits. Part (a) of the figure shows a simple level-clocked circuit. When the duty cycles of the phases ϕ_0 and ϕ_1 are equal, as is shown in part (e) of the figure, the clock period of circuit (a) cannot be made smaller than 36. By tuning the clocking scheme of circuit (a) to have a duty ratio of 4:7, as shown in part (f), a clock period of 33 can be achieved. Part (b) of the figure shows a retimed version of circuit (a). When clocked with the symmetric clocking scheme shown in part (g), circuit (b) achieves a clock period of 31, which is optimal for symmetric clocking schemes under any retiming. The optimal combination of retiming and tuning for circuit (a) is shown in part (c). When clocked with the waveforms in part (h), which have a duty ratio of 3:2, circuit (c) achieves a clock period of 30, which is the optimal for any combination of clocking scheme and retiming. Part (d) shows a typical level-clocked implementation of an edge-triggered circuit which is equivalent under retiming to the other circuits. Each pair of level-clocked latches in the circuit implements an edge-triggered latch. This edge-triggered circuit has a clock period of 36, which is the best that can be obtained by any retiming that is edge triggered.

methodologies than the ones considered here, however.)

In addition to timing verification, we consider a collection of timing analysis problems for two-phase circuits. The algorithms we provide identify the critical paths of the circuit and provide information on the sensitivity of the circuit's timing to changes in the propagation delays of its combinational logic blocks. For the clocking scheme in Figure 2-2(e), for example, the path DEA in the circuit of Figure 2-2(a) is critical. The propagation delay of B can increase by 12, however, without violating the circuit's proper timing by the clocking scheme in Figure 2-2(e). We give an $O(VE)$-time algorithm for the *noncritical sensitivity analysis problem* in which we wish to compute by how much we can increase the propagation delay of any given block of combinational logic without affecting the proper timing of the circuit by a given clocking scheme. We also give an $O(VE + V^2 \lg V)$-time algorithm that solves this problem for all combinational logic blocks in the circuit. Another problem we investigate is the *critical sensitivity analysis problem* for combinational logic blocks that lie on critical paths. We present an $O(VE)$-time algorithm that computes the minimum decrease in the propagation delay of any critical block that is required to remove it from the critical path.

Our next result deals with modifying, or "tuning", the clocking scheme of a circuit—that is, providing the circuit with new clocking waveforms. The two clocking schemes in Figures 2-2(e) and 2-2(f), for example, illustrate tuning. Observe, that the circuit shown in Figure 2-2(a) is properly timed by either clocking scheme, but the first clocking scheme has a period of 36, while the second has a clock period of 33. The notion of clock tuning encompasses more than a simple increase in the frequency of the clock. In particular, clock tuning allows also for the adjustment of the duty ratio of the clocking scheme. In the example of Figure 2-2(a), the circuit cannot be properly timed by any clocking scheme whose period is less than 33, since the delay of the path CDEA is 66 and must be distributed over at most two full clock periods. The clocking scheme shown in Figure 2-2(f) is thus an optimal tuning for the circuit in Figure 2-2(a).

The *tuning problem* for two-phase circuits is the problem of adjusting the phases of a clocking scheme so as to clock a given two-phase circuit as quickly as possible. We assume that the gaps γ_0 and γ_1 must be kept fixed and only the duty cycles of the phases can be adjusted. We give an $O(VE)$-time algorithm to solve the tuning problem. Previous algorithms for tuning have either addressed other types of clocking methodologies [9, 49, 57], or been uncharacterized with respect to worst-case running time [6, 51].

Another way to optimize a circuit is by *retiming*: a method for relocating latches within the circuit without affecting the functionality of the circuit. Retiming has been well studied in the context of edge-triggered circuits [28, 29, 31, 33, 45] and has been the subject of study in the context of single-phase, level-clocked circuits [52]. We extend the retiming technique to encompass the optimization of two-phase, level-clocked circuits. We consider three problems related to retiming.

The *retiming problem* for two-phase circuits asks whether, for a given two-phase circuit G and clocking scheme π, the circuit G can be retimed to be properly timed by π. As an example, consider the circuit in Figure 2-2(a). If we retime the circuit to be properly timed by the clocking scheme in Figure 2-2(g), we obtain the circuit in Figure 2-2(b). We provide an algorithm to solve the retiming problem that runs in $O(V^3)$ time.

The *retiming problem with symmetric clocking schemes* is the common special case of the retiming problem in which the two phases of the clocking scheme are identical and 180 degrees out of phase. Such a symmetric clocking scheme, which in general has the form $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$, is shown in Figure 2-2(g). We provide an algorithm that solves the retiming problem for symmetric clocking schemes in $O(VE + V^2 \lg V)$ time.

The *retiming problem for minimum latch count* asks for a retiming of a given circuit G that achieves a given symmetric clocking scheme π with the minimum number of latches. We describe an algorithm that solves this problem in $O(V^3 \lg V)$ time.

We consider three problems related to both retiming and tuning. The first is a general optimization problem, and the other two are progressively more specialized.

The *retiming and tuning problem* asks how a circuit can be retimed and its clock simultaneously tuned to achieve the minimum clock period over all possible clocking schemes. For example, the optimal retiming of the circuit in Figure 2-2(a) yields the circuit in Figure 2-2(c) with the clocking scheme in Figure 2-2(h). We provide an efficient approximation algorithm for this general tuning and retiming problem. For any given relative error $\epsilon > 0$, the approximation algorithm runs in $O(V^3(1/\epsilon)\lg(1/\epsilon) + (VE + V^2 \lg V)\lg(V/\epsilon))$ time and produces a retimed circuit that is properly timed by a clocking scheme whose period is at most $(1+\epsilon)$ times the optimal. We also provide an $O(V^{11})$-time algorithm that solves the general tuning and retiming problem exactly.

The *retiming and fixed-duty-ratio tuning problem* is the special case of the retiming and tuning problem where we ask how a circuit can be retimed and its clock tuned to achieve the minimum clock period over all clocking schemes that have a given duty ratio. We give an algorithm that, for any given relative error $\epsilon > 0$, runs in $O(V^3 \lg(V/\epsilon))$ time. This algorithm can be adapted to solve the *retiming and fixed-duty-cycle tuning problem* in which the duty cycle of one of the two phases of the clock is given and the other can be adjusted. We also give an $O(V^2E + V^3 \lg V)$-time algorithm that solves the retiming and fixed-duty-ratio tuning problem exactly.

When we require the two phases of the clocking scheme to be symmetric, we have a special case of the retiming and fixed-duty-ratio tuning problem that we call the *retiming and symmetric tuning problem*. For example, the optimal retiming of the circuit in Figure 2-2(a) with respect to symmetric clocking schemes yields the circuit in Figure 2-2(b) with the clocking scheme in Figure 2-2(g). We give an approximation scheme for the retiming and symmetric tuning problem that runs in $O((VE+V^2 \lg V)\lg(V/\epsilon))$ time for any relative error $\epsilon > 0$. We also give an exact algorithm for the retiming and symmetric tuning problem that runs in $O(V^2E)$ time.

Our optimization techniques can be used not only to facilitate the design of level-clocked circuits, but also to convert edge-triggered circuits into faster level-clocked circuits. The basis for the conversion is the fact that an edge-triggered latch can be implemented by a pair of level-sensitive latches. In Figure 2-2(d) we illustrate the typical level-clocked implementation of an edge-triggered circuit. This circuit has a clock-period of 36, but the algorithms presented in this chapter can automatically produce the optimal level-clocked circuit in Figure 2-2(c), which, with the clocking scheme in Figure 2-2(h), is timed properly with a clock period of 30. As testimony to the additional power gained by level-clocking, observe that no edge-triggered retiming of the circuit from Figure 2-2(d) improves upon its period of 36.

Most of the algorithms described in this chapter have been implemented in TIM, a software package for two-phase, level-clocked circuitry that we describe in Chapter 3 [48]. We have already used TIM to compare empirically two-phase, level-clocked circuits and edge-triggered circuits in terms of speed and number of storage elements [47]. TIM provides interactive feedback to designers. For example, rather than simply reporting the minimum clock period of a circuit, it performs a "sensitivity analysis" that reports the extent to which noncritical propagation delays can be increased without affecting the clock period.

The remainder of this chapter is organized as follows.

Section 2.2 describes necessary and sufficient conditions for a two-phase, level-clocked circuit to be properly timed. Section 2.3 then simplifies these conditions and uses them in an $O(VE)$-time algorithm that solves the timing verification problem. These conditions are also used by the sensitivity analysis algorithms that we present in Section 2.4. Specifically, we present an $O(VE)$-time algorithm that solves the noncritical sensitivity analysis problem for a single combinational block, and an $O(VE + V^2 \lg V)$-time algorithm that solves the noncritical analysis problem for all combinational blocks. We also give an algorithm that solves the critical analysis problem for a single combinational block in $O(VE)$ time. By viewing the simplified necessary and sufficient conditions as a two-dimensional linear program, Section 2.5 shows how the tuning problem can be solved in $O(VE)$ time.

Sections 2.6 and 2.7 describe how to solve the retiming problem for symmetric and general clocking schemes, respectively. The $O(VE+V^2 \lg V)$-time algorithm for symmetric clocking schemes is based on reducing the retiming problem to an efficiently solvable mixed-integer linear program. The $O(V^3)$-time algorithm for the general case uses a technique we call *integer monotonic programming*. Both algorithms also determine when the given clocking scheme cannot be achieved by any retiming. Section 2.8 presents an $O(V^3 \lg V)$-time algorithm that solves the retiming problem for minimum latch count.

Section 2.9 presents approximation algorithms for the three problems related to retiming and tuning. Specifically, we give an $O((VE + V^2 \lg V) \lg(V/\epsilon))$-time algorithm for the retiming and symmetric tuning problem, and an $O(V^3 \lg(V/\epsilon))$-time algorithm for the retiming and fixed-duty-ratio tuning problem, that achieve the optimal clock period to within any given relative error $\epsilon > 0$. These two algorithms can also be used to obtain the exact optimum in the special case where all propagation delays are integers. For the general retiming and tuning problem, we give an $O(V^3(1/\epsilon) \lg(1/\epsilon) + (VE + V^2 \lg V) \lg(V/\epsilon))$-time approximation algorithm.

Although we solve the three problems related to both retiming and tuning with efficient approximation algorithms, we have also discovered polynomial-time optimal algorithms for them. In Section 2.10 we describe an algorithm that solves the retiming and symmetric tuning problem optimally in $O(V^2E)$ time, the retiming and fixed-duty-ratio tuning problem optimally in $O(V^2E+V^3 \lg V)$ time, and the general retiming and tuning problem optimally in $O(V^{11})$ time. These results are based on a characterization of the feasible clock periods of retimed level-clocked circuits.

Section 2.11 extends many of our techniques to circuits that employ more than two phases. The algorithms for k-phase circuits are generally at most a factor of k slower than the corresponding algorithms for two-phase circuits. Section 2.12 concludes by discussing how our algorithms can be generalized to handle design issues that arise in practice. Appendix A.1 provides a proof that the conditions we give for the proper timing of a two-phase, level-clocked circuit are correct.

In independent work, Lockyear and Ebeling [32] have also obtained algorithms for retiming multiphase, level-clocked circuits. Their results include a polynomial-time algorithm for the symmetric retiming problem. They use this algorithm as a subroutine to solve the retiming and symmetric tuning problem. They also determine a set of constraints for the retiming problem, and they describe a Bellman-Ford-like algorithm for solving the constraints. Algorithms for retiming single-phase, level-clocked circuitry have appeared in [52]. Retiming heuristics were given in [3].

Early versions of our work appear in [21, 46].

2.2 Constraints for Proper Timing

In this section we give necessary and sufficient conditions for a two-phase, level-clocked circuit to be properly timed by a given clocking scheme. The section begins with a formal definition of the set of level-clocked circuits to which our results can be applied. We then precisely characterize the timing constraints that need to be satisfied by a properly timed circuit. These constraints are based on the general formulation from [20], but they are substantially simpler due to the additional structure inherent in two-phase, level-clocked circuits.

Since we represent circuits in terms of graphs, we first define some graph notations. For a directed graph $G = \langle V, E \rangle$, we denote an edge e from a vertex u to a vertex v by $u \xrightarrow{e} v$. If the edge name is unnecessary, we sometimes omit it. A path p from u to v is denoted by $u \overset{p}{\rightsquigarrow} v$. A path contains both its endpoints u and v. We shall use both edge and vertex weights. For an edge-weight function w and path p, the weight $w(p)$ is just the weight of p's constituent edges. For a vertex weight d, the weight $d(p)$ is the weight of the p's constituent vertices, including the weights of its endpoints. A cycle is a path that begins and ends with the same vertex v. For a vertex weight d, the weight $d(c)$ of a cycle c includes the weight of each vertex on the cycle only once.

We formally represent a two-phase, level-clocked circuit as a directed multigraph $G = \langle V, E, d, w, \chi \rangle$, where V is a collection of functional elements, E is the set of interconnections between functional elements, the *(propagation) delay* function d is a mapping from V to the nonnegative real numbers, the edge-weight function w is a mapping form E to the integers, and $\chi : V \to \{0, 1\}$ is an assignment of a *phase* to each functional element. For a two-phase circuit G to be *well formed*, we require

WF1. $w(e) \geq 0$ for all $e \in E$;

WF2. $w(c) > 0$ for every cycle $c \in G$;

WF3. for every edge $u \xrightarrow{e} v \in E$,

$$w(e) - \chi(u) + \chi(v) \equiv 0 \pmod 2 .$$

(The weight of a cycle—or a path—is just the sum of the weights of its constituent edges.)

In our circuit model, a vertex $v \in V$ corresponds to a functional element whose maximum propagation delay is equal to $d(v)$ and whose minimum propagation delay is zero. Level-clocked latches are not represented explicitly but rather, are represented implicitly by the

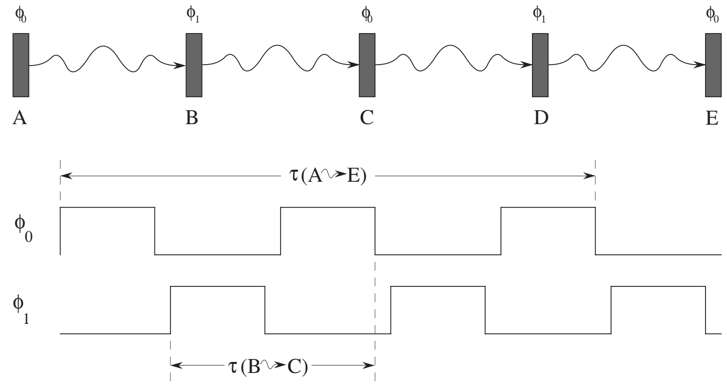


Figure 2-3: The rise-to-fall time for the path $A \leadsto E$ is $\phi_0 + 2(\gamma_0 + \phi_1 + \gamma_1 + \phi_0) = \phi_0 + 2\pi$. The rise-to-fall time for the path $B \leadsto C$ is $\phi_1 + \gamma_1 + \phi_0$. Each of the squiggly lines in the top part of the figure represents a path of zero or greater combinational delay.

weights $w(e)$ of edges $u \xrightarrow{e} v \in E$. A value $w(e) = 0$ indicates that a direct connection exists between the output of the functional element represented by u and an input of the functional element represented by v. A value $w(e) > 0$ indicates that the connection between the two functional elements consists of $w(e)$ level-clocked latches connected in series.

The requirement that $w(c) > 0$ precludes us from considering circuits with unclocked state. Since we represent latches implicitly in terms of weights on edges between functional edges, the phase that clocks a given latch in the circuit modeled by G is determined implicitly as well. If the phase of a vertex v is $\chi(v)$, it means that $\phi_{\chi(v)}$ clocks the last latch on any edge that ends at v. Condition WF3—which by induction generalizes to

WF3′. for every path $u \overset{p}{\leadsto} v$,

$$w(p) - \chi(u) + \chi(v) \equiv 0 \pmod 2$$

—ensures that the latches on any path alternate between ϕ_0 and ϕ_1. Since any cycle c is also a path, its weight must be even, in addition to being positive. Hence, every cycle c in a two-phase circuit must contain at least two latches—one controlled by each of the two phases.

We now turn to the issue of proper timing. Intuitively, a level-clocked circuit is properly timed if whenever a latch holds a value (its clock input is low), it holds the same value as in an identical circuit in which all functional elements have zero propagation delay. This notion of proper timing is "structural", in the sense that we require that a circuit operate correctly regardless of the functions computed by the functional elements. This requirement avoids potential difficulties with computational intractability. The semantics of proper timing are studied in [19]. Ishii and Leiserson [20] give a set of "Δ-constraints" that serve as necessary and sufficient conditions for the proper timing of a general class of level-clocked circuits. For two-phase circuits, this general formulation reduces to a much simpler set of necessary and sufficient conditions.

The conditions for proper timing of a circuit $G = \langle V, E, d, w, \chi \rangle$ are based on considering the operation of G when all propagation delays are 0. Let σ be any path, not necessarily simple, from a latch A to a latch B in the circuit that G represents. The *rise-to-fall time* $\tau(\sigma)$ of a path σ is the time it would take in the circuit for the effect of a rising edge of A's phase to propagate along σ and be stored in B by the falling edge of B's phase. For example, the rise-to-fall time of path $A \rightsquigarrow E$ in Figure 2-3 is $\phi_0 + 2(\gamma_0 + \phi_1 + \gamma_1 + \phi_0) = \phi_0 + 2\pi$. The rise-to-fall time of path $B \rightsquigarrow C$ is $\phi_1 + \gamma_1 + \phi_0$. The following proposition uses rise-to-fall times to give conditions for the proper timing of a circuit. Its proof is given in Appendix A.1.

Proposition 13 *A two-phase, level-clocked circuit is properly timed if and only if for all latches A and B in the circuit, the propagation delay along any path from A to B is no greater than the rise-to-fall time of the path.* $\square$

Given Proposition 13, we can formulate a set of conditions for the proper timing of a two-phase, level-clocked circuit. Specifically, we have the following lemma.

Lemma 14 *Let $G = \langle V, E, d, w, \chi\rangle$ be a circuit that employs a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$. Then G is properly timed by π if and only if for every path $u \stackrel{p}{\rightsquigarrow} v$ in G, we have*

$$d(p) \le \pi\left(\frac{1 + w(p)}{2}\right) + \phi_{1-\chi(v)} \tag{2.2}$$

if $\chi(u) \neq \chi(v)$, and

$$d(p) \le \pi\left(\frac{2 + w(p)}{2}\right) - \gamma_{1-\chi(v)} \tag{2.3}$$

if $\chi(u) = \chi(v)$.

Proof. In order to prove the theorem, we shall make the straightforward extension of our graph notation from our simple model, in which latches are represented implicitly by edge weights, to the underlying circuit, in which latches are represented explicitly.

($\Rightarrow$) The necessity of Inequalities (2.2) and (2.3) follows from Proposition 13. Consider any path $u \stackrel{p}{\rightsquigarrow} v$ in G, and extend it at both ends to produce a path $\sigma = A \rightsquigarrow u \stackrel{p}{\rightsquigarrow} v \rightsquigarrow B$ in the circuit modeled by G, where A and B are latches and the subpaths $A \rightsquigarrow u$ and $v \rightsquigarrow B$ are latch free. Thus, A is clocked by $\phi_{\chi(u)}$ and B is clocked by $\phi_{1-\chi(v)}$. By definition, the number of latches on the path σ is $w(p) + 2$, and the total propagation delay along σ is $d(\sigma) \ge d(p)$. A case analysis now yields the desired result.

If $\chi(u) \neq \chi(v)$, then A and B are both clocked by phase $\phi_{\chi(u)} = \phi_{1-\chi(v)}$ and $w(p)$ is an odd number. Thus, the effect of a rising edge on A's clock input takes $(1 + w(p))/2$ periods to reach B, plus an additional $\phi_{\chi(u)}$ for the time until the falling edge on B's clock input. Hence, we have

$$\begin{aligned}\tau(\sigma) &= \pi\left(\frac{1 + w(p)}{2}\right) + \phi_{\chi(u)} \\ &= \pi\left(\frac{1 + w(p)}{2}\right) + \phi_{1-\chi(v)} \; . \end{aligned} \tag{2.4}$$

The propagation delay along σ is at least $d(p)$, since σ contains p as a subpath. Hence, by Proposition 13, if the circuit is properly timed, we have $d(p) \le \tau(\sigma)$ and the constraint (2.2) holds.

If $\chi(u) = \chi(v)$, then latch A is clocked by phase $\phi_{\chi(u)}$, latch B is clocked by $\phi_{1-\chi(u)}$, and $w(p)$ is an even number. Thus, the effect of a rising edge on A's clock input takes $(2 + w(p))/2$ periods to reach B, minus the gap $\gamma_{1-\chi(v)}$ following the falling edge on B's clock input. Hence, we have

$$\tau(\sigma) = \pi\left(\frac{2 + w(p)}{2}\right) - \gamma_{1-\chi(v)} \ . \tag{2.5}$$

The propagation delay along σ is at least $d(p)$, since σ contains p as a subpath. Hence, by Proposition 13, if the circuit is properly timed, we have $d(p) \leq \tau(\sigma)$ and Inequality (2.3) holds.

($\Leftarrow$) To prove the sufficiency of Inequalities (2.2) and (2.3), assume that the circuit is not properly timed. Then Proposition 13 implies that there exists a latch-to-latch path $A \overset{\sigma}{\rightsquigarrow} B$ in the circuit with propagation delay greater than $\tau(\sigma)$. Without loss of generality, σ has the minimum rise-to-fall time of any such path, and hence, latch A is directly followed by a functional element u and latch B is directly preceded by a functional element v. Thus, $\sigma = A \to u \overset{p}{\rightsquigarrow} v \to B$, and we have $d(p) > \tau(\sigma)$. Using a case analysis similar to the one to prove necessity, we can conclude that either Inequality (2.2) or Inequality (2.3) is violated. □

The reader should note that the lemma requires that *all* paths in the circuit be considered, not just simple ones.

2.3 Verifying Proper Circuit Timing

In this section we consider the *verification problem* for two-phase, level-clocked circuits: Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi\rangle$ and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$, determine whether G is properly timed by π. Figure 2-4 gives an $O(VE)$-time algorithm TV for the timing verification problem. This bound improves the $O(E^2)$ bound that one obtains by applying the general verification algorithm in [20] to two-phase circuits. We analyze the running time of Algorithm TV and then prove its correctness.

We first prove a bound on the running time of Algorithm TV.

Lemma 15 *Algorithm* TV *terminates in $O(VE)$ time.*

Proof. The circuit transformation of Step 1 can be computed in $O(E)$ time. We can implement Step 2 to run in $O(VE)$ time as follows. First, we compute in $O(E)$ time a topological sort [5, Section 23.4] of all edges $e \in E$ with $w(e) = 0$. We then execute a doubly nested loop, where the outer loop is indexed by i, and the inner loop is indexed by each $e \in E$ consistent with the topological sort order if $w(e) = 0$ and in any order if $w(e) > 0$. Within the doubly nested loop, we maintain $D(v, i)$ as a running maximum of the right-hand side of the equation in Step 2 for each v, where $u \xrightarrow{e} v$. The order of execution guarantees that all right-hand side terms are computed before they are used. Since i takes on $O(V)$ values and e takes on $|E|$ values, the entire doubly nested loop runs in $O(VE)$ time. Step 3 checks $O(V^2)$ constraints in $O(V^2)$ time. Finally, Step 4 can be performed in $O(VE)$ time using the Bellman-Ford algorithm [5, Section 25.3]. Thus, the total running time of Algorithm TV is $O(VE)$. □

TV(G, π)

1. Modify G by replacing $w(e)$ on each edge $e \in E$ with $(w(e) \bmod 2) + 2$ if $w(e) \geq 4$.
2. Compute $D(v, i)$ for all $v \in V$ and $i = 0, 1, 2, \ldots, 3|V| - 3$ from the recurrence

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

3. Check the following constraints for each vertex $v \in V$ and $i = 0, 1, \ldots, 3|V| - 3$:

$$D(v,i) \leq \pi\left(\frac{1+i}{2}\right) + \phi_{1-\chi(v)} \quad \text{if } i \text{ is odd;}$$

$$D(v,i) \leq \pi\left(\frac{2+i}{2}\right) - \gamma_{1-\chi(v)} \quad \text{if } i \text{ is even.}$$

   If any constraint is violated, then return **no**.
4. If the graph G with weight $w(e) - 2d(u)/\pi$ on each edge $u \xrightarrow{e} v$ has a negative-weight cycle, then return **no**; otherwise, return **yes**.

Figure 2-4: Pseudocode for Algorithm TV, which tests whether a circuit $G = \langle V, E, d, w, \chi\rangle$ is properly timed under a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$. The algorithm returns **yes** if the circuit is properly timed and **no** otherwise.

We prove the correctness of Algorithm TV in three lemmas.

We first show that the transformation of G in Step 1 yields a new circuit G' with at most 3 latches per wire and such that G is properly timed if and only if G' is properly timed. The new circuit G' does not in general compute the same function as G. The reason for performing the transformation is that it allows us to use an upper bound of $3|V| - 3$ on index i in Steps 2 and 3. Without this transformation, the algorithm could be made to work by letting i range as high as the number of latches on the longest simple path in the circuit. Performing this transformation results in a more efficient verification algorithm.

Lemma 16 *Let $G = \langle V, E, d, w, \chi\rangle$ be a circuit, let $\hat{e} \in E$ be an edge with $w(\hat{e}) \geq 4$, and let $G' = \langle V, E, d, w', \chi\rangle$ be the circuit obtained by replacing the $w(e)$ latches on each edge $e \in E$ with $w'(e)$ latches, where*

$$w'(e) = \begin{cases} w(e) - 2 & \text{if } e = \hat{e} \text{ ,} \\ w(e) & \text{if otherwise .} \end{cases}$$

Then for any clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$, G is properly timed by π if and only if G' is properly timed by π.

Proof. To begin, we argue that G' is a well-formed two-phase circuit satisfying Conditions WF1, WF2, and WF3. First, we have $w'(\hat{e}) \geq 0$, since $w(e) \geq 4$. Second, the weight of any cycle remains positive because $w(\hat{e}) > 0$. Finally, Condition WF3 continues to hold because

$$w(\hat{e}) \equiv w(e) \pmod 2 .$$

($\Leftarrow$) We now show that if G' is properly timed by π, then so is G. Assume G' is properly timed. By Lemma 14, Inequalities (2.2) and (2.3) hold for the weight function w'. But since

$w(e) \geq w'(e)$ for all $e \in E$, the right-hand sides of these inequalities remain the same or are larger for the weight function w. Since the left-hand sides are the same, the inequalities must hold for circuit G.

($\Rightarrow$) To prove the converse, we first prove the following claim. Let $p = u_1 \overset{p_1}{\rightsquigarrow} v_1 \xrightarrow{e} u_2 \overset{p_2}{\rightsquigarrow} v_2$ be a path in a circuit that goes through an edge $e \in E$ such that $w(e) \geq 2$, and suppose that p_1 and p_2 both satisfy Inequalities (2.2) and (2.3). Then, we claim that p satisfies both inequalities.

The proof of the claim is a case analysis that depends on all possible assignments of phases to u_1, v_1, u_2, and v_2. Let us consider, for example, the situation where $\chi(u_1) = \chi(u_2) = \chi(v_2) = 0$ and $\chi(v_1) = 1$; the other cases are similar. From Lemma 14, we have

$$\begin{aligned} d(p_1) &\leq \left(\frac{2 + w(p_1)}{2}\right)\pi - \gamma_1 \ , \\ d(p_2) &\leq \left(\frac{1 + w(p_2)}{2}\right)\pi + \phi_1 \ . \end{aligned}$$

Adding these inequalities and using the fact that $w(e) \geq 2$, we obtain

$$\begin{aligned} d(p) &= d(p_1) + d(p_2) \\ &\leq \left(\frac{w(p_1) + w(p_2) + 3}{2}\right)\pi + \phi_1 - \gamma_1 \\ &\leq \left(\frac{w(p_1) + w(e) + w(p_2) + 1}{2}\right)\pi + \phi_1 - \gamma_1 \\ &\leq \left(\frac{1 + w(p)}{2}\right)\pi + \phi_1 \ , \end{aligned}$$

thus proving the claim.

Suppose, now, that G is properly timed by π, but that G' is not. Then there must exist a path in G' that violates Inequality (2.2) or Inequality (2.3) with respect to the weight function w' but not with respect to the weight function w. Let p be such a path with the fewest edges. Path p must pass through edge $\hat{e}$, since otherwise the inequalities for w and w' are identical. But then, since $w'(\hat{e}) \geq 2$, the claim we have just proved applies to p. Consequently, a subpath of p must violate one of the two inequalities with respect to w', which contradicts the supposition that p is the shortest such path. This contradiction completes the proof. □

Corollary 17 *Let $G = \langle V, E, d, w, \chi\rangle$ be a circuit, and let $G' = \langle V, E, d, w', \chi\rangle$ be the circuit obtained by replacing the $w(e)$ latches on each edge $e \in E$ with $w'(e) \leq 3$ latches, where*

$$w'(e) = \begin{cases} w(e) & \text{if } w(e) \leq 3 \ , \\ (w(e) \bmod 2) + 2 & \text{if } w(e) \geq 4 \ . \end{cases}$$

Then for any clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$, G is properly timed by π if and only if G' is properly timed by π.

Proof. For any edge $e \in V$ for which $w(e) \geq 4$, we have $(w(e) \bmod 2) + 2 = w(e) - 2c$, where c is the largest integer such that $w(e) - 2c \geq 2$. Thus, by Lemma 16, we can apply the transformation described in the lemma c times to reduce the number of latches on any given edge e for which $w(e) \geq 4$ to $(w(e) \bmod 2) + 2 \leq 3$. Moreover, we can do the same for all such edges. $\square$

The following lemma is used to justify Steps 2 and 3 of Algorithm TV. It shows that the infinitely many path constraints (2.2) and (2.3) describing the conditions for proper circuit timing are equivalent to a finite set of inequalities corresponding to simple paths and simple cycles in the circuit. The cycle inequalities motivate the computation of the maximum delay-to-latch ratio $R(G)$ in the algorithm, while the path inequalities motivate the computation of the various $D(v, i)$ values.

Lemma 18 *Let $G = \langle V, E, d, w, \chi \rangle$ be a circuit that employs a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$. Then G is properly timed by π if and only if for every* simple path *$u \overset{p}{\rightsquigarrow} v$, we have*

$$d(p) \leq \pi\left(\frac{1 + w(p)}{2}\right) + \phi_{1-\chi(v)} \ , \tag{2.6}$$

if $\chi(u) \neq \chi(v)$, and

$$d(p) \leq \pi\left(\frac{2 + w(p)}{2}\right) - \gamma_{1-\chi(v)} \ , \tag{2.7}$$

if $\chi(u) = \chi(v)$; and for every simple cycle c, we have

$$d(p) \leq \pi\left(\frac{w(c)}{2}\right) \ . \tag{2.8}$$

Proof. We show that the inequalities given in Lemma 14 are equivalent to Inequalities (2.6), (2.7), and (2.8).

Suppose first that these three inequalities follow from Inequalities (2.2) and (2.3) from Lemma 14. Since any simple path is a path, Inequalities (2.6) and (2.7) follow immediately from Inequalities (2.2) and (2.3). Now, suppose that the remaining Inequality (2.8) is violated for some cycle c. Let $u \xrightarrow{e} v$ be an edge on c, and consider a path that goes from u to v along e and then k times around the cycle c, finally ending at v, for some $k > 0$. Assume that $\chi(u) \neq \chi(v)$; the situation when $\chi(u) = \chi(v)$ is similar. Then by Inequality (2.2), we have

$$d(u) + k \cdot d(c) + d(v) \leq \pi\left(\frac{1 + w(e) + k \cdot w(c)}{2}\right) + \phi_{1-\chi(v)} \ .$$

Rewriting this inequality, we obtain

$$k\left(d(c) - \pi\frac{w(c)}{2}\right) \leq \pi\left(\frac{1 + w(e)}{2}\right) + \phi_{1-\chi(v)} - d(u) - d(v) \ . \tag{2.9}$$

But, since Inequality (2.8) is violated for c, we have $d(c) > \pi(w(c)/2)$, which means that the left-hand side of Inequality (2.9) can be made to exceed the right-hand side by choosing k sufficiently large. This contradiction proves that Inequality (2.8) holds.

We now prove that Inequalities (2.6), (2.7), and (2.8) imply Inequalities (2.2) and (2.3). Suppose Inequalities (2.6), (2.7), and (2.8) hold, and let $u \overset{p}{\rightsquigarrow} v$ be a path with the minimum number of edges that violates Inequality (2.2). (The proof for Inequality (2.3) is similar.)

Thus, we have

$$d(p) > \pi\left(\frac{1+w(p)}{2}\right) + \phi_{1-\chi(v)} \ .$$

The path p cannot be simple, because then it would violate Inequality (2.6) or Inequality (2.7), and thus it must contain a simple cycle c.

Consider the path p' derived from p by removing the cycle c. The propagation delay of p' is $d(p') = d(p) - d(c)$, and the number of latches on p' is $w(p') = w(p) - w(c)$. Using Inequality (2.8), we have

$$\begin{aligned} d(p') &= d(p) - d(c) \\ &> \pi\left(\frac{1+w(p)}{2}\right) + \phi_{1-\chi(v)} - d(c) \\ &\geq \pi\left(\frac{1+w(p)}{2}\right) + \phi_{1-\chi(v)} - \pi\left(\frac{w(c)}{2}\right) \\ &= \pi\left(\frac{1+w(p)-w(c)}{2}\right) + \phi_{1-\chi(v)} \\ &= \pi\left(\frac{1+w(p')}{2}\right) + \phi_{1-\chi(v)} \ . \end{aligned}$$

Thus, p' also violates Inequality (2.2), and p' has fewer edges than p, contradicting the minimality of p. □

Lemma 18 shows that the infinitely many constraints of Lemma 14 can be summarized by a finite set of constraints, but in general, the number of these constraints can still be exponential in the size (number of vertices and edges) of the circuit. The path delays $D(v,i)$ computed in Step 2 of Algorithm TV allow us to further reduce the number of constraints to $O(V^2)$, as is shown by the following lemma.

Lemma 19 *Let $G = \langle V, E, d, w, \chi\rangle$ be a circuit that employs a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$, and let W be an upper bound on the maximum number of latches on any simple path in G. For $v \in V$ and $i = 0, 1, \ldots, W$, let*

$$D(v,i) = \max\{d(p) : u \overset{p}{\rightsquigarrow} v \text{ and } w(p) = i\} \ . \tag{2.10}$$

Then G is properly timed by π if and only if for every vertex $v \in V$ and $i = 0, 1, \ldots, W$, we have

$$D(v,i) \leq \pi\left(\frac{1+i}{2}\right) + \phi_{1-\chi(v)} \tag{2.11}$$

if i is odd, and

$$D(v,i) \leq \pi\left(\frac{2+i}{2}\right) - \gamma_{1-\chi(v)} \tag{2.12}$$

if i is even; and for every simple cycle c, Inequality (2.8)

$$d(c) \leq \pi\left(\frac{w(c)}{2}\right)$$

holds.

Proof. ($\Rightarrow$) First, we assume that G is properly timed and show that Inequalities (2.11), (2.12), and (2.8) are satisfied. If i is odd, then by Condition WF3$'$, we have $\chi(u) \neq \chi(v)$; and thus by Inequality (2.2), we have

$$\begin{aligned} D(v,i) &= d(p) \\ &\leq \pi\left(\frac{1+w(p)}{2}\right) + \phi_{1-\chi(v)} \\ &= \pi\left(\frac{1+i}{2}\right) + \phi_{1-\chi(v)} \ , \end{aligned}$$

and hence Inequality (2.11) holds. Similarly, if i is even, we can prove that Inequality (2.12) holds. The statement of Lemma 18 states that Inequality (2.8) holds, and thus all three inequalities are satisfied.

($\Leftarrow$) To show the other direction of the lemma, assume that Inequalities (2.11), (2.12), and (2.8) are satisfied. We must prove that G is properly timed. Let $u \overset{p}{\rightsquigarrow} v$ be any simple path in G with $w(p)$ latches. By the definition (2.10), we have $d(p) \leq D(v, w(p)) = D(v,i)$ for some $i \leq W$, since p is simple. If $\chi(u) \neq \chi(v)$, then by Condition WF3$'$, the value i is odd, which means

$$\begin{aligned} d(p) &\leq D(v,i) \\ &\leq \pi\left(\frac{1+i}{2}\right) + \phi_{1-\chi(v)} \\ &= \pi\left(\frac{1+w(p)}{2}\right) + \phi_{1-\chi(v)} \ , \end{aligned}$$

and hence Inequality (2.6) holds. Similarly, if $\chi(u) = \chi(v)$, we can conclude that Inequality (2.7) holds. Since Inequality (2.8) holds by assumption, by Lemma 18, G is properly timed by π. □

We are now able to prove the correctness of Algorithm TV.

Theorem 20 *Algorithm* TV *solves the timing verification problem for a two-phase, level-clocked circuit* $G = \langle V, E, d, w, \chi\rangle$ *and a clocking scheme* $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$ *in* $O(VE)$ *time.*

Proof. Corollary 17 shows that it suffices to verify the circuit as modified by Step 1 of Algorithm TV. Moreover, the transformation results in at most 3 latches per edge of the circuit, and hence, the number of latches on any simple path in the circuit is at most $3|V| - 3$. A simple inductive argument shows that the recurrence in Step 2 computes the values $D(v,i)$ as defined by Equation (2.10), if we choose $W = 3|V| - 3$.

The remainder of the algorithm tests whether the constraints for proper timing from Lemma 19 are satisfied. Inequalities (2.11) and (2.12) are checked directly by Step 3. Inequality (2.8) is checked indirectly by Step 4, which can be seen as follows. Sum the edge weights around a cycle. If the cycle has a negative weight, then Inequality (2.8) is violated. Conversely, if there is no negative-weight cycle, then every cycle in the graph satisfies the inequality.

The running time bound follows from Lemma 15. □

NCSA(G, π, u)

1 Compute single-source shortest-paths lengths $l(v)$ from source u on graph G with edge-length $w'(e) = \lfloor w(e)/2 \rfloor \pi + (w(e) \bmod 2) \left(\gamma_{\chi(x)} + \phi_{1-\chi(x)}\right) - d(y)$ for each edge $x \xrightarrow{e} y \in E$.
2 Compute single-destination shortest-paths lengths $l'(v)$ to sink u on graph G with edge-length $w'(e) = \lfloor w(e)/2 \rfloor \pi + (w(e) \bmod 2) \left(\gamma_{\chi(x)} + \phi_{1-\chi(x)}\right) - d(y)$ for each edge $x \xrightarrow{e} y \in E$.
3 $\delta(u) \leftarrow \min\left\{l(v) + w'(e) : v \xrightarrow{e} u \in E\right\}$
4 $\delta(u) = \min\left\{\delta(u), \min_{v \in V}\left\{\phi_{\chi(v)} - d(v) + l'(v)\right\} + \min_{v \in V}\left\{l(v) + \gamma_{\chi(v)} + \phi_{1-\chi(v)}\right\}\right\}$
5 **if** $\delta(u) < 0$
6   **then return fail**
7   **else return** $\delta(u)$.

Figure 2-5: Algorithm NCSA for the noncritical sensitivity analysis problem. The algorithm takes as input a two-phase circuit G, a clocking scheme π, and a vertex u. It produces as output the maximum increase $\delta(u)$ in the delay $d(u)$ that will not affect the proper timing of G by the clocking scheme π.

2.4 Sensitivity Analysis

In this section we consider two *sensitivity analysis problems* for two-phase, level-clocked circuits. This analysis identifies timing bottlenecks in the circuit as well as parts of the circuit that have potential for further optimization.

The first problem we consider is the *noncritical sensitivity analysis problem*: Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$ and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ such that G is properly timed by π, determine for any given combinational block $u \in V$ the maximum possible increase $\delta(u)$ in its propagation delay $d(u)$ such that G is still properly timed by π.

The second problem we consider is the *critical sensitivity analysis problem*: Given a two-phase level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$ and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$, determine for any given combinational block $u \in V$ the minimum possible $\delta(u)$ such that G is properly timed by π when the propagation delay $d(u)$ decreases by $\delta(u)$.

2.4.1 Noncritical Sensitivity Analysis

In this section we present two algorithms for the noncritical sensitivity analysis problem. The first algorithm solves this problem for a single combinational block in the circuit in $O(VE)$ time. The second algorithm solves the noncritical sensitivity analysis problem for all combinational blocks in the circuit in $O(VE + V^2 \lg V)$ time.

Algorithm NCSA, shown in Figure 2-5, solves the noncritical sensitivity analysis problem for a single combinational block $u \in V$ in $O(VE)$ time. The algorithm computes the maximum increase $\delta(u)$ in the propagation delay $d(v)$ such that Inequalities (2.6), (2.7), and (2.8) that describe a properly timed circuit are not violated. The slack of Inequality (2.8) is computed in Step 3 after a single-source shortest-paths computation with source v and a single-destination shortest-paths computation with sink v on the graph G with edge-length $w'(e) = \lfloor w(e)/2 \rfloor \pi + (w(e) \bmod 2) \left(\gamma_{\chi(x)} + \phi_{1-\chi(x)}\right) - d(y)$ for each edge $x \xrightarrow{e} y \in E$. This edge-length accounts automatically for the slack between the rise-to-fall time and the propagation delay along any path. The slack of Inequalities (2.6) and (2.7) is computed in Step 4. In this step, the algorithm computes the simple path p through u with minimum $\tau(p) - d(p)$, where $\tau(p)$ is the rise-to-fall time of p. The algorithm returns the maximum value $\delta(u) \geq 0$, such that for $d(u) \leftarrow d(u) + \delta(u)$, the circuit G is still properly timed by the given π. If $\delta(u)$ is negative, then G is not properly timed by π, and in this case the algorithm fails.

The following lemma proves a bound on the running of Algorithm NCSA.

Lemma 21 *Algorithm* NCSA *terminates in $O(VE)$ time.*

Proof. The single-source shortest-paths problem in Step 1 and the single-destination shortest-paths problem in Step 2 can be solved in $O(VE)$ time using the Bellman-Ford algorithm [5]. The minimization in Step 3 requires $O(V)$ time, and the minimization in Step 4 requires $O(V^2)$ time. Therefore, the total running time of the algorithm is $O(VE)$. □

The following lemma proves the correctness of Algorithm NCSA.

Lemma 22 *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. Then, for any given combinational block $u \in V$, Algorithm* NCSA *correctly determines the maximum possible increase in its propagation delay $d(u)$ such that G is still properly timed by π.*

Proof. According to Lemma 18, the circuit G is properly timed by π if and only if Inequalities (2.6), (2.7), and (2.8) are satisfied. We will show that these inequalities are still satisfied when $d(u)$ increases by the value $\delta(u)$ that is computed by Algorithm NCSA. We will also show that some of these inequalities is violated for greater values of $\delta(u)$.

First, let us consider Inequality (2.8). Increasing $d(u)$ can only violate Inequality (2.8) for simple cycles that go through vertex u. From Step 3 we have

$$\begin{aligned}
\delta(u) &= \min\left\{l(v) + w'(e) : v \xrightarrow{e} u \in E\right\} \\
&= \min\left\{w'(e) : u \overset{c}{\rightsquigarrow} u \in E\right\} \\
&= \min\left\{\sum_{x \xrightarrow{e} y \in c} \left(\lfloor w(e)/2 \rfloor \pi + (w(e) \bmod 2)\left(\gamma_{\chi(x)} + \phi_{1-\chi(x)}\right) - d(y)\right) : u \overset{c}{\rightsquigarrow} u \in E\right\} \\
&= \min\left\{(\gamma_1 + \phi_0)\left(\sum_{x \xrightarrow{e} y \in c} w_0(e)\right) + (\gamma_0 + \phi_1)\left(\sum_{x \xrightarrow{e} y \in c} w_1(e)\right) - \sum_{x \in c} d(x) : u \overset{c}{\rightsquigarrow} u \in E\right\} \\
&= \min\left\{(\gamma_1 + \phi_0) w(c)/2 + (\gamma_0 + \phi_1) w(c)/2 - d(c) : u \overset{c}{\rightsquigarrow} u \in E\right\} \\
&= \min\left\{(\gamma_1 + \phi_0 + \gamma_0 + \phi_1) w(c)/2 - d(c) : u \overset{c}{\rightsquigarrow} u \in E\right\}
\end{aligned}$$

where $w_0(e)$ and $w_1(e)$ denote the number of latches for each edge $e \in E$ that are clocked on ϕ_0 and ϕ_1, respectively. From the last inequality, using Equation (2.1), we infer that

$$\delta(u) = \min\left\{\pi w(c)/2 - d(c) : u \overset{c}{\rightsquigarrow} u \in E\right\} .$$

Therefore, this value of $\delta(u)$ is the maximum by which we can increase $d(u)$ without violating Inequality (2.8).

ALLNCSA(G, π)
1 Compute all-pairs shortest-paths lengths $l(u, v)$ for each pair $u, v \in V$, on graph G
  with edge-length $w'(e) = \lfloor w(e)/2 \rfloor \pi + (w(e) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) - d(v)$
  for every edge $u \xrightarrow{e} v \in E$.
2 **for** each $u \in V$
3   **do** $\delta(u) = \min\left\{l(u, v) + w'(e) : v \xrightarrow{e} u \in E\right\}$
4 **for** each $u \in V$
5   **do** $\delta(u) = \min\left\{\delta(u), \min_{v \in V}\left\{\phi_{\chi(v)} - d(v) + l(v, u)\right\} + \min_{v \in V}\left\{l(u, v) + \gamma_{\chi(v)} + \phi_{1-\chi(v)}\right\}\right\}$
6 **for** each $u \in V$
7   **do if** $\delta(u) \geq 0$
8   **then return** $\delta(u)$
9   **else return fail**

Figure 2-6: Algorithm ALLNCSA which solves the noncritical sensitivity analysis problem for all combinational blocks in a circuit G. The algorithm takes as input a two-phase circuit G and a clocking scheme π. It produces as output the maximum increase $\delta(u)$ in the delay $d(u)$ of each combinational block $u \in V$ such that the proper timing of G by the clocking scheme π is not affected.

Now, let us consider Inequalities (2.6) and (2.7). In Step 4 the algorithm computes a value $\delta(u)$ such that

$$\begin{aligned}\delta(u) &\leq \min\left\{\phi_{\chi(x)} - d(x) + \sum_{e \in p} w'(e) + \sum_{e \in q} w'(e) + \gamma_{\chi(y)} + \phi_{1-\chi(y)} : x \overset{p}{\rightsquigarrow} u, u \overset{q}{\rightsquigarrow} y, x, y \in V\right\} \\ &= \min\left\{\tau(r) - d(r) : x \overset{r}{\rightsquigarrow} y, u \in r, \text{ and } x, y \in V\right\} .\end{aligned}$$

Therefore, if we increase $d(u)$ by this value of $\delta(u)$ we do not violate any of the Inequalities (2.6) and (2.7). Moreover, this is the maximum value of $\delta(u)$ that does not violate these inequalities, because it satisfies the one that corresponds to the path r with equality. □

Algorithm ALLNCSA, shown in Figure 2-6, solves the noncritical sensitivity analysis problem for all combinational blocks $u \in V$ in $O(VE + V^2 \lg V)$ time. This algorithm simply solves an all-pairs shortest-paths problem on G with edge-lengths chosen in a way that accounts automatically for the path or the cycle with the minimum slack.

In the following two lemmas, we prove the correctness of Algorithm ALLNCSA and a bound on its running time.

Lemma 23 *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. Then, for each combinational block $u \in V$, Algorithm* ALLNCSA *correctly determines the maximum possible increase in its propagation delay $d(u)$ such that G is still properly timed by π.*

Proof. For every $u \in V$, we repeat the proof of Lemma 22. □

Lemma 24 *Algorithm* ALLNCSA *terminates in $O(VE + V^2 \lg V)$ time.*

Proof. The all-pairs shortest-paths computation in Step 1 can be performed in $O(VE +$

```
CSA(G, π, u)
if TV(G, π) = yes
    then return 0
d_init(u) ← d(u)
d(u) ← 0
if TV(G, π) = no
    then return "G cannot be properly timed just by reducing d(u)"
    else (u) ← NCSA(G, π, u)
         δ(u) ← d_init(u) − d(u)
         return δ(u)
```

Figure 2-7: Algorithm CSA for the critical sensitivity analysis problem. The algorithm takes as input a circuit G, a clocking scheme π, and a vertex u. It produces as output the minimum $\delta(u)$ such that G is properly timed by π if we decrease the propagation delay $d(u)$ by $\delta(u)$.

$V^2 \lg V$) time using Johnson's algorithm [5]. The minimizations in Steps 2 and 4 require $O(V^2)$ time. Therefore, Algorithm AllNCSA terminates in $O(VE + V^2 \lg V)$ time. □

Note that either Algorithm NCSA or Algorithm AllNCSA can be used to detect combinational blocks that lie on critical paths or critical cycles for the given clocking scheme π. For each critical block u, the value $\delta(u)$ computed by these algorithms is simply zero. Moreover, Algorithm AllNCSA can detect if the circuit G is not properly timed by the given clocking scheme π. If the clocking scheme π does not satisfy the timing constraints around some simple cycle, then the shortest-paths computation in Step 1 detects a negative edge-weight cycle and the algorithm fails. If the clocking scheme π does not satisfy the timing constraints along some path, then $\delta(u)$ is negative, and the algorithm fails again.

2.4.2 Critical Sensitivity Analysis

Algorithm CSA, shown in Figure 2-7, solves the critical sensitivity analysis problem for a single combinational block $u \in V$. The algorithm returns 0 if G is properly timed by the given clocking scheme π. Otherwise, it saves the original propagation delay $d(u)$ in the variable $d_{init}(u)$ and computes the maximum nonnegative $d(u)$ that will not affect the proper timing of G by the clocking scheme π. The difference between the original propagation delay of u and its new propagation delay gives $\delta(u)$. It is straightforward to verify the correctness of Algorithm CSA. The algorithm terminates in $O(VE)$ steps, since Algorithm NCSA runs in $O(VE)$ time.

2.5 Period minimization by clock tuning

In this section, we address the *tuning problem*: Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$ and gaps γ_0 and γ_1, compute a clocking scheme $\pi^* = \langle \phi_0^*, \gamma_0, \phi_1^*, \gamma_1 \rangle$ with minimum period such that G is properly timed by π^*. We show that two-dimensional linear programming can be used to solve this problem in $O(VE)$ time.

The basic idea behind the period minimization algorithm is to view π, ϕ_0, and ϕ_1 as

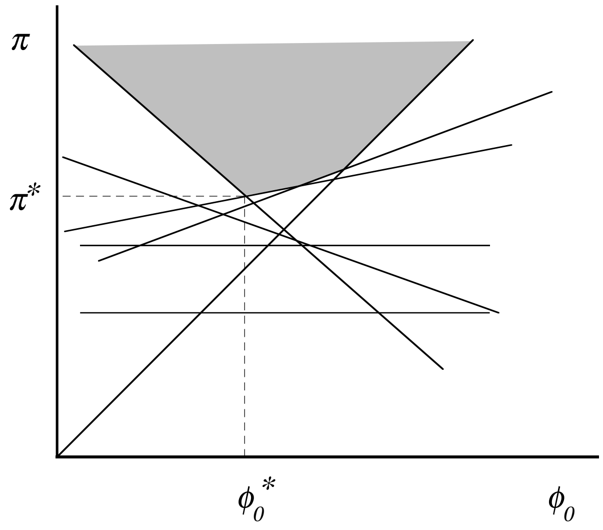


Figure 2-8: The tuning problem as a two-dimensoinal linear program in the (ϕ_0, π) plane. The lines with negative slope describe constraints on paths $u \rightsquigarrow v$ with $\chi(u) = 0$ an $\chi(v) = 1$. The lines with positive slope correspond to constraints on paths $u \rightsquigarrow v$ with $\chi(u) = 1$ and $\chi(v) = 0$. The horizontal lines describe constraints on paths with $\chi(u) = \chi(v)$, and the lower bound on the clock period due to the constraints on cycles. The shaded area is the set of feasible points for the linear program. The circuit G is properly timed by every clocking scheme that corresponds to a point in the shaded area.

variables. By Equation (2.1), we have $\phi_1 = \pi - \gamma_0 - \gamma_1 - \phi_0$, and thus, the constraints defined in Lemma 19 can be viewed as inequalities in the two variables π and ϕ_0. In fact, they form a linear program in which the objective is to minimize π. This program can be described by a set of lines in the (ϕ_0, π) plane, as shown in Figure 2-8. We distinguish three kinds of lines corresponding to Inequalities (2.11), (2.12), and (2.8). The next three lemmas show how to efficiently derive each of the three sets of linear constraints that form the linear program.

Lemma 25 *In $O(VE)$ time, the constraints defined by Inequality (2.11) can be reduced to an equivalent set of $O(V)$ linear inequalities in the variables π and ϕ_0.*

Proof. The constraints defined by Inequality (2.11) depend on both π and ϕ_0. We can separate these constraints into two sets depending on the value of $\chi(v)$. If $\chi(v) = 1$, then the inequality becomes

$$\pi \geq \frac{D(v,i) - \phi_0}{(i+1)/2} , \tag{2.13}$$

which defines a half-plane of feasible (ϕ_0, π) points for each fixed i. If $\chi(v) = 0$, the inequality becomes

$$\pi \geq \frac{D(v,i) + \gamma_0 + \gamma_1 + \phi_0}{(3+i)/2} , \tag{2.14}$$

which once again defines a half-plane of feasible (ϕ_0, π) points for each fixed i, since γ_0 and γ_1 are constants (see Figure 2-8). Each inequality holds for each $v \in V$ and for each odd i in the range $1 \leq i \leq 3|V| - 3$. The $O(V^2)$ constraints defined by Inequality (2.11) can be determined in $O(VE)$ time by computing the values $D(v,i)$, as in Step 2 of Algorithm TV. These constraints can be reduced in $O(V^2)$ time to an equivalent set of $O(V)$ constraints by selecting, for each odd i in the range $1 \leq i \leq 3|V| - 3$, the particular constraint for which $D(v,i)$ is maximized. The total running time is $O(VE) + O(V^2) = O(VE)$. □

Lemma 26 *In $O(VE)$ time, the constraints defined by Inequality (2.12) can be reduced to a single lower bound on the clock period π.*

Proof. The constraints defined by Inequality (2.12) can be rewritten as

$$\pi \geq \frac{D(v,i) + \gamma_{1-\chi(v)}}{(i+2)/2} , \tag{2.15}$$

for even i. Each of these constraints depends on π but not on ϕ_0 (see Figure 2-8). Consequently, these constraints together determine a single lower bound on π which is independent of the duty cycle of either phase. After computing the values of $D(v,i)$ in $O(VE)$ time, this bound on π can be determined by simply finding the maximum of the $O(V^2)$ right-hand sides of Inequality (2.15). □

The third of our lemmas focuses on Inequality (2.8). As in Lemma 26, we can compute a single lower bound on the clock period π which is independent of the duty cycles. This lower bound can be found by solving a "tramp steamer" problem.

The *tramp steamer* problem (also known as the *minimum cost-to-time ratio cycle* problem) was formulated in [7] as follows. Let $G = \langle V, E, s, t\rangle$ be a directed graph in which each edge $u \xrightarrow{e} v \in E$ has an integer cost $s(e)$ and an integer transit time $t(e)$, such that for any cycle c in G, we have $\sum_{e\in c} t(e) > 0$. For any cycle c in G, define the cost-to-time ratio of the cycle by

$$R(c) = \frac{\sum_{e\in c} s(e)}{\sum_{e\in c} t(e)} .$$

The problem is to find

$$R(G) = \min\{R(c) : c \text{ is a cycle in } G\} ,$$

which is the minimum such ratio over all cycles in the graph. If $t(e) \geq 0$ for all $e \in E$, then the algorithm from [16] can solve the tramp steamer problem in $O(TE)$ time, where

$$T = \sum_{u\in V} \max\{t(e) : u \xrightarrow{e} v \in E\} .$$

The following lemma relates the constraints determined by Inequality (2.8) to the tramp steamer problem.

Lemma 27 *In $O(VE)$ time, the constraints defined by Inequality (2.8) can be reduced to a single lower bound on the clock period π.*

Proof. Given the circuit $G = (V, E, d, w, \chi)$, define $\tilde{G} = \langle V, E, s, t\rangle$ to be the graph obtained by assigning to each edge $u \xrightarrow{e} v \in E$ a cost $s(e) = -d(u)$ and a transit time $t(e) = w(e)$. Then we claim Inequality (2.8) is satisfied if and only if

$$\pi \geq -2R(\tilde{G}) , \tag{2.16}$$

where $R(\tilde{G})$ is the minimum cost-to-time ratio of any cycle in $\tilde{G}$.

We first prove that Inequality (2.8) implies Inequality (2.16). Let c be the cycle in $\tilde{G}$ with minimum cost-to-time ratio, that is, $R(c) = R(\tilde{G})$. By Inequality (2.8), we have $d(c) \leq \pi(w(c)/2)$, and hence

$$\begin{aligned}
\pi &\geq \frac{2d(c)}{w(c)} \\
&= 2\frac{\sum_{u\in c} d(u)}{\sum_{e\in c} w(e)} \\
&= 2\frac{\sum_{e\in c} -s(e)}{\sum_{e\in c} t(e)} \\
&= -2R(\tilde{G}) \ .
\end{aligned}$$

The proof for the other direction of the claim is similar.

Using the algorithm for the tramp steamer problem given in [16], the cycle constraints can be checked in $O(VE)$ time. In order to obtain this running time, we must guarantee that the transit time of any path with $|V|$ edges is $O(V)$. Indeed, this requirement is met, since from Corollary 17 the number of latches on any edge can be reduced to at most 3. □

The following theorem combines Lemmas (25), (26), and (27) to solve the tuning problem for two-phase circuits.

Theorem 28 *The tuning problem can be solved for a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi\rangle$ and gaps γ_0 and γ_1 in $O(VE)$ time.*

Proof. By Lemma 19, any clock period π must satisfy Inequalities (2.11), (2.12), and (2.8), which, by Lemmas (25), (26), and (27), reduce to Inequalities (2.13), (2.14), (2.15), and (2.16), which are linear in ϕ_0 and π. We additionally must ensure that only valid clocking schemes are considered, which we can do by adding the constraints $\phi_0 \geq 0$ and $\phi_1 = \pi - \gamma_0 - \gamma_1 - \phi_0 \geq 0$. Thus, all the constraints can be phrased as linear inequalities in ϕ_0 and π, as is shown in Figure 2-8.

By linear programming theory [41], the optimal clock period π^* can be obtained at a point (ϕ_0^*, π^*) corresponding to the intersection of these $O(V)$ constraints. Megiddo's algorithm [37] can solve such a two-dimensional linear program in $O(V)$ time. Alternatively, one can first compute the $O(V^2)$ intersections among Inequalities (2.13) and (2.14), $\phi_0 \geq 0$, and $\pi - \gamma_0 - \gamma_1 - \phi_0 \geq 0$ (the nonhorizontal constraints). Then, let (ϕ_0', π') be the intersection point of maximum π value, and let π'' be the greatest lower bound on π derived from the remaining (horizontal) inequalities. The optimal period is $\pi^* = \max\{\pi', \pi''\}$, since (ϕ_0', π') is above every nonhorizontal line, and all linear inequalities constrain π from below. A feasible phase corresponding to π^* is $\phi_0^* = \phi_0'$. In either case, the overall running time is $O(VE)$. □

2.6 Retiming with Symmetric Clocking Schemes

In this section, we present an $O(VE + V^2 \lg V)$-time algorithm for the *retiming problem with symmetric clocking schemes*: Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi\rangle$ and a symmetric clocking scheme $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$, compute a retiming of G which is properly timed by π, or else determine that no such retiming exists. Like several previous retiming algorithms [29, 52], our algorithm casts retiming for a symmetric clocking scheme as a mixed-integer linear program.

The retiming transformation relocates the latches in a circuit G without changing the functionality of the circuit. Given an integer $r(v)$ for each vertex $v \in V$, we retime the circuit G by removing $r(v)$ latches from each output wire of v and inserting $r(v)$ latches to each input wire of v. The variable $r(v)$ is called the *lag* of vertex v, because it counts by how many phases we delay the output of v's computation when we retime G. After retiming, we obtain a retimed circuit $G_r = \langle V, E, d, w_r, \chi_r \rangle$, where

$$w_r(e) = w(e) + r(v) - r(u) \tag{2.17}$$

for every edge $u \xrightarrow{e} v \in E$, and

$$\chi_r(v) = \begin{cases} \chi(v) & \text{if } r(v) \text{ is even ,} \\ 1 - \chi(v) & \text{if } r(v) \text{ is odd ,} \end{cases}$$

for every vertex $v \in V$. To ensure that G_r is a well-formed two-phase circuit, we require that all edge weights in G_r are nonnegative, which is equivalent to the condition that

$$w(e) + r(v) - r(u) \geq 0 \tag{2.18}$$

for every edge $u \xrightarrow{e} v \in E$. We need not check any other conditions for G_r to be well formed, since retiming does not change the weight of a cycle, which means that all cycles retain the positive weight they had in G, and one can verify that Condition WF3 holds over all edges in G_r.

Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$ and a symmetric clocking scheme $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$, the problem of retiming with symmetric clocking schemes is to compute a retiming function r such that G_r is a well-formed circuit which is properly timed by π, or to determine that no such retiming exists. The following lemma gives a set of necessary and sufficient constraints that such a retiming r must satisfy.

Lemma 29 *Let $G = \langle V, E, d, w, \chi \rangle$ be a two-phase, level-clocked circuit, let $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$ be a symmetric clocking scheme, and let $r : V \to \mathbf{Z}$ be a retiming function. Then, the retimed circuit G_r is properly timed by π if and only if*

$$r(u) - r(v) \leq w(e) \tag{2.19}$$

holds for every edge $u \xrightarrow{e} v \in E$, and

$$r(u) - r(v) \leq \sum_{i \xrightarrow{e} j \in p} \left(w(e) - \frac{2}{\pi} d(j) \right) + \left(2 - \frac{2}{\pi}(d(u) + \gamma) \right) \tag{2.20}$$

for every path $u \overset{p}{\leadsto} v$ in G_r.

Proof. ($\Rightarrow$) Let G_r be properly timed by π. Since all edge weights in G_r are nonnegative, we have $w_r \geq 0$, which by Equation (2.17) implies Inequality (2.19). By Lemma 14, Inequalities

(2.2) and (2.3) must also hold, but both reduce to

$$d(p) \leq \pi\left(\frac{w_r(p)}{2}\right) + \pi - \gamma\ ,$$

since the clocking scheme π is symmetric with period $\pi = 2\phi + 2\gamma$. Let us consider a path $u \overset{p}{\leadsto} v$. Using the fact that $w_r(p) = w(p) + r(v) - r(u)$, which can be proved by induction from Equation (2.17), we obtain

$$\begin{aligned} d(p) &\leq \pi\left(\frac{w_r(p)}{2}\right) + \pi - \gamma \\ &= \pi\left(\frac{w(p) + r(v) - r(u)}{2}\right) + \pi - \gamma\ , \end{aligned}$$

which can be rewritten as

$$\begin{aligned} r(u) - r(v) &\leq w(p) - \frac{2}{\pi}d(p) + \left(2 - \frac{2\gamma}{\pi}\right) \\ &= \sum_{i \xrightarrow{e} j \in p} \left(w(e) - \frac{2}{\pi}d(j)\right) - \frac{2}{\pi}d(u) + \left(2 - \frac{2\gamma}{\pi}\right) \\ &= \sum_{i \xrightarrow{e} j \in p} \left(w(e) - \frac{2}{\pi}d(j)\right) + \left(2 - \frac{2}{\pi}(d(u) + \gamma)\right)\ . \end{aligned}$$

Therefore, Inequality (2.20) holds.

($\Leftarrow$) All of the implications in the proof of the forward direction can be reversed. □

Lemma 29 provides necessary and sufficient conditions that a retiming r must satisfy such that G_r is a well-formed circuit which is properly timed by a clocking scheme π. Unfortunately, there are an infinite number of constraints in the set specified by Inequality (2.20). The following theorem shows that the number of constraints can be reduced to $O(V + E)$.

Lemma 30 *Let $G = \langle V, E, d, w, \chi\rangle$ be a two-phase, level-clocked circuit, and let $\pi = \langle \phi, \gamma, \phi, \gamma\rangle$ be a symmetric clocking scheme. Then there exists a retiming $r : V \to \mathbf{Z}$ of G such that G_r is properly timed by π if and only if there exists an assignment of a real value $R(v)$ and an integer value $r(v)$ to each vertex $v \in V$ such that the following conditions are satisfied:*

$$\begin{aligned} r(u) - r(v) &\leq w(e) && \text{for all } u \xrightarrow{e} v \in E, && (2.21) \\ R(v) - r(v) &\leq 0 && \text{for all } v \in V, && (2.22) \\ R(u) - R(v) &\leq w(e) - \frac{2}{\pi}d(v) && \text{for all } u \xrightarrow{e} v \in E, && (2.23) \\ r(u) - R(u) &\leq 2 - \frac{2}{\pi}(d(u) + \gamma) && \text{for all } u \in V. && (2.24) \end{aligned}$$

Proof. ($\Leftarrow$) From Lemma 29 it suffices to show that Inequalities (2.21), (2.22), (2.23), and (2.24) imply Inequalities (2.19) and (2.20). Inequality (2.19) is immediately satisfied, since by Inequality (2.21) we have $r(u) - r(v) \leq w(e)$ for all $u \xrightarrow{e} v \in E$. To prove

Inequality (2.20), consider a path $u \overset{p}{\leadsto} v$. Inequality (2.23) holds for every edge on p, and if we sum all these inequalities, we obtain

$$R(u) - R(v) \le \sum_{i \xrightarrow{e} j \in p} \left(w(e) - \frac{2}{\pi} d(j) \right) ,$$

since the left-hand side telescopes. Adding this inequality to Inequalities (2.22) and (2.24) yields Inequality (2.20).

($\Rightarrow$) By Lemma 29 we need only show that if r is an assignment of integers to the vertices in V that satisfies Inequalities (2.19) and (2.20), then we can find an assignment R of reals to the vertices in V such that r and R satisfy Inequalities (2.21), (2.22), (2.23), and (2.24). Inequality (2.18) directly implies Inequality (2.21). We construct an auxiliary graph to determine R and show that the remaining three inequalities are satisfied.

Define the auxiliary graph $H = (V_H, E_H, w_H)$ by

$$\begin{array}{rcl} V_H &=& V \cup \{t\} \\ E_H &=& E \cup \{u \to t : u \in V\} \cup \{t \to u : u \in V\} \\ w_H(e) &=& \left\{ \begin{array}{ll} r(v) & \text{for all } v \xrightarrow{e} t \in E_H \ , \\ w(e) - \frac{2}{\pi} d(v) & \text{for all } u \xrightarrow{e} v \in E \ , \\ -r(u) + 2 - \frac{2}{\pi}(d(u) + \gamma) & \text{for all } t \xrightarrow{e} u \in E_H \ , \end{array} \right. \end{array}$$

where t is an additional vertex not in V. Define $R(v)$ for all $v \in V_H$ as the length of a shortest (least-weight) path in H from v to t, which is well-defined if H contains no negative-weight cycles [5, Chapter 25], a fact that we shall prove shortly.

Assuming that H contains no negative-weight cycles, we can prove Inequalities (2.22), (2.23), and (2.24) by relying on the following basic inequality of shortest paths [5, Chapter 25]:

$$R(u) \le R(v) + w_H(e) \tag{2.25}$$

for every edge $u \xrightarrow{e} v$ in E_H. To prove Inequality (2.22), we consider Inequality (2.25) with $w_H(e) = r(u)$ for all edges $u \xrightarrow{e} t$:

$$\begin{array}{rcl} R(u) &\le& R(t) + r(u) \\ &\le& r(u) \ , \end{array}$$

since the shortest path from t to itself has length $R(t) = 0$. Inequalities (2.23) and (2.24) follow from similar reasoning by considering Inequality (2.25) with the other two classes of edge weights.

It remains to show that H contains no negative-weight cycles. Suppose, for the sake of contradiction, that $u \overset{c}{\leadsto} u$ is a negative-weight cycle in H that contains the minimum number of edges among all negative-weight cycles in H. The cycle c visits t at most once, since otherwise, c would contain a negative-weight subcyle with fewer edges. We consider the cases $t \in c$ and $t \notin c$ separately.

If t is a vertex in c, we can break the cycle c into three parts: $c = t \xrightarrow{e_1} u \overset{p}{\leadsto} v \xrightarrow{e_2} t$, where

p does not visit t. Breaking the weight of c into its constituent parts, we obtain

$$\begin{aligned} w_H(c) &= w_H(e_1) + w_H(p) + w_H(e_2) \\ &= \left(-r(u) + 2 - \frac{2}{\pi}(d(u)+\gamma)\right) + \sum_{i \xrightarrow{e} j \in p} \left(w(e) - \frac{2}{\pi}d(j)\right) + r(v) \\ &< 0 \,, \end{aligned}$$

which, by moving $r(u)$ and $r(v)$ to the right-hand side of the strict inequality, directly contradicts Inequality (2.20).

If t is not a vertex in c, we consider a path p that consists of $k > 0$ repetitions of the cycle c, and which starts and ends at some vertex u on c. Since $w_H(c) < 0$, we can make $w_H(p) = k \cdot w_H(c)$ as negative as we wish by picking k sufficiently large. From Inequality (2.20), we obtain

$$\begin{aligned} 0 &= r(u) - r(u) \\ &\le \sum_{i \xrightarrow{e} j \in p} \left(w(e) - \frac{2}{\pi}d(j)\right) + \left(2 - \frac{2}{\pi}(d(u)+\gamma)\right) \\ &= w_H(p) + \left(2 - \frac{2}{\pi}(d(u)+\gamma)\right) \\ &= k \cdot w_H(c) + \left(2 - \frac{2}{\pi}(d(u)+\gamma)\right) \\ &< 0 \,, \end{aligned}$$

by picking $k > -\left(2 - (2/\pi)(d(u)+\gamma)\right)/w_H(c)$. This contradiction completes the proof. □

The set of constraints defined in Lemma 30 form a mixed-integer linear programming problem. Although mixed-integer linear programming is in general NP-hard [13], the simple form of the constraints in the lemma allows the problem to be solved efficiently. In particular, Inequalities (2.21), (2.22), (2.23), and (2.24) constitute a mixed-integer linear programming problem of the following form.

Problem MI *Let $H = (V, V_I, E, a)$ be an edge-weighted, directed graph, where $V = \{1, 2, \ldots, n\}$ is the vertex set, V_I (the "integer" vertices) is a subset of V, the edge set E is a subset of $V \times V$, and for each edge $(i,j) \in E$ the edge weight $a(i,j)$ is a real number. Find a vector $x = (x_1, x_2, \ldots, x_n)$ satisfying the constraints that*

$$x_i - x_j \le a(i,j)$$

for all $(i,j) \in E$, and that $x_i \in \mathbf{Z}$ for all $i \in V_I$, or determine that no feasible vector exists. □

Problem MI can be solved in $O(VE + V^2 \lg V)$ time by applying Algorithm MILP from [30]. Thus, we obtain the following theorem.

Theorem 31 *The retiming problem with symmetric schemes can be solved for a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$ and a symmetric clocking scheme $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$ in $O(VE + V^2 \lg V)$ time.*

```
RwSCS(G, π)
Generate Inequalities (2.21), (2.22), (2.23), and (2.24) from G and π.
Apply Algorithm MILP on Inequalities (2.21), (2.22), (2.23), and (2.24) to compute a retiming r.
if all constraints are satisfied
    then return r
    else fail
```

Figure 2-9: Algorithm RwSCS for retiming with symmetric clocking schemes. The algorithm takes as input a two-phase, level-clocked circuit $G = (V, E, d, w, \chi)$ and a symmetric clocking scheme $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$. It produces as output a retiming r such that G_r is properly timed by π, or else it determines that no such retiming is possible.

Proof. Algorithm RwSCS in Figure 2-9 solves the retiming problem with symmetric phases. It simply applies Algorithm MILP from [30] to the constraints in Lemma 30. Since $|V_H| = |V| + 1$ and $|E_H| = O(V + E)$, the running time of RwSCS is $O(VE + V^2 \lg V)$. □

2.7 Retiming with General Clocking Schemes

In this section, we study the *retiming problem*: Given a circuit $G = \langle V, E, d, w, \chi \rangle$ and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$, compute a retiming function r such that G_r is a well-formed circuit which is properly timed by π, or else determine that no such retiming exists. First, we make a short digression in order to introduce *integer monotonic programming*, a problem related to integer linear programming in which both sides of the inequalitites are monotone, but not necessarily linear, functions of the unknowns. We consider a simple case of integer monotonic programming in which each left-hand side is a function of a single unknown and show that problems of this nature admit a unique minimum solution $\overline{x}$. We describe a generic procedure for finding $\overline{x}$. We then cast the retiming problem in the form of a simple integer monotonic programming problem and use the generic procedure to obtain an $O(V^3)$-time algorithm for the retiming problem.

The integer monotonic programming problem is defined as follows.

Definition *Let S be a set of constraints over the unknowns $x_1, x_2, \ldots, x_n$, in which the kth constraint has the form*

$$f_k(x_1, x_2, \ldots, x_n) \geq g_k(x_1, x_2, \ldots, x_n) \ ,$$

where the functions f_k and g_k are monotonically increasing with respect to each x_i, for $j = 1, 2, \ldots, n$. The integer monotonic programming problem is to find a vector $x = \langle x_1, x_2, \ldots, x_n \rangle$ of integers satisfying S, or determine that no feasible vector exists. An integer monotonic programming problem is simple if each f_k is a function of a single unknown; thus, the kth constraint has the simpler form

$$f_k(x_i) \geq g_k(x_1, x_2, \ldots, x_n) \ .$$

□

Based on the monotonicity of f_k and g_k, we can argue that if a simple integer monotonic program has a solution over the nonnegative integers, then it has a unique "minimum solution".

Lemma 32 *Let $x = \langle x_1, x_2, \ldots, x_n \rangle$ and $x' = \langle x'_1, x'_2, \ldots, x'_n \rangle$ be two solutions to some simple integer monotonic program, and let $x''_i = \min\{x_i, x'_i\}$ for all $i = 1, 2, \ldots, n$. Then, $x'' = \langle x''_1, x''_2, \ldots, x''_n \rangle$ is also a solution.*

Proof. Consider the constraint $f_k(x_i) \geq g_k(x_1, x_2, \ldots, x_n)$ in the simple integer monotonic program. Assume, without loss of generality, that $x''_i = x_i$. It follows that

$$\begin{aligned} f_k(x''_i) &= f_k(x_i) \\ &\geq g_k(x_1, x_2, \ldots, x_n) \\ &\geq g_k(x''_1, x''_2, \ldots, x''_n), \end{aligned}$$

since g_k is monotonically increasing with respect to all its arguments. Therefore, x'' is also a solution to the monotonic program. □

Corollary 33 *For any simple integer monotonic program having a solution in which $x_i \geq 0$ for $i = 1, 2, \ldots, n$, there exists a unique minimum solution $\langle \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n \rangle$ which is minimum in the sense that for all other solutions $\langle x_1, x_2, \ldots, x_n \rangle$, we have $\overline{x}_i \leq x_i$ for $i = 1, 2, \ldots, n$.*

Proof. The proof follows from Lemma 32 and the nonnegativity of the x_i. □

If a simple integer monotonic programming problem has a solution over the nonnegative integers, the relaxation procedure MONORELAX in Figure 2-10 can find the minimum solution. After initializing the x_i, the procedure performs a sequence of *relaxations* over the set of constraints. Each relaxation step (lines 4–5) consists of determining a constraint $f_k(x_i) \geq g_k(x_1, x_2, \ldots, x_n)$ which is violated and then incrementing x_i. If there is a solution, the running time is proportional to $\sum_{i=1}^{n} \overline{x}_i$ multiplied by the time it takes to find a violated constraint. If there is no solution, however, the procedure runs forever. Later in this section, we shall present a procedure based on MONORELAX to solve the retiming problem. This new procedure always terminates, however, because for the special case of the retiming problem we can prove that whenever the unknowns x_i exceed an upper bound, then the problem has no solution. The following theorem proves that MONORELAX finds the minimum solution when a solution exists.

Theorem 34 *Let S be the set of constraints in a simple integer monotonic programming problem over the unknowns $x_1, x_2, \ldots, x_n$, and suppose S has a solution. Then,* MONORELAX *finds the minimum solution $\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n$.*

Proof. We first show that after each iteration the invariant $\overline{x}_i \geq x_i$ holds for $i = 1, 2, \ldots, n$. The invariant initially holds, since $x_i = 0$ for every i. Consider an iteration of the procedure in which a constraint $f_k(x_i) \geq g_k(x_1, x_2, \ldots, x_n)$ is violated, which means $f_k(x_i) < g_k(x_1, x_2, \ldots, x_n)$. Since no constraints are violated for the minimum solution $\overline{x}$, and since

```
MONORELAX(S)
for i ← 1 to n
      do x_i ← 0
while there exists an unsatisfied constraint in S
      do pick an unsatisfied constraint f_k(x_i) ≥ g_k(x_1, x_2, ..., x_n)
          x_i ← x_i + 1
return ⟨x_1, x_2, ..., x_n⟩
```

Figure 2-10: Procedure MONORELAX for solving a simple integer monotonic program S over unknowns $x_1, x_2, \ldots, x_n$. The procedure returns a solution if and only if the constraints in S can be satisfied. Otherwise, it runs forever.

g_k is monotonic and $\overline{x}_i \geq x_i$, we have

$$\begin{aligned} f_k(\overline{x}_i) &\geq g_k(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) \\ &\geq g_k(x_1, x_2, \ldots, x_n) \\ &> f_k(x_i) \ . \end{aligned}$$

But since f_k is monotonic, $f_k(\overline{x}_i) > f_k(x_i)$ implies that $\overline{x}_i > x_i$. Thus, after incrementing x_i, the invariant $\overline{x}_i \geq x_i$ continues to hold.

To complete the correctness proof, observe that when all constraints are satisfied, we obtain a solution x, which, by Corollary 33 and the fact that $x_i \leq \overline{x}_i$, is equal to $\overline{x}$. Moreover, we must eventually achieve this unique minimum solution, because exactly $\sum_{i=1}^{n} \overline{x}_i$ relaxations can occur, since each relaxation increases $\sum_{i=1}^{n} x_i$ by exactly 1. □

We now turn to the retiming problem with general two-phase clocking schemes. Recall that given a circuit $G = \langle V, E, d, w, \chi \rangle$ and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$, we wish to compute a retiming function r such that G_r is a well-formed circuit which is properly timed by π, or else determine that no such retiming exists. When π is not symmetric, Inequalities (2.2) and (2.3) cannot be simplified as in the retiming problem with symmetric clocking schemes. Nevertheless, we can cast the retiming problem as a simple integer monotonic programming problem. The following lemma gives a set of necessary and sufficient conditions that such a retiming r must satisfy.

Lemma 35 *Let $G = \langle V, E, d, w, \chi \rangle$ be a two-phase, level-clocked circuit, let $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ be a clocking scheme, and let $r : V \to \mathbf{Z}$ be a retiming function. Then, the retimed circuit G_r is properly timed by π if and only if for every edge $u \xrightarrow{e} v \in E$, we have*

$$r(u) - r(v) \leq w(e) \ , \tag{2.26}$$

and for every path $u \overset{p}{\rightsquigarrow} v$, we have

$$\begin{aligned} d(p) \ \leq \ & \pi\left(\frac{1 + w(p)}{2}\right) + \phi_{\chi(u)} \\ & + \pi \left\lfloor \frac{r(v)}{2} \right\rfloor + (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \end{aligned}$$

$$-\pi\left\lfloor\frac{r(u)}{2}\right\rfloor-(r(u) \bmod 2)\left(\phi_{\chi(u)}+\gamma_{\chi(u)}\right)\ , \tag{2.27}$$

if $\chi(u) \neq \chi(v)$*, and*

$$\begin{aligned} d(p) &\leq \pi\left(\frac{2+w(p)}{2}\right)-\gamma_{1-\chi(u)} \\ &+\pi\left\lfloor\frac{r(v)}{2}\right\rfloor+(r(v) \bmod 2)\left(\gamma_{1-\chi(u)}+\phi_{\chi(u)}\right) \\ &-\pi\left\lfloor\frac{r(u)}{2}\right\rfloor-(r(u) \bmod 2)\left(\phi_{\chi(u)}+\gamma_{\chi(u)}\right)\ , \end{aligned} \tag{2.28}$$

if $\chi(u) = \chi(v)$*.*

Proof. ($\Rightarrow$) Let G_r be a well-formed circuit that is properly timed by π. Since G_r is well formed, all edge-weights w_r in G_r must satisfy $w_r \geq 0$, which by Equation (2.17) implies Inequality (2.26). Since G_r is properly timed by π, Inequalities (2.2) and (2.3) from Lemma 14 must hold. We prove that Inequalities (2.27) and (2.28) follow from Inequalities (2.2) and (2.3).

The proof is a case analysis that depends on examining all possible assignments of original phases to u and v and all possible parities of $r(u)$ and $r(v)$ for a path $u \overset{p}{\leadsto} v$ in G_r. Let us consider, for example, a path $u \overset{p}{\leadsto} v$, where $\chi(u) \neq \chi(v)$. Let r be a retiming with even $r(u)$ and odd $r(v)$; the other cases are similar. In this case, $\chi_r(u) = \chi(u)$, and $\chi_r(v) = 1 - \chi(v)$. Using Inequality (2.3) in Lemma 14 and the fact that $w_r(p) = w(p) + r(v) - r(u)$, we obtain

$$\begin{aligned} d(p) &\leq \pi\left(\frac{2+w_r(p)}{2}\right)-\gamma_{1-\chi_r(v)} \\ &= \pi\left(\frac{2+w(p)+r(v)-r(u)}{2}\right)-\gamma_{\chi(v)}. \end{aligned}$$

Since $\chi(u) \neq \chi(v)$, we have $\gamma_{\chi(v)} = \pi - \phi_{\chi(u)} - \gamma_{\chi(u)} - \phi_{\chi(v)}$ by Equation (2.1), and therefore

$$\begin{aligned} d(p) &\leq \pi\left(\frac{2+w(p)+r(v)-r(u)}{2}\right)+\phi_{\chi(u)}+\gamma_{\chi(u)}+\phi_{\chi(v)}-\pi \\ &= \pi\left(\frac{2+w(p)+r(v)-r(u)}{2}\right)+\phi_{\chi(u)}+\gamma_{\chi(u)}+\phi_{1-\chi(u)}-\pi \\ &= \pi\left(\frac{2+w(p)+r(v)-r(u)}{2}-1\right)+\phi_{\chi(u)}+\gamma_{\chi(u)}+\phi_{1-\chi(u)} \\ &= \pi\left(\frac{1+w(p)+(r(v)-1)-r(u)}{2}\right)+\phi_{\chi(u)}+\gamma_{\chi(u)}+\phi_{1-\chi(u)} \\ &= \pi\left(\frac{1+w(p)}{2}\right)+\phi_{\chi(u)} \\ &+\pi\left(\frac{r(v)-1}{2}\right)+\gamma_{\chi(u)}+\phi_{1-\chi(u)} \\ &-\pi\left(\frac{r(u)}{2}\right). \end{aligned}$$

Using the fact that $r(v) \bmod 2 = 1$ and $r(u) \bmod 2 = 0$, we obtain

$$\begin{aligned} d(p) \quad \leq \quad & \pi\left(\frac{1+w(p)}{2}\right) + \phi_{\chi(u)} \\ & +\pi\left\lfloor\frac{r(v)}{2}\right\rfloor + (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \\ & -\pi\left\lfloor\frac{r(u)}{2}\right\rfloor - (r(u) \bmod 2)\left(\phi_{\chi(u)} + \gamma_{\chi(u)}\right) \ , \end{aligned}$$

thus proving Inequality (2.27).

($\Leftarrow$) All the implications in the proof of the forward direction can be reversed. □

Inequalities (2.27) and (2.28) can be intuitively understood in terms of rise-to-fall times. As we saw in the proof of Lemma 35, Inequality (2.27) follows from Inequality (2.2), which holds for every path $u \overset{p}{\leadsto} v$ in G with $\chi(u) \neq \chi(v)$. The first line on the right-hand-side of Inequality (2.27) is just the initial allowance of time along p. The second line gives the *net* allowance of time added to the path p due to the latches that are shifted onto or off p through v. The term $\lfloor r(v)/2 \rfloor \pi$ counts the number of whole periods that are shifted through v, and the other term accounts for the effect of shifting fractional periods. By shifting onto p a single latch through the last vertex v in the path, we have a net allowance of $\gamma_{\chi(u)} + \phi_{1-\chi(u)}$, since the new latch on the boundary of p is clocked on $\phi_{1-\chi(u)}$. Similarly, by shifting off p a single latch through v, the net allowance is $-\gamma_{1-\chi(u)} - \phi_{\chi(u)}$, since the $\phi_{\chi(u)}$-latch that used to be on the boundary of p is no longer there. Finally, the third line on the right-hand-side of Inequality (2.27) gives the net allowance of time added to the path p due to the latches that are shifted onto or off of p through u. The interpretation of Inequality (2.28) is analogous.

Although Lemma 35 provides necessary and sufficient conditions that a retiming must satisfy, the set specified by Inequalities (2.26), (2.27), and (2.28) contains an infinite number of constraints. The following lemma shows that of this infinite number, all but $O(V^2)$ are redundant.

Lemma 36 *Let $G = \langle V, E, d, w, \chi\rangle$ be a two-phase, level-clocked circuit, let $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ be a clocking scheme, and let $r : V \to \mathbf{Z}$ be a retiming function. Moreover, let p be the shortest (least-weight) path from u to v in the graph $G' = \langle V, E, w'\rangle$ with edge-weight function $w'(e) = \pi w(e)/2 - d(j)$ for each edge $i \xrightarrow{e} j$ in E. Then, the retimed circuit G_r is properly timed by π if and only if for every edge $u \xrightarrow{e} v \in E$, Inequality (2.26)*

$$r(u) - r(v) \leq w(e)$$

holds, and for every pair of vertices $u, v \in V$, we have

$$\begin{aligned} d(p) \quad \leq \quad & \pi\left(\frac{1+w(p)}{2}\right) + \phi_{\chi(u)} \\ & +\pi\left\lfloor\frac{r(v)}{2}\right\rfloor + (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \\ & -\pi\left\lfloor\frac{r(u)}{2}\right\rfloor - (r(u) \bmod 2)\left(\phi_{\chi(u)} + \gamma_{\chi(u)}\right) \ , \end{aligned} \tag{2.29}$$

if $\chi(u) \neq \chi(v)$, *and*

$$
\begin{aligned}
d(p) \quad \le \quad & \pi\left(\frac{2+w(p)}{2}\right) - \gamma_{1-\chi(u)} \\
& +\pi\left\lfloor\frac{r(v)}{2}\right\rfloor + (r(v) \bmod 2)\left(\gamma_{1-\chi(u)} + \phi_{\chi(u)}\right) \\
& -\pi\left\lfloor\frac{r(u)}{2}\right\rfloor - (r(u) \bmod 2)\left(\phi_{\chi(u)} + \gamma_{\chi(u)}\right) \ , \qquad (2.30)
\end{aligned}
$$

if $\chi(u) = \chi(v)$.

Proof. We prove that Inequality (2.29) holds if and only if Inequality (2.27) holds. Inequality (2.27), which must hold for every path $u \overset{p}{\rightsquigarrow} v$ in G, can be rewritten

$$
\begin{aligned}
\pi\left\lfloor\frac{r(u)}{2}\right\rfloor + (r(u) \bmod 2)\left(\phi_{\chi(u)} + \gamma_{\chi(u)}\right) - \pi\left\lfloor\frac{r(v)}{2}\right\rfloor - (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) - \frac{\pi}{2} \\
\le \quad \pi\left(\frac{w(p)}{2}\right) - d(p) \\
= \quad \sum_{i \overset{e}{\to} j \in p}\left(\pi\frac{w(e)}{2} - d(j)\right) - d(u) \ . \qquad (2.31)
\end{aligned}
$$

Among all paths from u to v, the tightest constraint is generated by the path p that minimizes the sum on the right-hand side of Inequality (2.31). This path is the shortest (least-weight) path from u to v in G'. Therefore, Inequality (2.29) holds exactly when Inequality (2.27) holds. The proof for Inequality (2.30) is similar. □

The next lemma shows that the retiming problem can be reduced to the simple integer monotonic programming problem.

Lemma 37 *The retiming problem for a two-phase, level-clocked circuit* $G = \langle V, E, d, w, \chi\rangle$ *and a clocking scheme* $\pi = \langle\phi_0, \gamma_0, \phi_1, \gamma_1\rangle$ *can be reduced to the simple integer monotonic programming problem.*

Proof. The retiming problem for G and π is described by Inequalities (2.26), (2.29), and (2.30). In order to prove the lemma, we must show that each of these inequalities can be written in the form $f(r(v)) \ge g(r(u))$, where f and g are monotonic functions.

For every edge $u \overset{e}{\to} v \in E$, Inequality (2.26) can be written as

$$
r(v) + w(e) \ge r(u) \ ,
$$

which has the desired form, since both sides of the inequality are monotonic.

Let us concentrate, now, on Inequality (2.29). The analysis for Inequality (2.30) is similar. Inequality (2.29) can be written in the form $f(r(v)) \ge g(r(u))$ by letting

$$
\begin{aligned}
f(r(v)) &= \pi\left\lfloor\frac{r(v)}{2}\right\rfloor + (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) + \pi\left(\frac{1+w(p)}{2}\right) - d(p) + \phi_{\chi(u)} \ , \\
g(r(u)) &= \pi\left\lfloor\frac{r(u)}{2}\right\rfloor + (r(u) \bmod 2)\left(\phi_{\chi(u)} + \gamma_{\chi(u)}\right) \ .
\end{aligned}
$$

RETIME(G, π)
1 $Q \leftarrow \{$constraints from Lemma 36 of the form "$f(r(v)) \geq g(r(u))$"$\}$
2 **for** every vertex $v \in V$
3   **do** $r(v) \leftarrow 0$
4 **while** $Q \neq \emptyset$
5   **do** remove a constraint "$f(r(v)) \geq g(r(u))$" from Q
6   **if** $f(r(v)) < g(r(u))$
7   **then repeat** $r(v) \leftarrow r(v) + 1$
8   **if** $r(v) > 3|V| - 1$
9   **then fail**
10   **until** $f(r(v)) \geq g(r(u))$
11                 $Q \leftarrow Q \cup \{$all constraints with $r(v)$ on the right-hand side$\}$
12 **return** r

Figure 2-11: An algorithm which, given a two-phase circuit $G = (V, E, d, w, \chi)$ and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$, determines a retiming of G that is properly timed by π, or determines that no such retiming is possible.

We demonstrate that f is monotonically increasing with respect to its argument $r(v)$ by showing that the difference $f(r(v)+1) - f(r(v))$ is positive for every $r(v)$. The proof for the monotonicity of g with respect to $r(u)$ is similar.

$$\begin{aligned}
f(r(v)+1) - f(r(v)) &= \pi \left\lfloor \frac{r(v)+1}{2} \right\rfloor + ((r(v)+1) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \\
&\quad - \pi \left\lfloor \frac{r(v)}{2} \right\rfloor - (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \\
&\geq \pi \left(\left\lfloor \frac{r(v)+1}{2} \right\rfloor - \left\lfloor \frac{r(v)}{2} \right\rfloor \right) - (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \\
&= \pi\,(r(v) \bmod 2) - (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \\
&= (r(v) \bmod 2)\left(\pi - \gamma_{\chi(u)} - \phi_{1-\chi(u)}\right) \\
&> 0,
\end{aligned}$$

since $\lfloor (r(v)+1)/2 \rfloor - \lfloor r(v)/2 \rfloor = (r(v) \bmod 2)$ and $\pi > \gamma_{\chi(u)} + \phi_{1-\chi(u)}$. □

Figure 2-11 gives the pseudocode for Algorithm RETIME which solves the retiming problem for a given two-phase, level-clocked circuit G and a clocking scheme π. Algorithm RETIME operates in essentially the same way as procedure MONORELAX. The only difference is that Algorithm RETIME can detect if the retiming problem has no solution, in which case it returns that the problem is infeasible. We now prove a bound on the running time of Algorithm RETIME, and then we shall prove its correctness.

Lemma 38 *Algorithm* RETIME *can be implemented to terminate in $O(V^3)$ time.*

Proof. To compute the constraints in line 1 of the algorithm, we need to compute the shortest-paths between every pair of vertices in G. This computation can be performed in $O(VE + V^2 \lg V)$ time using Johnson's algorithm for all-pairs shortest-paths [5, Section 26.3].

To implement the set Q, we can use a FIFO queue and a flag for each constraint that indicates whether the constraint is in the queue.

For every constraint $f(r(v)) \geq g(r(u)) \in Q$, the functions $f(r(v))$ and $g(r(u))$ can be computed in $O(1)$ time, and since we have $|V|$ variables, the body of the **while** loop (line 4–line 11) can be completed in $O(V)$ time. Since the **repeat** loop in line 7 always increments some previously incremented variable $r(v)$, and since no variable $r(v)$ becomes greater than $3\,|V|$, line 7 is executed $O(V^2)$ times. Therefore, the total running time of Algorithm RETIME is $O(V^3)$. □

In order to show that Algorithm RETIME terminates with the right answer, it suffices to prove that if the retiming problem is feasible, then there exists a minimum solution $\overline{r}$ in which $\overline{r}(v) \leq 3|V| - 1$ for all $v \in V$, and that Algorithm RETIME computes that solution. In the following two lemmas, we prove that these conditions are met.

Lemma 39 *Suppose that the retiming problem defined by Inequalities (2.26), (2.27), and (2.28) has a solution. Then there exists a solution r such that $0 \leq r(v) \leq 3|V| - 1$ for all $v \in V$.*

Proof. We first show that under the conditions of the lemma, there exists a nonnegative solution. Let r be a solution that satisfies Inequalities (2.26), (2.27), and (2.28). Then, for any integer c, the function $r'(v) = r(v) + 2c$ for all $v \in V$, also satisfies these constraints, which can be seen by direct substitution into the three inequalities. Thus, if there is a solution to the retiming problem, by picking c large enough, we can find a nonnegative one.

Since there exists a nonnegative solution, there exists a nonnegative solution $\overline{r}$ which is minimum in the sense that for any other nonnegative solution r, we have $\overline{r}(v) \leq r(v)$ for all $v \in V$. We shall show that $\overline{r}(v) \leq 3|V| - 1$ for all $v \in V$.

Assume now, for the purpose of contradiction, that there exists a vertex $t \in V$ such that $\overline{r}(t) \geq 3|V|$. Under this assumption, we shall show that there exists a solution r' to Inequalities (2.26), (2.27), and (2.28) which is smaller than $\overline{r}$, thereby contradicting the minimality of $\overline{r}$.

Define the set $V_t \subseteq V$ to include t and all vertices that can reach t in G_r using only those edges $u \xrightarrow{e} v$ for which $\overline{r}(v) - \overline{r}(u) \leq 3$. Let p be an arbitrary simple path from a vertex $x \in V_t$ to t. Summing the inequalities $\overline{r}(v) - \overline{r}(u) \leq 3$ for every edge $u \xrightarrow{e} v$ along p, we obtain

$$\begin{aligned} \overline{r}(t) - \overline{r}(x) &\leq \sum_{u \xrightarrow{e} v \in p} 3 \\ &\leq 3\,(|V| - 1) \end{aligned}$$

since the sum telescopes and a simple path has at most $|V| - 1$ edges. By assumption, $\overline{r}(t) \geq 3|V|$, and hence, $\overline{r}(x) \geq 3$. Thus, all vertices in V_t are retimed by at least 3 in the minimal retiming $\overline{r}$.

Let us define a new function r' by the equation

$$r'(v) = \begin{cases} \overline{r}(v) & \text{if } v \in V - V_t\ ; \\ \overline{r}(v) - 2 & \text{if } v \in V_t\ . \end{cases} \tag{2.32}$$

The function r' is nonnegative, because $\overline{r}(v) \geq 0$ for all $v \in V - V_t$ and $\overline{r}(v) \geq 3$ for all

$v \in V_t$. Moreover, r' is nowhere larger than $\overline{r}$, and it is strictly smaller because $t \in V_t$ implies $r'(t) < \overline{r}(t)$. Thus, if we can show that r' satisfies Inequalities (2.26), (2.29), and (2.30), we obtain the contradiction we desire.

Let us compare the circuits $G_{r'}$ and $G_{\overline{r}}$. For an edge $u \xrightarrow{e} v$, we have three possible situations:

- $w_{r'}(e) = w_{\overline{r}}(e)$ if $u, v \in V_t$ or $u, v \in V - V_t$;
- $w_{r'}(e) = w_{\overline{r}}(e) + 2$ if $u \in V_t$ and $v \in V - V_t$;
- $w_{r'}(e) = w_{\overline{r}}(e) - 2$ if $u \in V - V_t$ and $v \in V_t$.

Observe that in the third case, $w_{r'}(e) \geq 0$, since the definition of V_t implies that $\overline{r}(v) - \overline{r}(u) \geq 4$, and thus,

$$\begin{aligned} w_{\overline{r}}(e) &= w(e) + \overline{r}(v) - \overline{r}(u) \\ &\geq w(e) + 4 \\ &\geq 4 \ . \end{aligned}$$

Thus, r' is a legal retiming.

We wish to show that if $G_{\overline{r}}$ is properly timed by a clocking scheme π, then so is $G_{r'}$. The circuit $G_{r'}$ differs from $G_{\overline{r}}$ by the addition of 2 latches on some edges and the subtraction of 2 latches on some other edges that have at least 4 latches on them in $G_{\overline{r}}$. By Lemma 16, the 2-latch subtraction does not affect proper timing. Adding a pair of latches on an edge does not affect Condition WF3 (clock phases alternate along paths), and by Lemma 14, increasing the number of latches on a path cannot violate proper timing. Thus, $G_{r'}$ is properly timed by π.

Thus, by Lemma 35, Inequalities (2.26), (2.29), and (2.30) hold for r', which contradicts the minimality of $\overline{r}$ and completes the proof. □

We conclude this section with the following theorem.

Theorem 40 *The retiming problem can be solved for a two-phase, level-clocked circuit* $G = \langle V, E, d, w, \chi \rangle$ *and a clocking scheme* $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$ *in* $O(V^3)$ *time.*

Proof. From the invariant in the proof of Theorem 34 we have that $r(v) \leq \overline{r}(v)$ for all $v \in V$, at every point during the execution of Algorithm RETIME. Since $\overline{r}(v) \leq 3|V| - 1$, according to Lemma 39, we conclude that whenever a variable $r(v)$ exceeds $3|V| - 1$ during Algorithm RETIME, the monotonic program must be infeasible. It follows that Algorithm RETIME computes the minimum retiming $\overline{r}$ or correctly discerns that no solution exists. The running time of Algorithm RETIME follows from Lemma 38. □

2.8 Retiming for Minimum Latch Count

In this section we consider the *retiming problem for minimum latch count*: Given a two-phase circuit $G = \langle V, E, d, w, \chi \rangle$ and a symmetric clocking scheme $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$, we wish to compute a retimed circuit G_r that is properly timed by π and uses the minimum number of latches. We show that this problem can be solved in $O(V^3 \lg V)$ time by reducing it to the dual of an uncapacitated minimum-cost flow problem.

The following lemma gives necessary and sufficient conditions for a retiming G_r to have the minimum number of latches.

Lemma 41 *Let $G = \langle V, E, d, w\rangle$ be a two-phase circuit, let $\pi = \langle \phi, \gamma, \phi, \gamma\rangle$ be a symmetric clocking scheme, and let $r : V \to \mathbf{Z}$ be a retiming function. Then, the retimed circuit G_r achieves π with the minimum number of latches under any other retiming if and only if the assignment r minimizes the expression*

$$\sum_{v\in V} (indegree(v) - outdegree(v))\, r(v)$$

subject to constraint (2.19)

$$r(u) - r(v) \le w(e)$$

for every edge $u \xrightarrow{e} v \in E$, and constraint (2.20)

$$r(u) - r(v) \le \sum_{i \xrightarrow{e} j \in p} \left(w(e) - \frac{2}{\pi} d(j)\right) + \left(2 - \frac{2}{\pi}(d(u) + \gamma)\right)$$

for every path $u \overset{p}{\rightsquigarrow} v$ in G.

Proof. According to Lemma 29, the circuit G_r is properly timed by π if and only if the two sets of constraints in the statement of the lemma are satisfied. Moreover, the number of latches in G_r is given by the expression

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

Therefore, the lemma holds. □

We can now prove the following theorem.

Theorem 42 *The retiming problem for minimum latch count can be solved in $O(V^3 \lg V)$ time.*

Proof. The retiming problem for minimum latch count is reduced to the dual of an uncapacitated minimum-cost flow problem [41] on the graph defined by Inequalities (2.19) and (2.20). The cost of each edge equals the right-hand side of the corresponding inequality. The demand/supply of each vertex v equals the difference between the number of edges coming into v and the number of edges coming out of v. We can use a scaling algorithm by Orlin to solve this problem that runs in $O(V^3 \lg U)$ steps, where U is the maximum demand/supply. In our retiming problem, we have $U \le |V|$, and therefore, the algorithm runs in $O(V^3 \lg V)$ time. □

We can give polynomial-time algorithms for several other retiming problems for minimum latch count. If every fanout wire of each gate in the circuit has the same output, then it can be shown that retiming for minimum number of latches with maximal latch-sharing can be solved in $O(V^3)$ time [50]. This algorithm is more efficient than the algorithm for the general problem, because it solves a minimum-cost flow problem on a graph with unit demands and supplies. If the objective is to achieve the minimum clock period with the minimum number of latches, we can find a retiming in $O(V^2E + V^3 \lg^2 V)$ time. First, we compute a set of $O(V^3)$ possible clock periods using the $O(V^2E)$-time Algorithm PIFDR, and then we binary search this set for the minimum feasible clock period.

2.9 Approximation Schemes for Minimum-Period Retiming

In this section we present "fully polynomial-time approximation schemes" for three problems related to both retiming and tuning. A *fully polynomial-time approximation scheme* [5] is an optimization algorithm that takes, in addition to its other input parameters, a parameter $\epsilon > 0$ specifying a relative error. The algorithm must produce an answer that is within $(1 + \epsilon)$ of the optimal answer for the problem and must run in time polynomial in the input and in $1/\epsilon$.

The first problem we consider is *retiming and fixed-duty-ratio tuning*: Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$, a real number $\rho \geq 0$, and gaps γ_0 and γ_1, we wish to compute a retiming function r and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$, such that G_r is properly timed by π, and π has the minimum period among all clocking schemes of duty-ratio ρ that can be achieved by retiming the circuit G. We give an $O(V^3 \lg(V/\epsilon))$-time algorithm that, for any given relative error $\epsilon > 0$, computes a retiming r and a clocking scheme π with duty ratio ρ, such that G_r is properly timed by π, and the period of π is at most $(1 + \epsilon)$ times the optimal period. The same algorithm can be used to solve the *retiming and fixed-duty-cycle tuning problem*, in which instead of the duty-ratio ρ we are given one of the phases ϕ_0 or ϕ_1.

The second problem we consider is *retiming and symmetric tuning*: Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$ and gap γ, compute a retiming function r and a symmetric clocking scheme $\pi = \langle \phi, \gamma, \phi, \gamma \rangle$, such that G_r is properly timed by π, and π has the minimum period among all symmetric clocking schemes with gap γ that can be achieved by retiming G. This problem is a special case of the retiming and fixed-duty-ratio tuning problem, and the same basic algorithm can be applied to yield a retiming r and a clocking scheme π with period at most $(1 + \epsilon)$ times the optimal period, for any given relative error $\epsilon > 0$. In this case, the algorithm terminates in $O((VE + V^2 \lg V) \lg(V/\epsilon))$ steps.

Finally, we consider the general *retiming and tuning problem*: Given a two-phase, level-clocked circuit $G = \langle V, E, d, w, \chi \rangle$ and gaps γ_0 and γ_1, we compute a retiming function r and a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1 \rangle$, such that G_r is properly timed by π, and π has the minimum period among all clocking schemes that can be achieved by retiming G. We give an $O(V^3(1/\epsilon) \lg(1/\epsilon) + (VE + V^2 \lg V) \lg(V/\epsilon))$-time algorithm that, for any given relative error $\epsilon > 0$, computes a retiming r and a clocking scheme π with period at most $(1 + \epsilon)$ times the optimal period.

2.9.1 Retiming and Fixed Duty-Ratio Tuning

Algorithm R&FDRT, given in Figure 2-12, approximately solves the retiming and fixed duty-ratio tuning problem using a binary search over a space of possible clock periods. Since the problem requires that only fixed duty-ratio clocking schemes be considered, each possible clock period corresponds to a unique clocking scheme. The algorithm checks whether this

R&FDRT$(G, \rho, \gamma_0, \gamma_1, \epsilon)$
$d_{\max} \leftarrow \max_{v \in V} d(v)$
$\pi^+ \leftarrow |V| d_{\max} + \gamma_0 + \gamma_1$
$\phi_0 \leftarrow (\pi^+ - \gamma_0 - \gamma_1)/(1+\rho)$
$\pi^- \leftarrow \gamma_0 + \gamma_1$
while $\pi^+ - \pi^- > \epsilon d_{\max}$
    do $\pi \leftarrow (\pi^+ + \pi^-)/2$
        $\phi'_0 \leftarrow (\pi - \gamma_0 - \gamma_1)/(1+\rho)$
        if RETIME$(G, \langle \phi'_0, \gamma_0, \rho\phi'_0, \gamma_1 \rangle) \neq$ **fail**
            then $\pi^+ \leftarrow \pi$
                $\phi_0 \leftarrow \phi'_0$
            else $\pi^- \leftarrow \pi$
$r \leftarrow$ RETIME$(G, \langle \phi_0, \gamma_0, \rho\phi_0, \gamma_1 \rangle)$
return π and r

Figure 2-12: Algorithm R&FDRT, which solves the retiming and fixed duty-ratio tuning problem. The algorithm takes as input a two-phase circuit $G = (V, E, d, w, \chi)$, a duty ratio ρ, gap widths γ_0, γ_1, and a relative error $\epsilon > 0$. It computes a retimed circuit G_r and a period π such that G_r is properly timed by a clocking scheme whose period is π, whose gap widths are γ_0, γ_1, and whose duty ratio is ρ. The period π is guaranteed to be at most $(1+\epsilon)$ times the period of any clocking scheme that can be achieved by G under retiming and whose duty ratio is ρ.

clocking scheme is achievable using Algorithm RETIME from Section 2.7. Binary search can be employed because if a given clock period is achievable, every greater clock period is also achievable.

Algorithm R&FDRT binary searches over a range of clock periods that is guaranteed to include the minimum clock period π^*. As a lower bound on π^*, the algorithm uses $\gamma_0 + \gamma_1$, which follows from Equation (2.1). As an upper bound, the algorithm uses the value $|V|\, d_{\max} + \gamma_0 + \gamma_1$, where $d_{\max}$ is the maximum delay of any individual functional element. To see that a clock period of $|V|\, d_{\max} + \gamma_0 + \gamma_1$ is always achievable for any duty ratio, we consider the inequalities in Lemma 18. For each inequality, we choose the largest left-hand side—at most $|V|\, d_{\max}$—and the smallest right-hand side—$w(p) = 1$ in Inequality (2.6), and $w(p) = 0$ in Inequality (2.7), and $w(c) = 2$ in Inequality (2.8). With the value $\pi = |V|\, d_{\max} + \gamma_0 + \gamma_1$, the inequalities are all satisfied.

It remains to show that the search terminates when it has isolated a sufficiently accurate approximation π of the true minimum π^*. The algorithm maintains that approximation π and the optimal clock period π^* both fall within an interval $[\pi^-, \pi^+]$. The search terminates when $\pi^+ - \pi^- \leq \epsilon\, d_{\max}$, at which point we have

$$\begin{aligned} \pi &\leq \pi^+ \\ &\leq \pi^- + \epsilon\, d_{\max} \\ &\leq \pi^* + \epsilon\, d_{\max} \\ &\leq \pi^* + \epsilon\, \pi^* \\ &\leq (1+\epsilon)\, \pi^* . \end{aligned}$$

```
R&ST(G, γ, ε)
d_max ← max_{v∈V} d(v)
π⁺ ← |V| d_max + 2γ
φ_0 ← π⁺/2 − γ
π⁻ ← 2γ
while π⁺ − π⁻ > ε d_max
      do π ← (π⁺ + π⁻)/2
         φ'_0 ← π/2 − γ
         if RwSCS(G, ⟨φ'_0, γ, φ'_0, γ⟩) ≠ fail
           then π⁺ ← π
                φ_0 ← φ'_0
           else π⁻ ← π
r ← RwSCS(G, ⟨φ_0, γ, φ_0, γ⟩)
return π and r
```

Figure 2-13: Algorithm R&ST, which solves the retiming and symmetric tuning problem. The algorithm takes as input a two-phase circuit $G = (V, E, d, w, \chi)$, a gap width γ, and a relative error $\epsilon > 0$. It computes a retimed circuit G_r and a period π such that G_r is properly timed by a symmetric clocking scheme whose period is π and whose gap widths are both γ. The period π is guaranteed to be at most $(1+\epsilon)$ times the period of any symmetric clocking scheme that can be achieved by retiming G.

Algorithm R&FDRT runs in $O(V^3 \lg(V/\epsilon))$ time, as can be seen by the following analysis. Lines 1–4 require only $O(V)$ time. Each execution of the **while** loop (lines 5–11) is dominated by the $O(V^3)$-time call to RETIME. The **while** loop is executed $O(\lg(V/\epsilon))$ times, since the range between the initial upper and lower bounds on π is $|V|\, d_{\max}$, and we continue the search until the range is $\epsilon\, d_{\max}$, dividing the range by 2 in every iteration.

Algorithm R&FDRT can be adjusted to find an exact solution when the propagation delays of the combinational elements are integers. Specifically, the retiming and fixed-duty-ratio tuning problem can be solved exactly in $O(V^3 \lg(V d_{\max}/\delta))$ time, where

$$\delta = \min\left\{ \frac{1}{3\,|V|-1}, \frac{\gamma_0}{2(3\,|V|-1)^2}, \frac{\gamma_1}{2(3\,|V|-1)^2} \right\} .$$

Using the integrality of the delays, it is straightforward to show that the clock period of any two clocking schemes defined by Inequalities (2.11) and (2.12) must differ by at least δ. The proof relies on the fact that the optimal clocking scheme is defined by the intersection of Equation (2.1) with one of the lines defined by Inequalities (2.11), (2.12), and (2.8).

2.9.2 Retiming and Symmetric Tuning

The retiming and symmetric tuning problem is a special case of the retiming and fixed duty-ratio problem, and therefore, it can be solved with Algorithm R&FDRT in $O(V^3 \lg(V/\epsilon))$ time. We can solve this special case in $O((VE + V^2 \lg V) \lg(V/\epsilon))$ time using Algorithm R&ST shown in Figure 2-13.

The new algorithm R&ST(G, γ, ϵ), takes as arguments the circuit G, a value γ for the gaps between the two phases, and a relative error ϵ. It is essentially identical to Algorithm R&FDRT, except that the calls to Algorithm RETIME are replaced by calls to Algorithm RWSCS. Since retiming with symmetric clocking schemes can be accomplished in $O(VE + V^2 \lg V)$ time, the same analysis as for Algorithm R&FDRT yields a running time of $O((VE + V^2 \lg V) \lg(V/\epsilon))$ for Algorithm R&ST.

Algorithm R&ST can be adjusted to find an exact solution when the propagation delays of the combinational elements are integers. Specifically, the retiming and symmetric tuning problem can be solved exactly in $O((VE + V^2 \lg V) \lg(V d_{\max}/\delta))$ time, where

$$\delta = \min\left\{\frac{1}{3|V|-1}, \frac{\gamma_0}{2(3|V|-1)^2}, \frac{\gamma_1}{2(3|V|-1)^2}\right\} .$$

The proof relies on the integrality of the delays and the fact that the optimal clocking scheme is given by the intersection of Equation (2.1) with one of the lines defined by Inequalities (2.11), (2.12), and (2.8).

2.9.3 Retiming and Tuning

We now describe a polynomial-time approximation scheme for the general retiming and tuning problem. Algorithm GR&T, shown in Figure 2-14, computes a retimed circuit G_r and a clocking scheme π such that G_r is properly timed by π and the period of π is at most $(1+\epsilon)$ times the minimum clock period π^* that can be achieved by any retiming G_r. The algorithm runs in $O(V^3(1/\epsilon)\lg(1/\epsilon) + (VE + V^2 \lg V)\lg(V/\epsilon))$ time.

Algorithm GR&T proceeds as follows. In the beginning, it employs Algorithm R&ST to compute a symmetric clocking scheme π^+ whose period is an upper bound on π^*. Specifically, π^+ satisfies $\pi_s^* \le \pi^+ \le (1+\epsilon)\pi_s^*$, where π_s^* is the period of the shortest symmetric clocking scheme with gaps $\gamma = \max\{\gamma_0, \gamma_1\}$ that can be achieved by a retimed circuit G_r. Subsequently, Algorithm GR&T computes a lower bound π^- on the optimal clock period π^*. The bounds π^+ and π^- are then updated in the body of the outer **while** loop (lines 7–22), as long as they differ by more than the constant δ computed in line 6. Each time through the loop, the algorithm searches for a retiming r such that G_r is properly timed by a clocking scheme of period $\pi = (\pi^+ + \pi^-)/2$. This search is conducted in the inner **while** loop (lines 11–19), which considers clocking schemes of period π with ϕ_0 assuming values in increments of 3δ from the interval $(0, \pi - \gamma_0 - \gamma_1)$. If a retimed circuit G_r is properly timed by one of these clocking schemes, then the upper bound π^+ assumes the value π (lines 16–19). If no retiming of period π exists among the sampled values for ϕ_0, however, then the lower bound π^- is increased to π (lines 20–22).

The linear search for ϕ_0 in increments of 3δ (lines 11–19) is a key part of Algorithm GR&T. The following lemma, which is illustrated in Figure 2-15, is used in the correctness proof of Algorithm GR&T. The lemma considers a feasible clocking scheme for a circuit. If the clock period is lengthened by an amount Δ, the lemma shows that there is a range of width 3Δ of values for ϕ_0 such that proper timing is unaffected.

Lemma 43 *Let $G = \langle V, E, d, w, \chi\rangle$ be a 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'. Then, for any $\Delta \ge 0$ and ϕ_0 in the range $\phi_0' - \Delta \le \phi_0 \le \phi_0' + 2\Delta$ such that $0 < \phi_0 < \pi - \gamma_0 - \gamma_1$, the circuit G is properly timed by the clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$, where $\pi = \pi' + \Delta$ and $\phi_1 = \pi - \gamma_0 - \gamma_1 - \phi_0$.*

```
GR&T(G, \gamma_0, \gamma_1, \epsilon)
d_max <- max_{v \in V} d(v)
\gamma <- max{\gamma_0, \gamma_1}
\pi^+ <- period of clocking scheme returned by R&ST(G, \gamma, \epsilon)
\phi_0^+ <- duty cycle of phase 0 in clocking scheme returned by R&ST(G, \gamma, \epsilon)
\pi^- <- max{\pi^+/2(1 + \epsilon), \gamma_0 + \gamma_1}
\delta <- \pi^- \epsilon/2
while \pi^+ - \pi^- > \delta
      do \pi <- (\pi^+ + \pi^-)/2
         \phi_0 <- min{2\delta, (\pi - \gamma_0 - \gamma_1)/2} - 3\delta
         feasible <- FALSE
         while (\pi - \gamma_0 - \gamma_1) - \phi_0 > \delta and feasible = FALSE
              do   |> Begin linear search over \phi_0 for feasible (\phi_0, \pi).
                 \phi_0 <- min{\phi_0 + 3\delta, \pi - \gamma_0 - \gamma_1 - \delta}
                 \phi_1 <- \pi - \gamma_0 - \gamma_1 - \phi_0
                 if RETIME(G, <\phi_0, \gamma_0, \phi_1, \gamma_1>) \neq fail
                   then   |> Found feasible (\phi_0, \pi).
                          feasible <- TRUE
                          \pi^+ <- \pi
                          \phi_0^+ <- \phi_0
         if feasible=FALSE
           then   |> \pi is not feasible for any \phi_0.
                \pi^- <- \pi
\phi_1^+ <- \pi^+ - \gamma_0 - \gamma_1 - \phi_0^+
r <- RETIME(G, <\phi_0^+, \gamma_0, \phi_1^+, \gamma_1>)
return <\phi_0^+, \gamma_0, \phi_1^+, \gamma_1> and r
```

Figure 2-14: Algorithm GR&T for solving the general retiming and tuning problem. The algorithm takes as input a two-phase circuit G, gap widths γ_0 and γ_1, and a relative error $\epsilon > 0$. It computes a retimed circuit G_r and a clocking scheme π such that G_r is properly timed by π and the period of π is less than $(1+\epsilon)$ times the period of any clocking scheme that can be achieved by retiming G.

Proof. Since G is properly timed by π', it satisfies Inequalities (2.2) and (2.3) in Lemma 14 for the clocking scheme π'. In order to prove that G is properly timed by π, we shall show that G also satisfies the constraints in Lemma 14 for the clocking scheme π.

Inequality (2.2) implies that for every path $u \overset{p}{\leadsto} v$ in G_r with $\chi(u) \neq \chi(v)$ and $\chi(u) = 0$, we must have

$$\begin{aligned} d(p) &\leq \pi'\left(\frac{1+w(p)}{2}\right) + \phi_0' \\ &= \pi\left(\frac{1+w(p)}{2}\right) - (\pi - \pi')\left(\frac{1+w(p)}{2}\right) + \phi_0' \\ &\leq \pi\left(\frac{1+w(p)}{2}\right) - \Delta + \phi_0' \end{aligned}$$

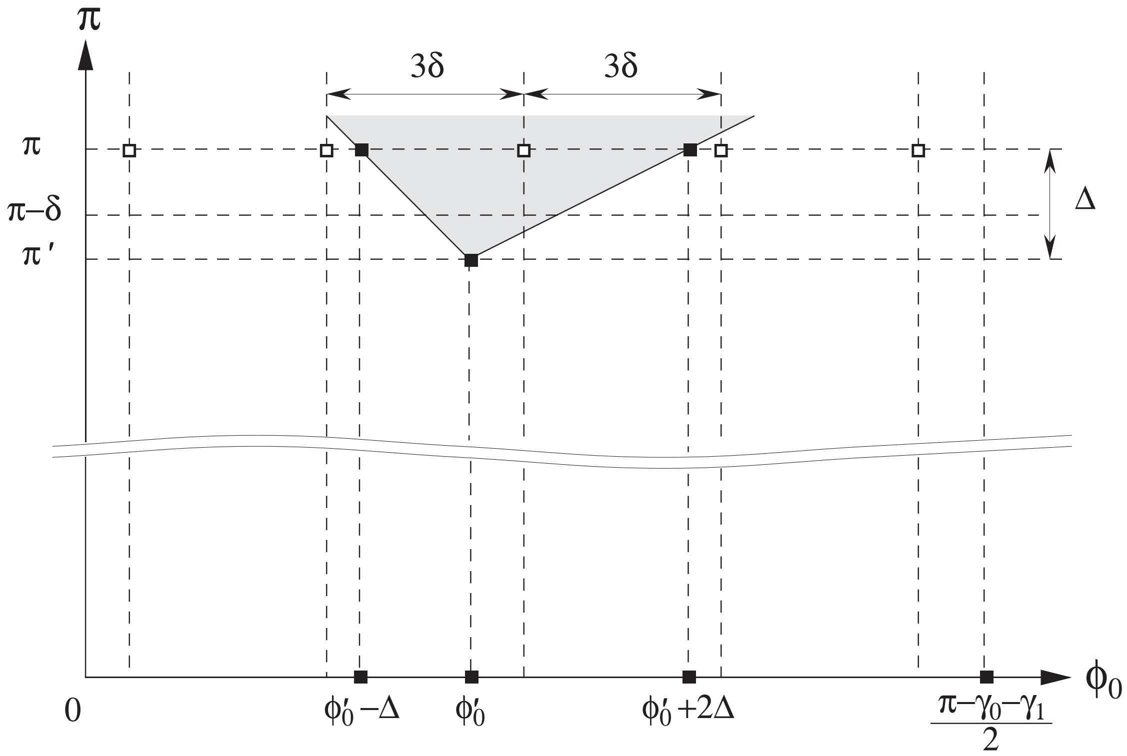


Figure 2-15: Illustration of the importance of Lemma 25 for the linear search in Algorithm GR&T. The white rectangles denote the points of the (ϕ_0, π)-plane that are visited during the linear search with a specified π. If a circuit G is properly timed by a clocking scheme $\langle \phi_0', \gamma_0, \phi_1', \gamma_1 \rangle$, then it is properly timed by every clocking scheme in the shaded area. The line with slope -1 is the steepest among the constraints for π' on paths $u \rightsquigarrow v$ with $\chi(u) = 0$ and $\chi(v) = 1$. The line with slope $1/2$ is the steepest among the constraints for π' on paths $u \rightsquigarrow v$ with $\chi(u) = 1$ and $\chi(v) = 0$. If $\pi' \leq \pi - \delta$, then at least one of the white rectangles lies in the shaded area, and the circuit G is properly timed at that point.

$$\leq \quad \pi \left(\frac{1 + w(p)}{2} \right) + \phi_0 \ ,$$

since $\pi \geq \pi'$, $w(p) \geq 1$, and $\phi_0' - \Delta \leq \phi_0$. When $\chi(u) = 1$, we can show in a similar way and using the inequality $\phi_0 \leq \phi_0' + 2\Delta$ that G still satisfies the path constraints for π. Therefore, G satisfies Inequality (2.2) for π.

The constraints described by Inequality (2.3) depend on the length of the clock period and not on the duty cycles of the particular phases. By increasing the clock period from π' to π these constraints are still satisfied.

Since G satisfies Inequalities (2.2) and (2.3) for π, Lemma 14 implies that it is properly timed by π. □

Now, we can prove the correctness of Algorithm GR&T.

Lemma 44 *Algorithm* GR&T *computes a solution to the general retiming and tuning problem whose relative error is at most* ϵ.

Proof. Algorithm GR&T maintains two invariants during its execution. We shall use these invariants to show that Algorithm GR&T returns a clocking scheme π such that $\pi \leq (1+\epsilon)\pi^*$, where π^* is the minimum clock period that can be achieved by any retiming G_r.

The first property that remains invariant during the execution of Algorithm GR&T is that there exists a retimed circuit G_r that is properly timed by $\pi^+ = \langle \phi_0^+, \gamma_0, \phi_1^+, \gamma_1 \rangle$. Initially, the clocking scheme π^+ is essentially the symmetric scheme returned by Algorithm R&ST, except that one of the gaps may be smaller. Shortening a gap does not affect the proper timing of G_r, however, as long as the clock period remains the same. Subsequently, the parameters of the clocking scheme π^+ are updated in lines 18–19, in which case the algorithm has found a retiming r such that G_r is properly timed by the new clocking scheme. Therefore, the invariant is maintained throughout the execution of the algorithm.

The second property that remains invariant during the execution of Algorithm GR&T is that $\pi^- \leq \pi^* + \delta$. Initially, π^- assumes the value $\max\{\pi^+/2(1+\epsilon), \gamma_0 + \gamma_1\}$, where π^+ is the period of the optimal symmetric clocking scheme within a relative error ϵ. We can show that this value is a lower bound for π^* as follows. The term $\gamma_0 + \gamma_1$ is a lower bound for π^* by definition of the clock period. For the term $\pi^+/2(1+\epsilon)$, if we let π_s^* be the period of the optimal symmetric clocking scheme with gaps $\gamma = \max\{\gamma_0, \gamma_1\}$, then since we initially have $\pi^+ \leq \pi_s^*(1+\epsilon)$, it follows that

$$\begin{aligned} \pi^+/2(1+\epsilon) &\leq \pi_s^*(1+\epsilon)/2(1+\epsilon) \\ &\leq \pi_s^*/2 \ . \end{aligned}$$

Moreover, we have

$$\pi_s^*/2 \leq \pi^* \ ,$$

since if $\pi_s^*/2 > \pi^*$, then by extending the shorter gap and the shorter duty cycle in π^* to become equal to the longer gap and duty cycle respectively, we would obtain a feasible symmetric clocking scheme with period at most $2\pi^* < \pi_s^*$, thus contradicting the optimality of π_s^*. Therefore, we initially have $\pi^- \leq \pi^* \leq \pi^* + \delta$, and the invariant holds. During the execution of the algorithm, π^- is updated in line 22 whenever the linear search yields no retimed circuit G_r that achieves a clock period $\pi = (\pi^+ + \pi^-)/2$. The linear search is performed in increments of 3δ, and if it fails to find any feasible point, Lemma 43 guarantees that any feasible clocking scheme has period at least $\pi - \delta$. Consequently, the optimal clock period π^* must satisfy $\pi^* \geq \pi - \delta$, and since $\pi > \pi^-$, it follows that $\pi^* + \delta \geq \pi^-$, and the invariant still holds.

The boundary conditions of the linear search are established in lines 9 and 13 of the code. The duty cycle ϕ_0 is initialized to a value such that the predicate in line 11 is always satisfied and the inner loop is executed at least once. In the first iteration of the inner loop, we have $0 < \phi_0 \leq 2\delta$ and $\phi_1 > 0$. The linear search proceeds in increments of 3δ, and it continues until ϕ_0 attains a value such that $(\pi - \gamma_0 - \gamma_1) - \delta \leq \phi_0 < \pi - \gamma_0 - \gamma_1$.

The Algorithm GR&T terminates when $\pi^+ - \pi^- \leq \delta$ and returns a clock period π that satisfies

$$\begin{aligned} \pi &\leq \pi^+ \\ &\leq \pi^- + \delta \end{aligned}$$

$$\begin{aligned} &\le \pi^* + 2\delta \\ &\le \pi^* + \pi^*\epsilon \\ &\le (1+\epsilon)\pi^* \ . \end{aligned}$$

Therefore, Algorithm GR&T returns a clocking scheme whose period is within a relative error of ϵ of the optimal period. □

Theorem 45 *Algorithm* GR&T *can solve the general retiming and tuning problem for a two-phase, level-clocked circuit* $G = \langle V, E, d, w, \chi \rangle$ *with relative error* ϵ *in* $O(V^3(1/\epsilon)\lg(1/\epsilon) + (VE + V^2 \lg V)\lg(V/\epsilon))$ *time.*

Proof. The correctness of Algorithm GR&T follows from Lemma 44. It remains to show that it terminates in $O(V^3(1/\epsilon)\lg(1/\epsilon) + (VE + V^2\lg V)\lg(V/\epsilon))$ steps.

Lines 1–4 of the algorithm complete in $(VE + V^2 \lg V)\lg(V/\epsilon))$ time. The external **while** loop (lines 7–22) performs a binary search in the interval $[\pi^-, \pi^+]$. The resolution of the search is $\delta = \pi^-\epsilon/2$, and therefore, the number of potential periods is

$$\begin{aligned} \frac{\pi^+ - \pi^-}{\pi^-\epsilon/2} &\le \frac{\pi^+ - (\gamma_0 + \gamma_1)}{((\pi^+ - \gamma_0 - \gamma_1)/2(1+\epsilon))(\epsilon/2)} \\ &= \frac{4(1+\epsilon)}{\epsilon} \\ &= O\left(1/\epsilon\right) \ . \end{aligned}$$

Thus, the binary search causes lines 7–22 to be executed $O(\lg(1/\epsilon))$ times. The internal **while** loop (lines 11–19) performs a linear search in the interval $(0, (\pi - \gamma_0 - \gamma_1)/2)$ in increments of 3δ. In this case, the number of points checked is at most

$$\frac{(\pi - \gamma_0 - \gamma_1)/2}{3((\pi^+ - \gamma_0 - \gamma_1)/2(1+\epsilon))(\epsilon/2)} = \frac{4}{3}\left(\frac{1+\epsilon}{\epsilon}\right) \ .$$

Therefore, for each π, lines 11–19 are executed $O(1/\epsilon)$ times. Line 15 terminates in $O(V^3)$ time, and therefore, each iteration of lines 7–22 in Algorithm GR&T requires $O(V^3(1/\epsilon))$ time. Combining the time bounds, we conclude that Algorithm GR&T terminates in $O(V^3(1/\epsilon)\lg(1/\epsilon) + (VE + V^2 \lg V)\lg(V/\epsilon))$ time. □

The practical efficiency of Algorithm GR&T can be improved by updating δ in the **then** clause of line 22 immediately after π^- has been updated with its new value. As π^- increases, the resolution δ of the linear search decreases, and thus, fewer points are checked. The worst-case running time of the algorithm does not change, however. Although the linear search is inefficient, we do not know how to replace it by a more efficient binary search. The reason for this difficulty is that whenever a particular clock period is not achievable for a specific ϕ_0, we do not know how to tell whether the duty cycle of phase 0 should become shorter or longer.

POLYR&FDRT$(G, \rho, \gamma_0, \gamma_1)$
1 Use Algorithm PIFDR to compute the set $\Pi(\rho)$ of possible periods.
2 Sort the elements in $\Pi(\rho)$.
3 Use Algorithm RETIME to binary search the elements of $\Pi(\rho)$ for the minimum period π that can be achieved by a retiming r.
4 **return** π and r.

Figure 2-16: Algorithm POLYR&FDRT for the retiming and fixed-duty-ratio tuning problem. The algorithm takes as input a circuit $G = \langle V, E, d, w, \chi \rangle$, a duty ratio ρ, and two gap widths γ_0 and γ_1. It computes a retimed circuit G_r and a period π such that G_r is properly timed by a clocking scheme whose period is π, whose gap widths are γ_0, γ_1, and whose duty ratio is ρ. The period π is guaranteed to be the minimum period over all clocking schemes that can be achieved by G after retiming and tuning with fixed duty-ratio ρ.

2.10 Polynomial-Time Algorithms for Minimum-Period Retiming

In this section we present polynomial-time algorithms for three minimum-period retiming problems. We have already investigated these three problems in Section 2.9, but the algorithms we described in that section were fully polynomial-time approximation schemes. The polynomial-time algorithms in this section compute exact solutions to these problems. We first give an $O(V^2E + V^3 \lg V)$-time algorithm for the retiming and fixed-duty-ratio tuning problem. We adapt this algorithm to solve the retiming and symmetric tuning problem in $O(V^2E)$ time. Finally, we give an $O(V^{11})$-time algorithm for the general retiming and tuning problem. This algorithm is interesting only from a theoretical perspective, however, because the degree of the polynomial in its running time renders it impractical for large circuits.

2.10.1 Retiming and Fixed-Duty-Ratio Tuning

Algorithm POLYR&FDRT, shown in Figure 2-16, solves the retiming and fixed-duty-ratio tuning problem by binary searching a set $\Pi(\rho)$ of $O(V^3)$ possible clock periods. The set $\Pi(\rho)$ is guaranteed to contain the minimum clock period that can be achieved by G after retiming and tuning with fixed duty-ratio ρ. Step 1 computes $\Pi(\rho)$ using Algorithm PIFDR which is described in Subsection 2.10.5. Step 2 sorts the elements in $\Pi(\rho)$, and Step 3 performs a binary search over the potential periods in order to identify the optimal. The binary search of $\Pi(\rho)$ is possible, because if a clock period π cannot be achieved by retiming, then no clock period $\pi' < \pi$ can be achieved, for a fixed duty-ratio ρ. Algorithm RETIME is used as a subroutine in the search to test whether a potential clock period can be achieved by retiming.

Algorithm POLYR&FDRT runs in $O(V^2E+V^3 \lg V)$ time. The computation of $\Pi(\rho)$ by Algorithm PIFDR can be performed in $O(V^2E)$ time, as it is shown in Subsection 2.10.5. The $O(V^3)$ elements of $\Pi(\rho)$ can be sorted in $O(V^3 \lg V)$ time. The binary search takes $O(V^3 \lg V)$ time, since Algorithm RETIME runs in $O(V^3)$ time. Therefore, Algorithm POLYR&FDRT terminates in $O(V^2E + V^3 \lg V)$ steps.

PolyR&ST(G, γ)
1 Use Algorithm PiFDR to compute the set $\Pi(\rho)$ of possible periods.
2 Use Algorithm RwSCS to perform a median-based binary search of the elements in $\Pi(\rho)$ for the minimum period π that can be achieved by a retiming r.
3 **return** π and r.

Figure 2-17: Algorithm PolyR&ST for the retiming and symmetric tuning problem. The algorithm takes as input a circuit $G = \langle V, E, d, w, \chi \rangle$ and a gap width γ. It computes a retimed circuit G_r and a period π such that G_r is properly timed by a symmetric clocking scheme whose period is π, and whose gap widths are γ. The period π is guaranteed to be the minimum period over all symmetric clocking schemes that can be achieved by G under retiming.

2.10.2 Retiming and Symmetric Tuning

The retiming and symmetric tuning problem is a special case of the retiming and fixed-duty-ratio problem in which only symmetric clocking schemes are considered. This problem can be solved in $O(V^2E)$ time using Algorithm PolyR&ST shown in Figure 2-17.

Algorithm PolyR&ST takes as arguments the circuit G and a value γ for the gaps between the two phases. The computation of the set $\Pi(\rho)$ in Step 1 is identical to the computation in Step 1 of Algorithm PolyR&FDRT. In Step 2, Algorithm RwSCS is used as a subroutine to binary search $\Pi(\rho)$ for the minimum period that can be achieved by retiming G. Contrary to Algorithm PolyR&FDRT, however, the elements of $\Pi(\rho)$ are not sorted, because the $O(V^3 \lg V)$ time required to sort its $O(V^3)$ elements would not allow us to achieve the $O(V^2E)$ running time. Instead, the binary search is performed by computing the median of the periods still under consideration at each iteration of the search.

Algorithm PolyR&ST runs in $O(V^2E)$ time. The computation of $\Pi(\rho)$ in Step 1 requires $O(V^2E)$ time. Each iteration of the binary search in Step 2 requires $O(VE + V^2 \lg V)$ time. Since the median of a set of n elements can be found in $O(n)$ time [2, 10, 18], and since the binary search halves the number of periods under consideration at each iteration, Step 2 completes in

$$\begin{aligned}\sum_{i=0}^{O(\lg V)} \left(O\left(\frac{V^3}{2^i}\right) + O(VE + V^2 \lg V) \right) &= O(V^3 + (VE + V^2 \lg V) \lg V) \\ &= O(V^3 + VE \lg V)\end{aligned}$$

steps. Therefore, Algorithm PolyR&ST terminates in $O(V^2E) + O(V^3 + VE \lg V) = O(V^2E)$ time.

2.10.3 Retiming and Tuning

Algorithm PolyR&T, shown in Figure 2-18, solves the retiming and tuning problem using a linear search over a space Π of possible duty-cycle/period pairs. Step 1 employs Algorithm Pi to compute a set Π of $O(V^8)$ pairs (ϕ_0, π) that is guaranteed to contain a pair corresponding to an optimal clocking scheme that can be achieved by retiming G. Algorithm Pi is described in Subsection 2.10.4. Step 3 performs a linear search over the

POLYR&T(G, γ_0, γ_1)
1 Use Algorithm PI to compute the set Π of possible (ϕ_0, π) pairs.
2 Use Algorithm RETIME to search the elements in Π for the minimum period π that can be achieved by a retiming r.
3 **return** π and r.

Figure 2-18: Algorithm POLYR&T for the retiming and tuning problem. The algorithm takes as input a circuit $G = \langle V, E, d, w, \chi \rangle$ and gap widths γ_0, γ_1. It computes a retimed circuit G_r and a period π such that G_r is properly timed by a clocking scheme whose period is π, and whose gap widths are γ_0, γ_1. The period π is guaranteed to be the minimum period over all clocking schemes that can be achieved by G under retiming.

elements of Π in order to identify the optimal. Algorithm RETIME is used as a subroutine in the search to test whether a potential clocking scheme can be achieved by retiming. Even though this linear search is inefficient, we do not know of a way to replace it by a binary search, beacause if a period π is not achievable we do not know how to tell whether ϕ_0 should be made longer or shorter.

Algorithm POLYR&T runs in $O(V^{11})$ time. The computation of Π by Algorithm PI can be performed in $O(V^8)$ time, as it is shown in Subsection 2.10.4. The linear search takes $O(V^{11})$ time, since Algorithm RETIME runs in $O(V^3)$ time, and there are $O(V^8)$ elements in Π. Therefore, Algorithm POLYR&T terminates in $O(V^{11})$ steps.

2.10.4 Possible Periods for Retiming and Tuning

In this section we describe Algorithm PI that constructs the set Π of pairs (ϕ_0, π) that is used by Algorithm POLYR&T. We first explain the derivation of Π based on the conditions for proper timing that must hold for every retiming of the original circuit G. Then, we describe the operation of Algorithm PI, and we argue that it terminates in $O(V^8)$ time. Finally, we prove its correctness by showing that the pair (ϕ_0, π) corresponding to the minimum-period clocking scheme that can be achieved by retiming G is included among the $O(V^8)$ elements of Π.

The construction of Π is based on Lemma 18 which states that a given two-phase, level-clocked circuit G is properly timed by a given clocking scheme π if and only if Inequalities (2.6), (2.7), and (2.8) hold. The set Π is obtained by considering the Inequalities (2.6), (2.7), and (2.8) for every possible retiming of G. The intersections of the lines defined by all these inequalities are guaranteed to include all optimal pairs (ϕ_0, π). This approach is similar to the approach taken for the clock tuning problem. There are two differences, however. First, the number of intersections increases when tuning is combined with retiming, because the relocation of latches introduces new constraints. Second, the intersections formed between constraints that correspond to different retimings are redundant. Every optimal point that can be achieved for some retiming, however, is included in Π.

Algorithm PI is shown in Figure 2-19. For every constraint that is significant for the proper timing of some retiming of G, the set $\mathcal{C}$ holds its corresponding line on the (ϕ_0, π) plane. In Step 2 the set $\mathcal{C}$ is augmented by the single lower bound on π that holds for every retiming of G and is given in Inequality (2.16). Step 3 is similar to Step 2 in Algorithm TV.

PI(G, γ_0, γ_1)

1. Initialize $\mathcal{C}$ to be the empty set of lines on the (ϕ_0, π)-plane.
2. Augment $\mathcal{C}$ by the line corresponding to the single lower bound on the clock period π that is defined in Lemma 27.
3. For each $u \in V$, compute $DD(u, v, i)$ for all $v \in V$ and $i = 0, 1, \ldots, 3|V| - 3$, from the recurrence
$$DD(u, v, i) = d(v) + \max\left\{DD(u, x, i - w(e)) : x \xrightarrow{e} v \text{ and } i \geq w(e)\right\} .$$
4. For each $u, v \in V$, and for $0 \leq i \leq 6|V| - 2, 0 \leq i + l \leq 3|V| - 3, j = 0, 1$, augment $\mathcal{C}$ by the line corresponding to the following constraint:
$$\begin{aligned} \pi &\geq \frac{DD(u, v, i) - \phi_j}{(1 + i + l)/2} \quad \text{if } i + l \text{ is odd;} \\ \pi &\geq \frac{DD(u, v, i) - \gamma_j}{(2 + i + l)/2} \quad \text{if } i + l \text{ is even.} \end{aligned}$$
5. Return $\Pi = \{(\phi_0, \pi) : (\phi_0, \pi) \text{ is an intersection between two lines in } \mathcal{C}\}$.

Figure 2-19: Algorithm PI takes a circuit G, and two gap widths γ_0 and γ_1. The algorithm runs in $O(V^8)$ time and computes a set Π of $O(V^8)$ pairs (ϕ_0, π) that include the minimum-period clocking scheme that can be achieved by retiming G.

First, we compute a topological sort of all edges $e \in E$ with $w(e) = 0$. We then execute a triply nested loop that computes the longest propagation delay between every pair of vertices $u, v \in V$, for paths with up to $3|V| - 3$ latches. The outer loop is indexed by u, the middle loop is indexed by i, and the inner loop is indexed by each $e \in E$ consistent with the topological sort order if $w(e) = 0$ and in any order if $w(e) > 0$. Step 4 computes $O(V^2)$ constraints for each pair of vertices $u, v \in V$. Finally, Step 5 computes and returns the $O(V^8)$ intersections among the $O(V^4)$ constraints.

We now prove a bound on the running time of Algorithm PI.

Lemma 46 *Algorithm* PI *terminates in* $O(V^8)$ *time.*

Proof. The initialization of Step 1 is computed in $O(1)$ time. Step 2 terminates in $O(VE)$ time according to Lemma 27. In Step 3, the topological sort requires $O(E)$ time. In the triply nested loop, u takes on $O(V)$ values, i takes on $O(V)$ values, and e takes on $O(E)$ values. Thus, the total number of steps for Step 3 is $O(V^2E)$. Step 4 requires $O(V^4)$ time. Finally, the $O(V^8)$ intersections in Step 5 are computed in $O(V^8)$ time. Thus, the total running time of Algorithm PI is $O(V^8)$. □

We prove the correctness of Algorithm PI in two lemmas.

Lemma 47 *Let* $G = \langle V, E, d, w, \chi \rangle$ *be a two-phase, level-clocked circuit. Then Inequality (2.8) can be reduced to a single constraint that holds for any retimed circuit* G_r.

Proof. According to Lemma 27, the constraints defined by Inequality (2.8) can be reduced to the single lower bound given by Inequality (2.16), for any given retiming of the original circuit G. Moreover, these constraints remain unaltered for every retiming of G, because retiming does not change the propagation delay or the number of latches around any cycle in the circuit. Therefore, Inequality (2.8) can be reduced to a single constraint that holds for any retiming of G. □

Lemma 48 *Let $G = \langle V, E, d, w, \chi \rangle$ be a two-phase, level-clocked circuit. Then Inequalities (2.6) and (2.7) lead to $O(V^4)$ constraints that are significant for the proper timing of any retimed circuit G_r.*

Proof. We shall first prove the lemma for Inequality (2.6). Consider a fixed retiming G_r of the original circuit G and a pair of vertices $u, v \in V$ with $\chi_r(u) = 0$ and $\chi_r(v) = 1$; the situation with $\chi_r(u) = 1$ and $\chi_r(v) = 0$ is similar. Inequality (2.6) applies in this case, and for every simple path $u \overset{p}{\rightsquigarrow} v$ in G_r, we have

$$d(p) \le \pi\left(\frac{1 + w(p) + r(v) - r(u)}{2}\right) + \phi_0 \ . \tag{2.33}$$

For a fixed $w(p)$, there may exist an exponential number of such constraints, since the number of simple paths $u \overset{p}{\rightsquigarrow} v$ with initially $w(p)$ latches on them may be exponential. We can reduce these exponentially many constraints down to a single constraint by considering the simple path $u \overset{q}{\rightsquigarrow} v$ with the longest delay. Therefore, all simple paths $u \overset{p}{\rightsquigarrow} v$ in G_r with $w(p) = i$ must satisfy the single tightest constraint

$$DD(u, v, i) \le \pi\left(\frac{1 + w(p) + r(v) - r(u)}{2}\right) + \phi_0 \ , \tag{2.34}$$

where the longest path delay $DD(u, v, i)$ is defined as

$$DD(u, v, i) = \max\{d(p) : p \text{ is a path from } u \text{ to } v, \text{ and } w(p) = i\} \ , \tag{2.35}$$

for every pair of vertices $u, v \in V$. Note that the longest propagation delay $DD(u, v, i)$ is computed over *all* paths $u \overset{p}{\rightsquigarrow} v$ in G_r with $w(p) = i$, instead of just over the simple paths between the two vertices. We use this definition of $DD(u, v, i)$ for efficieny reasons: computing $DD(u, v, i)$ as defined requires $O(V^2 E)$ steps, whereas computing $DD(u, v, i)$ over simple paths is an intractable problem, since even finding a simple path with any given number of edges is intractable [13]. It is straightforward to verify, however, that no incorrect or redundant constraints are generated when $DD(u, v, i)$ is defined as in Equation (2.35). If $DD(u, v, i)$ is attained for a simple path, then Inequality (2.34) is one of the constraints defined by Inequality (2.33). If $DD(u, v, i)$ is attained for a path p that is not simple, however, then we can show that Inequality (2.34) can be also derived from a set of simple paths and simple cycles. Let p consist of a simple path $u \overset{p'}{\rightsquigarrow} v$ and simple cycles $c_1, \ldots, c_n$, for some n. If the circuit is properly timed, then from Inequality (2.6) we have

$$d(p') \le \pi\left(\frac{1 + w(p') + r(v) - r(u)}{2}\right) + \phi_0 \ ,$$

and from Inequality (2.8) we have

$$d(c_i) \leq \pi \left(\frac{w(c_i)}{2} \right) ,$$

for $i = 1, \ldots, n$. Inequality (2.34) follows immediately by adding these inequalities by parts, and therefore, it is not a redundant or incorrect constraint.

We now show that for a given pair of vertices $u, v \in V$ there are $O(V^2)$ constraints that determine the clock period for *any* retiming of the original circuit and for any initial number of latches on any path $u \rightsquigarrow v$. According to Corollary 17, every simple path with more than $3|V|-3$ latches does not determine the clock period of a two-phase circuit. Moreover, Lemma 39 states that for every circuit G that can be retimed to achieve a specific clock period, there exists a minimum retiming r that assigns integers in the interval $[0, 3|V|-1]$. Therefore, every path with more than $6|V|-2$ latches initially can be ignored, and every retiming that leaves more than $3|V|-3$ latches on a path can be ignored. Thus, after rearranging, for each pair of vertices $u, v \in V$ we are left with the $O(V^2)$ constraints

$$\pi \geq \frac{DD(u,v,i) - \phi_0}{(1 + i + r(v) - r(u))/2} , \tag{2.36}$$

where $0 \leq i \leq 6\,|V| - 2$, $0 \leq i + r(v) - r(u) \leq 3\,|V| - 3$, and $i + r(v) - r(u)$ is odd. Since there are $O(V^2)$ such pairs, we have a total of $O(V^4)$ constraints.

The constraints corresponding to Inequality (2.7) can be handled in a similar way. For each pair of vertices $u, v \in V$ with $\chi_r(u) = \chi_r(v) = 0$, we have the $O(V)$ constraints

$$\pi \geq \max_i \frac{DD(u,v,i) - \gamma_1}{(2 + i + r(v) - r(u))/2} , \tag{2.37}$$

where $0 \leq i \leq 6\,|V| - 2$, $0 \leq i + r(v) - r(u) \leq 3\,|V| - 3$, and $i + r(v) - r(u)$ is even. The situation with $\chi_r(u) = \chi_r(v) = 1$ is similar. Since there are $O(V^2)$ such pairs, we have a total of $O(V^3)$ constraints. Therefore, the total number of constraints for the proper timing of any retimed circuit G_r is $O(V^4)$. □

Thus, we obtain the following theorem.

Theorem 49 *In $O(V^8)$ time, Algorithm* PI *correctly computes a set Π of $O(V^8)$ pairs (ϕ_0, π) that includes the minimum-period clocking schemes that can be achieved by a retiming of G.*

Proof. The running time of the algorithm follows directly from Lemma 46. The correctness of the algorithm follows directly from Lemmas 47 and 48. □

2.10.5 Possible Periods for Retiming and Fixed-Duty-Ratio Tuning

In this section we describe the $O(V^2E)$-time Algorithm PIFDR that is employed by Algorithms POLYR&FDRT and POLYR&ST to compute the set $\Pi(\rho)$. This set has $O(V^3)$ elements which include the minimum clock period that can be achieved by retiming G and simultaneously tuning its clocking scheme with a fixed duty-ratio ρ. We show that when the duty-ratio is fixed, there are at most $O(V^3)$ constraints that determine the clock period of

PIFDR($G, \rho, \gamma_0, \gamma_1$)

1. Initialize $\Pi(\rho)$ to be the empty set.
2. Augment $\Pi(\rho)$ by the single lower bound on the clock period π that is defined in Lemma 27.
3. For each $u, v \in V$ compute
$$i_{\min}(u,v) = \min\left\{w(p) : u \overset{p}{\rightsquigarrow} v \text{ in } G\right\}.$$
4. For each $u \in V$, compute $DD(u,v,i)$ for all $v \in V$ and $i = 0, 1, \ldots, 3|V| - 3$, from the recurrence
$$DD(u,v,i) = d(v) + \max\left\{DD(u,x,i-w(e)) : x \xrightarrow{e} v \text{ and } i \geq w(e)\right\} .$$
5. Use Algorithm ODDCOUNT for each $u, v \in V$ and for $j = 0, 1$, to augment $\Pi(\rho)$ by the lower bound on the clock period π that is defined by the following constraint:
$$\pi \geq \max_i \frac{(\rho^{-j}+1)DD(u,v,i) + \gamma_0 + \gamma_1}{(\rho^{-j}+1)(1+i+l)/2+1} ,$$
where $-i_{\min}(u,v) \leq l \leq 3|V| - 1, i_{\min}(u,v) \leq i \leq 3|V| - 3 - l$, and $i + l$ is odd.
6. Use Algorithm EVENCOUNT for each $u, v \in V$ and for $j = 0, 1$, to augment $\Pi(\rho)$ by the lower bound on the clock period π that is defined by the following constraint:
$$\pi \geq \max_i \frac{DD(u,v,i) - \gamma_j}{(2+i+l)/2} ,$$
where $-i_{\min}(u,v) \leq l \leq 3|V| - 1, i_{\min}(u,v) \leq i \leq 3|V| - 3 - l$, and $i + l$ is even.
7. Return $\Pi(\rho)$.

Figure 2-20: Algorithm PIFDR takes a circuit G, a duty-ratio ρ, and two gap widths γ_0 and γ_1. The algorithm runs in $O(V^2E)$ time and computes a set $\Pi(\rho)$ of $O(V^3)$ pairs (ϕ_0, π) that describe clocking schemes with duty-ratio ρ. The set $\Pi(\rho)$ includes the minimum-period clocking scheme with duty-ratio ρ that can be achieved by retiming G.

any retiming of G. We then show how to compute the clock periods corresponding to these constraints in $O(V^3)$ time, and we argue that Algorithm PIFDR terminates in $O(V^2E)$ steps.

Algorithm PIFDR, shown in Figure 2-20, relies on the observation that in retiming and fixed-duty-ratio tuning the optimal clocking scheme is determined by the intersection on the (ϕ_0, π)-plane of Equation (2.1)

$$\begin{aligned}\pi &= \phi_0 + \gamma_0 + \phi_1 + \gamma_1 \\ &= (1+\rho)\phi_0 + \gamma_0 + \gamma_1\end{aligned}$$

with one of the lines defined by Inequalities (2.6), (2.7), and (2.8). Step 2 computes in $O(VE)$ time the single intersection of Equation (2.1) with the line defined by Inequality (2.8). For every pair of vertices $u, v \in G$, Step 3 computes the latch count $i_{\min}(u,v)$ of the path $u \overset{p}{\rightsquigarrow} v$ in G with the fewest latches. The maximum propagation delays $DD(u,v,i)$

are computed in Step 4 for every pair of vertices $u, v \in V$ and for $i = 0, 2, \ldots, 3|V| - 3$. Step 5 computes $O(V^3)$ intersections of Equation (2.1) with the lines defined by Inequality (2.6). For each of the $O(V^2)$ pairs $u, v \in V$, this step employs the $O(V)$-time Algorithm OddCount, shown in Figure 2-21, to compute $O(V)$ clock periods that are generated when every path from u to v has an odd number of latches after retiming. Algorithm OddCount is invoked twice in this step, once for each possible phase of u after retiming. Step 6 is similar to Step 5. In this step, Algorithm EvenCount is employed to compute $O(V^3)$ clock periods from Inequality (2.7) and for paths with an even number of latches after retiming. The operation of Algorithm EvenCount is almost identical to that of Algorithm OddCount, the only difference being that Algorithm EvenCount operates with respect to Inequality (2.7) instead of Inequality (2.6).

The correctness of Algorithm PiFDR is proved in the following lemma.

Lemma 50 *Let $G = \langle V, E, d, w, \chi \rangle$ be a two-phase, level-clocked circuit. Then Inequalities (2.6), (2.7), and (2.8) lead to $O(V^3)$ constraints that are significant for the proper timing of any retimed circuit G_r with a clocking scheme of fixed duty-ratio ρ.*

Proof. We shall prove the lemma for Inequality (2.6). The proof for Inequalities (2.7) and (2.8) is similar. Consider a retiming G_r of G and pair of vertices $u, v \in V$ with $\chi_r(u) = 0$ and $\chi_r(v) = 1$. Inequality (2.6) applies in this case. Following the same steps as in the proof of Lemma 48, we are left with the $O(V^2)$ constraints

$$\pi \geq \frac{DD(u,v,i) - \phi_0}{(1 + i + r(v) - r(u))/2}, \tag{2.38}$$

where $0 \leq i \leq 6|V| - 2$, $0 \leq i + r(v) - r(u) \leq 3|V| - 3$, and $i + r(v) - r(u)$ is odd. From Equation (2.1) and the definition of the duty-ratio ρ we have

$$\begin{aligned} \phi_0 &= \pi - \gamma_0 - \phi_1 - \gamma_1 \\ &= \pi - \gamma_0 - \rho\phi_0 - \gamma_1 \\ &= \frac{\pi - \gamma_0 - \gamma_1}{\rho + 1}. \end{aligned}$$

Thus, substitution of ϕ_0 in Inequality (2.38) yields

$$\pi \geq \frac{(\rho+1)DD(u,v,i) + \gamma_0 + \gamma_1}{(\rho+1)(1 + i + r(v) - r(u))/2 + 1},$$

where $-i_{\min}(u,v) \leq r(v) - r(u) \leq 3|V| - 1$, $i_{\min}(u,v) \leq i \leq 3|V| - 3 - (r(v) - r(u))$, and $i + r(v) - r(u)$ is odd. Therefore, for any fixed $l = r(v) - r(u)$ in the interval $-i_{\min}(u,v) \leq l \leq 3|V| - 1$, there is a single constraint

$$\pi \geq \max_i \frac{(\rho+1)DD(u,v,i) + \gamma_0 + \gamma_1}{(\rho+1)(1 + i + l)/2 + 1},$$

where i ranges over all integers such that $i_{\min}(u,v) \leq i \leq 3|V| - 3 - l$, and $i + l$ is odd. Therefore, there are $O(V)$ potential periods corresponding to the pair of vertices $u, v \in V$. Since there are $O(V^2)$ such pairs, the total number of constraints is $O(V^3)$. $\square$

ODDCOUNT$(G, u, v, j, DD, \gamma_0, \gamma_1, i_{\min}(u,v))$

1 Initialize S and S' to contain the line $\pi = 0$.

2 **for** $i = i_{\min}(u,v)$ **to** $3|V| - 3$

3   **do** $\pi_i(l) = \frac{(\rho^{-j}+1)DD(u,v,i)+\gamma_0+\gamma_1}{(\rho^{-j}+1)(1+i+l)/2+1}$

4 **if** $i_{\min}(u,v) \bmod 2 = 0$ $\triangleright$ computation for odd values of l

5   **then** $b \leftarrow i_{\min}(u,v)$

6   **else** $b \leftarrow i_{\min}(u,v) - 1$

7 **for** $i = 3|V| - 3$ **downto** $i_{\min}(u,v)$ **with** even i

8   **do if** $\pi_i(-b) \geq S_{top}(-b)$

9   **then** **while** $\pi_i(l) \cap S_{top}(l) \leq S_{top}(l) \cap S_{top-1}(l)$ **and** $S \neq$EMPTY

10   **do** $Pop(S)$

11         $Push(\pi_i, S)$

12 **if** $i_{\min}(u,v) \bmod 2 = 1$ $\triangleright$ computation for even values of l

13   **then** $b \leftarrow i_{\min}(u,v)$

14   **else** $b \leftarrow i_{\min}(u,v) - 1$

15 **for** $i = 3|V| - 3$ **downto** $i_{\min}(u,v)$ **with** odd i

16   **do if** $\pi_i(-b) \geq S'_{top}(-b)$

17   **then** **while** $\pi_i(l) \cap S'_{top}(l) \leq S'_{top}(l) \cap S'_{top-1}(l)$ **and** $S' \neq$EMPTY

18   **do** $Pop(S')$

19         $Push(\pi_i, S')$

20 **return** S and S'.

Figure 2-21: Algorithm ODDCOUNT takes a circuit G, a function DD, gap widths γ_0 and γ_1, a pair of vertices u and v, the phase of u after retiming, and the minimum initial latch count of any path from u to v in G. The algorithm runs in $O(V)$ time and computes the subset of $\Pi(\rho)$ that is generated by an odd number of latches on any path between u and v after retiming.

The $O(V)$-time Algorithm ODDCOUNT that is employed by Algorithm PIFDR to compute the clock periods due to odd latch counts after retiming is shown in Figure 2-21. This algorithm computes the boundary curves of the region on the (l, π) plane that is defined by the maximization in Step 5 of Algorithm PIFDR. The boundary curves are functions of the variable l, and they are returned in two stacks S and S'. Stack S contains the curves for odd values of l, and stack S' contains the curves for even values of l. We shall describe the computation of S in lines 4–11; the computation of S' in lines 12–19 is similar. The **for** statement of lines 7–11 scans the curves that correspond to even values i. The scan starts from the maximum value of i and proceeds towards its minimum value. In the beginning of the iteration for curve π_i, the boundary curves among $\pi_{3|V|-3}, \ldots, \pi_{i-1}$ are already on S. While π_i lies above the leftmost vertex of the region that is defined by the curves on S, then the top of S is popped. As soon as π_i creates a new vertex on the left of the current leftmost vertex, then π_i is pushed onto S. These conditions are checked by comparing the π-coordinates of intersections between π_i and elements of the stack. The π-coordinates are determined by the $\cap$ operator in the following way. When two curves intersect for l in the interval $-i_{\min}(u,v) \leq l \leq 3|V| - 3$, the operator $\cap$ returns the π-coordinate of the intersection. By convention, if two curves intersect for l outside the interval $-i_{\min}(u,v) \leq l \leq 3|V| - 3$, or if $S_{top-1}(l)$ in line 9 is empty, then the operator $\cap$ returns the values $-\infty$ and 0, respectively. During the computation of S, we can keep track of the values of l that correspond to vertices of the region defined by S. Thus, the maximum in Step 5 of Algorithm PiFDR can be found in $O(1)$ time, for every value of l, once S and S' are known.

The following lemma shows that Algorithm OddCount runs in linear time.

Lemma 51 *Algorithm* OddCount *terminates in $O(V)$ time.*

Proof. The time required to compute and compare intersection points is $O(1)$, as is the time required to push and pop S. Each curve π_i is pushed onto S or S' at most once. Curves popped from S or S' are discarded, so each curve π_i is popped from S or S' at most once. Thus, the total time needed to process each curve is $O(1)$, and since there are $O(V)$ curves, the total running time of Algorithm OddCount is $O(V)$. □

The following lemma is used to prove the correctness of Algorithm OddCount.

Lemma 52 *Consider the* **for** *loop in lines 7–11 of Algorithm* OddCount. *After each iteration of this loop with an even value i', the stack S contains all curves π_k such that for every odd l in the interval $-i_{\min}(u,v) \leq l \leq 3|V|-1$ and for some even k in the interval $i' \leq k \leq 3|V|-3$, we have*

$$\pi_k(l) = \max_i \pi_i(l) \ ,$$

where i ranges over all even values in the interval $i' \leq i \leq 3|V|-3$.

Proof. The proof is by induction on the number of iterations. The base case for $i' = 2\lfloor(3|V|-3)/2\rfloor$ holds, since $\pi_i(l) > 0$ for all i, l in their corresponding intervals.

For the inductive step, we assume that the lemma holds for all values $i > i'$, and we prove that it also holds for i'. Consider the curve $\pi_{i'}(l)$, and let $\pi_t(l)$ be the curve on the top of S.

If $\pi_{i'}(-b) < \pi_t(-b)$, where $b = 2\lfloor i_{\min}(u,v)/2\rfloor$, then the algorithm does not push $\pi_{i'}$ onto S. We will show that in this case we have $\pi_{i'}(l) < \pi_t(l)$ for all $l \geq -b$, and therefore the lemma holds. Since $t > i'$, the definitions of $\pi_{i'}(l)$ and $\pi_t(l)$ and the inequality $\pi_{i'}(-b) < \pi_t(-b)$ imply, after some algebraic manipulation, that

$$-b > \frac{(\rho^{-j}+1)\,(DD(u,v,t)i/2 - DD(u,v,i)t/2) + (\gamma_0+\gamma_1)(i/2-t/2)}{(\rho^{-j}+1)^2\,(DD(u,v,i) - DD(u,v,t))\,/2} - \frac{2}{(\rho^{-j}+1)} - 1 \ . \tag{2.39}$$

The curves $\pi_{i'}(l)$ and $\pi_t(l)$, however, have a single intersection $(l_{i',t}, \pi_{i',t})$, where $l_{i',t}$ equals the right-hand side of Inequality (2.39), and thus, we have $-b > l_{i',t}$. Since $\pi_{i'}(l)$ and $\pi_t(l)$ are continuous for $l \geq -b$, we conclude from the inequality $\pi_{i'}(-b) < \pi_t(-b)$ that $\pi_{i'}(l) < \pi_t(l)$ for all $l \geq -b$.

Now, let us consider the situation where $\pi_{i'}(-b) \geq \pi_t(-b)$. While the condition of the **while** statement in line 9 of the algorithm holds, the stack S is popped. We distinguish the following two cases:

$l_{i',t} < -b$ **or** $l_{i',t} \geq 3|V|-1$. In this case, we have $\pi_{i'}(l) \geq \pi_t(l)$ for all l in the interval $-b \leq l \leq 3|V|-1$, by the continuity of $\pi_{i'}(l)$ and $\pi_t(l)$ (see Figure 2-22(a)). Therefore, $\pi_t(l)$ is not a boundary curve when $\pi_{i'}(l)$ is also considered, and it can be discarded without violating the lemma.

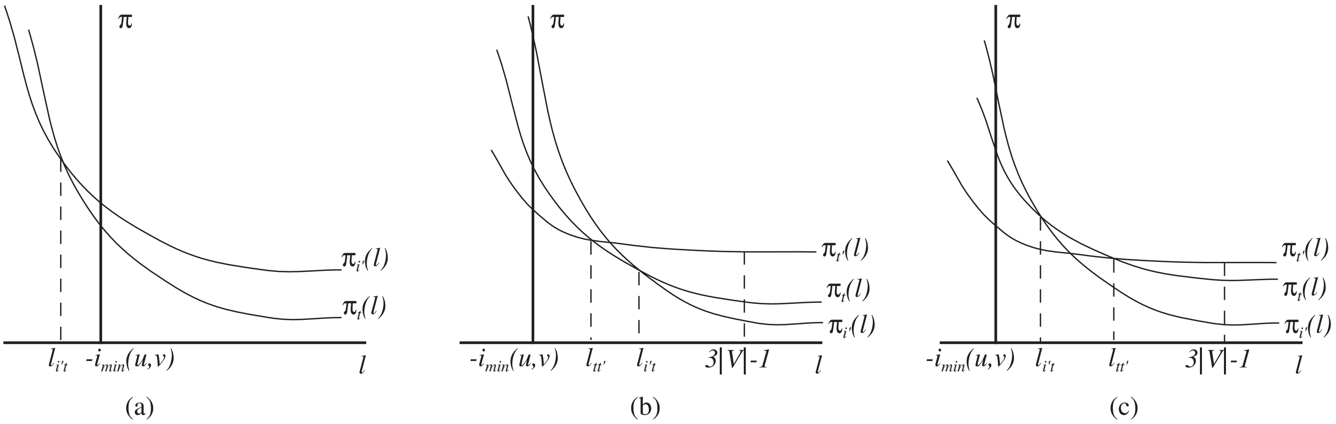


Figure 2-22: Parts (a) and (b) illustrate the situations in which the curve $\pi_{i'}(l)$ lies above the curve $\pi_t(l)$ on the top of the stack S, resulting into popping $\pi_t(l)$ off of S. In part (a), the curves $\pi_{i'}(l)$ and $\pi_t(l)$ intersect outside the interval $[-b, 3|V|-3]$, whereas in Part (b) they intersect inside the interval. Part (c) illustrates the situation in which the curve $\pi_{i'}(l)$ is pushed onto the stack S.

$-b \leq l_{i',t} \leq 3|V|-1$. In this case we have $\pi_{i'}(l) \geq \pi_t(l)$ for all l in the interval $-b \leq l \leq l_{i',t}$. Since $\pi_t(l) \geq \pi_{t'}(l)$ for $-b \leq l \leq l_{t,t'}$, where $l_{t,t'}$ is the l-coordinate of the intersection between $\pi_t(l)$ and the curve $\pi_{t'}(l)$ below the top of the stack S (see Figure 2-22(b)), by the monotonicity and continuity of $\pi_t(l)$ and $\pi_{t'}(l)$ we have $l_{t,t'} \leq l_{i',t}$. Therefore, $\pi_t(l)$ is not a boundary curve when $\pi_{i'}(l)$ is taken into consideration, and it can be discarded without violating the lemma.

The curve $\pi_{i'}(l)$ is pushed onto the stack S the first time the condition of the **while** statement is not satisfied. In this case we have $\pi_{i'}(l) \geq \pi_t(l)$ for all $-b \leq l \leq l_{i',t}$, and therefore the lemma holds (see Figure 2-22(c)). □

Corollary 53 *When Algorithm* ODDCOUNT *terminates, the stack S contains all curves π_k such that for every odd l in the interval $-i_{\min}(u,v) \leq l \leq 3|V|-1$ and for some even k in the interval $i_{\min}(u,v) \leq k \leq 3|V|-3$, we have*

$$\pi_k(l) = \max_i \pi_i(l) \ ,$$

where i ranges over all even values in the interval $i_{\min}(u,v) \leq i \leq 3|V|-3$. Similarly, the stack S' contains all curves π_k such that for every even l in the interval $-i_{\min}(u,v) \leq l \leq 3|V|-1$ and for some odd k in the interval $i_{\min}(u,v) \leq k \leq 3|V|-3$, we have

$$\pi_k(l) = \max_i \pi_i(l) \ ,$$

where i ranges over all odd values in the interval $i_{\min}(u,v) \leq i \leq 3|V|-3$.

Proof. The proof for S follows directly from Lemma 52 for $i' = i_{\min}(u,v)$. The proof for S' is similar. □

Thus, we obtain the following theorem.

Theorem 54 *In $O(V^2E)$ time, Algorithm* PIFDR *correctly computes a set $\Pi(\rho)$ of $O(V^3)$ real numbers π that includes the period of the optimal clocking scheme with duty-ratio ρ that can be achieved by a retiming of G.*

Proof. The initialization are performed in $O(1)$ time. Step 2 performs a tramp-steamer computation and terminates in $O(VE)$ time. Step 4 can be performed in $O(V^2E)$ time. According to Lemma 51, Step 5 terminates in $O(V^3)$ time, since there are $O(V^2)$ pairs $u, v \in V$. Therefore, the total running time of Algorithm PIFDR is $O(V^2E)$ time.

The correctness of the algorithm follows directly from Lemma 50 and Corollary 53. □

2.11 Multiphase clocking

Many of the results for two-phase clocking can be generalized to multiphase clocking disciplines. In this section, we sketch the formal framework for multiphase clocking and how the various two-phase algorithms for timing verification and optimization can be generalized. We derive a verification algorithm for k-phase clocking schemes that runs in $O(kVE)$ time on a "simple" k-phase circuit with $|V|$ combinational elements and $|E|$ interconnections. Clock tuning for simple k-phase circuits can be performed, but our best algorithms to date require general linear programming. The algorithms for retiming to achieve a given clocking scheme can be generalized to run in $O(VE + V^2 \lg V)$ time when the given clocking scheme is symmetric, and in $O(kV^3)$ time when the given clocking scheme is not symmetric. The retiming and symmetric tuning problem can be approximately solved in $O((VE + V^2 \lg V)\lg(kV/\epsilon))$ time, where ϵ is the relative error of the approximation. The retiming and fixed-duty-ratio tuning problem can be approximately solved in $O(kV^3 \lg(kV/\epsilon))$ time. The approximation scheme for the simultaneous clock tuning and retiming of two-phase circuits can also be generalized, but the resulting polynomial running time is impractically large. Some of the results in this section are similar to results obtained independently by Lockyear and Ebeling [32].

We define a *k-phase clocking scheme* to be a $2k$-tuple $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1, \ldots, \phi_{k-1}, \gamma_{k-1} \rangle$ of real numbers that satisfies the following constraints:

CS1. $\pi = \sum_{i=0}^{k-1}(\phi_i + \gamma_i)$;
CS2. $0 < \phi_i < \pi$ for $i = 0, 1, \ldots, k-1$;
CS3. $\phi_i + \gamma_i > 0$ for $i = 0, 1, \ldots, k-1$;
CS4. $\gamma_i + \phi_{i+1} > 0$ for $i = 0, 1, \ldots, k-1$;
CS5. $\phi_i + \gamma_i + \phi_{i+1} < \pi$, for $i = 0, 1, \ldots, k-1$;

where we assume here and henceforth that addition in subscripts is performed modulo k. In this formulation, we allow overlapping phases, that is, the γ_i "gaps" can be negative. Condition CS3 says that phases rise in order from 0 to $k-1$, and Condition CS4 says that they fall in order. For the simple k-phase circuits we consider, Conditions CS3 and CS4 are not both strictly necessary. One can show that if phases rise in one order, there is no advantage to having them fall in another order. Condition CS5 guarantees that at least one of the k phases is down at any point in time; for two-phase circuitry, this constraint reduces to $\gamma_i > 0$ for $i = 0, 1$. We say a simple k-phase clocking scheme is *symmetric* if $\phi_i = \phi$ and $\gamma_i = \gamma$ for all some constants ϕ and γ and all $i = 0, 1, \ldots, k-1$.

A *simple k-phase circuit* has the following properties:

MC1. It employs a simple k-phase clocking scheme.

MC2. It contains no latch-free cycles.

MC3. Any purely combinational path that begins with a latch controlled by some ϕ_i ends with a latch controlled by ϕ_{i+1}.

We denote such a simple k-phase circuit by $G = \langle V, E, d, w, \chi \rangle$, where χ now maps each vertex to a number in $\{0, 1, \ldots, k-1\}$ corresponding to the input phase of the vertex. Condition MC3 ensures that the latches on any path are clocked in order by the phases in the clocking scheme.

We now summarize the generalizations of our algorithms for two-phase circuits to simple k-phase circuits.

Clock period constraints. Inequalities (2.2) and (2.3), which provide necessary and sufficient conditions for the proper timing of two-phase circuits, generalize straightforwardly to multiphase circuits. Let $G = \langle V, E, d, w, \chi \rangle$ be a simple k-phase circuit that employs a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1, \ldots, \phi_{k-1}, \gamma_{k-1} \rangle$. Then G is properly timed if and only if, for every path $u \overset{p}{\leadsto} v$ in G,

$$d(p) \le \pi \left\lfloor \frac{w(p)+1}{k} \right\rfloor + \psi(\chi(u), \chi(v)) \ , \tag{2.40}$$

where $\psi(i, j) = \phi_i + \gamma_i + \phi_{i+1} + \cdots + \gamma_j + \phi_{j+1}$.

Timing verification. To check whether a simple multiphase circuit is properly timed by a simple k-phase clocking scheme, we can use the constraints defined by Inequality (2.40) to derive a set of cyclic constraints identical to those described by Inequality (2.8), except with k in the denominator instead of 2. These cyclic constraints can be checked in $O(VE)$ time as in Step 4 of Algorithm VERIFY. If the cyclic constraints are met, the path constraints (2.40) need only be satisfied by simple paths. The $D(v, i)$ values in Algorithm VERIFY must now be computed for $i = 0, 1, \ldots, (2k-1)|V|$, and the entire verification algorithm runs in $O(kVE)$ time.

Sensitivity analysis. The sensitivity analysis algorithms that we presented for two-phase circuitry can be extended for k-phase clocking schemes. For a single gate, noncritical sensitivity analysis can be performed in $O(kVE)$ time, since the $D(v, i)$ and $D'(v, i)$ values must now be computed for $i = 0, 1, \ldots, (2k-1)|V|$. For all gates, noncritical sensitivity analysis can still be performed in $O(VE + V^2 \lg V)$ time with appropriate choice of the edge-lengths for the all-pairs shortest-paths algorithm. Critical sensitivity analysis can be performed in $O(kVE)$ time for a single gate.

Clock tuning. Clock tuning for simple multiphase circuits can be performed in much the same way as for two-phase circuits, but the problem no longer succumbs to a simple, two-dimensional linear program. Linear programming still suffices to solve the problem, however. Each ϕ_i becomes a variable in a linear program, which can be solved with standard techniques [41]. If the number of phases is assumed to be fixed, then the problem can be solved in $O(kV^2)$ steps, that is, in a number of steps proportional to the number of constraints for proper timing [36]. Certain special cases can be handled without resorting to general linear programming. For example, a circuit with a three-phase, nonoverlapping clock can be tuned in $O(VE)$ time using the three-dimensional linear programming algorithm of Megiddo [37].

Retiming with symmetric clocking schemes. When clock phases are symmetric, simple k-phase circuits can be retimed to achieve a given clock period in $O(VE + V^2 \lg V)$ time, independent of k. The constraints that must be solved are a natural analog of the two-phase constraints described by Inequalities (2.21), (2.22), (2.23), and (2.24):

$$\begin{array}{rcll} r(u) - r(v) & \leq & w(e) & \text{for all } u \xrightarrow{e} v \in E \\ R(v) - r(v) & \leq & 0 & \text{for all } v \in V \\ R(u) - R(v) & \leq & w(e) - \frac{k}{\pi} d(v) & \text{for all } u \xrightarrow{e} v \in E \\ r(v) - R(v) & \leq & \frac{k}{\pi}(\phi - d(v)) + 1 & \text{for all } v \in V \ . \end{array}$$

As with the two-phase constraints, these constraints can be solved using the algorithm MILP from [30].

Retiming (with arbitrary clocking schemes). Even when clock phases are not symmetric, we can retime to achieve a given simple k-phase clocking scheme in $O(kV^3)$ time. For simple multiphase circuits, inequalities analogous to Inequalities (2.29) and (2.30) can be formulated as simple summations. In particular, Inequality (2.40) can be rewritten as

$$\sum_{i=\chi(v)}^{\chi(v)+r(v)} (\gamma_i + \phi_{i+1}) - \left(\gamma_{\chi(v)} + \phi_{\chi(v)+1}\right) + \pi \left\lfloor \frac{w(p)+1}{k} \right\rfloor + \psi(\chi(u), \chi(v)) - d(p)$$

$$\geq \sum_{i=\chi(u)-1}^{\chi(u)-1+r(u)} (\phi_i + \gamma_i) - \left(\phi_{\chi(u)-1} + \gamma_{\chi(u)-1}\right) \qquad (2.41)$$

for any retimed circuit. These constraints can be solved using integer monotonic programming. By maintaining the two summations dynamically, the claimed running time of $O(kV^3)$ can be obtained, where the additional factor of k stems from the fact that some $r(v)$ can now be as high as $O(kV)$.

Retiming for minimum latch count. For k-phase, symmetric clocking schemes, we can prove that this problem amounts to computing an assignment $r : V \to \mathbf{Z}$ such that

$$\sum_{v \in V} (\text{indegree}(v) - \text{outdegree}(v))\, r(v)$$

is minimized, subject to

$$r(u) - r(v) \leq w(e)$$

for every edge $u \xrightarrow{e} v \in E$, and

$$r(u) - r(v) \leq \left\lfloor w(p) - \frac{k}{\pi}\,(d(p) - \phi) + 1 \right\rfloor$$

for every path $u \overset{p}{\rightsquigarrow} v$ in G. This problem is still the dual of an uncapacitated minimum-cost flow problem, and it can be solved in $O(V^3 \lg V)$ time.

Retiming for minimum clock period. The algorithms for retiming a simple k-phase circuit to achieve a given clocking scheme can be used to obtain fully polynomial-time approximation schemes for several problems related to retiming for minimum clock period. Specifically, using binary search in the clock period domain as described in Section 2.9, we can solve the retiming and symmetric tuning problem in $O((VE + V^2 \lg V)\lg(kV/\epsilon))$ time with relative error ϵ. Similarly, we can solve the retiming and fixed-duty-ratio tuning problem in $O(kV^3 \lg(kV/\epsilon))$ time. A straightforward generalization of the linear search scheme in Section 2.9 yields a fully polynomial-time approximation scheme for the general retiming and tuning problem for simple k-phase circuits. The running time of this algorithm contains a factor ϵ^{-k}, however, which can be prohibitively large for a small relative error ϵ.

2.12 Conclusion

Our algorithms for verifying and optimizing level-clocked circuits have their limitations. For example, as is the case with most work on retiming, it is hard to incorporate data-dependent propagation delays in the framework without making most of the interesting questions NP-hard. Several issues, however, are amenable to efficient algorithmic solutions. We address some of these issues in this section. We first discuss how to cope in our ideal model with nonideal clocking waveforms. We then show how to adjust our approximation algorithms in order to compute exact solutions to the various retiming and tuning problems. We move on to describe the incorporation in our model of precharged gates, nonuniform propagation delays, and nonzero minimum propagation delays. We conclude by discussing generalizations of our algorithms to handle gated clocks and nonsimple multiphase clocking disciplines.

A phenomenon that arises in practice is that clock phases are never truly square waves; the waveform rises or falls over an interval of time. To cope with this effect in our ideal model, one can ensure that the gaps between clock phases are sufficiently large that consecutive latches clocked on opposite phases are not high simultaneously. Similarly, the effect of clock skew can be handled by adjusting the gap widths.

The fully polynomial-time approximation schemes in Section 2.9 can be adjusted to find an exact solution when the propagation delays of the combinational elements are integers. Specifically, the retiming and symmetric tuning problem can be solved exactly in $O((VE + V^2 \lg V)\lg(Vd_{\max}/\delta))$ time, where

$$\delta = \min\left\{\frac{1}{3\,|V| - 1}, \frac{\gamma_0}{2(3\,|V| - 1)^2}, \frac{\gamma_1}{2(3\,|V| - 1)^2}\right\} .$$

Similarly, the retiming and fixed-duty-ratio problem can be solved exactly in $O(V^3 \lg(Vd_{\max}/\delta))$

time.

An important extension of our work is the incorporation of precharged gates, nonuniform propagation delays, and nonzero minimum propagation delays to our circuit model. With precharged gates, retiming and clock tuning still have efficient algorithmic solutions, but there are many subtleties that arise in the formulation of the constraints. The incorporation of functional elements where the propagation delay may differ for different input-output pairs (the "nonuniform propagation delay" model from [31]) changes the time complexities of our algorithms, but the essential algorithms remain unchanged. When minimum propagation delays (sometimes called "contamination" delays) are incorporated in the model, the output of a functional element does not become invalid until some specified minimum amount of time after an input changes. A polynomial-time algorithm for the timing verification problem when minimum propagation delays are included in the circuit model has appeared in [56]. A polynomial-time algorithm for the clock tuning problem with minimum propagation delays has appeared in [53]. We believe that many of our optimization algorithms can be generalized to handle such circuits in polynomial time. This is a topic of current research.

Two generalizations of our work which seem more problematic are the handling of gated clocks and nonsimple multiphase clocking disciplines. When logic circuits involve the clock signals themselves (so-called gated clocks), it is possible to show, in the general case, that many timing verification and optimization problems are NP-hard. Nevertheless, we suspect that by making conservative assumptions regarding the behavior of such circuits, many of these problems become tractable. Nonsimple multiphase circuits also exhibit many subtleties. A path in such circuits may pass through latches in an arbitrary order, rather than the canonical order assumed in a simple multiphase circuit. Though the proper timing of such circuits can be verified using the analysis and algorithms from [20], the timing optimization of such circuits is possibly more complex. Whether these problems have efficient solutions is a topic for further research.

Chapter 3

TIM: A Timing Package for Level-Clocked Circuitry

3.1 Introduction

In this chapter we describe TIM, a new timing package we have developed for circuitry that employs a nonoverlapping two-phase clocking scheme. TIM's features include optimizations for retiming and sensitivity analysis as well as more conventional operations such as verification of proper timing and optimal tuning of clocking schemes. The entire package has been written using the C programming language, and it has been integrated into the SIS tools from Berkeley. Copies of the software have been available over the Internet since June 1993 and can be obtained by sending a request to `marios@lcs.mit.edu`.

Several tools have been developed for analyzing the timing of circuitry that contains level-clocked latches [1, 4, 6, 23, 40, 51, 54]. 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. Moreover, the algorithms in TIM are based on the algorithms described in Chapter 2, and thus they are provably correct and run in polynomial time.

TIM has been applied on a variety of circuits that have been obtained from academic and industrial sources. The results from the application of our tool are presented in Chapter 4.

This chapter is organized as follows. Section 3.2 gives an overview of the system, and Section 3.3 presents TIM's circuit model which extends the ideal model that we assumed in Chapter 2. Section 3.4 describes the algorithms that we implemented in TIM and the new constraints for the extended model. Section 3.5 concludes this chapter by discussing the performance of our implementation.

Parts of this chapter represent joint research with Keith Randall.

3.2 System Overview

TIM provides the following classes of operations for circuitry that employs nonoverlapping two-phase clocking schemes:

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

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

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.

3.3 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 $\pi = \phi_0 + \gamma_0 + \phi_1 + \gamma_1$. 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 Chapter 2. A two-phase, level-clocked circuit is modeled as a vertex-weighted, edge-weighted graph $G = \langle V, E, d, w, \chi \rangle$. 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. 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.

In TIM we have extended our ideal model from Chapter 2 to include non-ideal latches: All latches in TIM are assumed to have equal propagation delays, equal setup times and equal hold times. (The equality restriction is not necessary for the algorithms that perform timing verification, clock tuning and sensitivity analysis. The proofs for the correctness and the running time of the retiming algorithms, however, rely on the assumption that all latches have identical delay characterisitics.) Each latch has two delay parameters associated with it. The first parameter, which we denote by d_{out}, gives the time between the rising edge of the phase that clocks the latch and the moment that data appear at the latch output. The second parameter, which we denote by d_{thru}, gives the propagation delay of data through the latch when the clocking phase is already high at the time that data arrive at the latch input. Setup and hold times are embedded in the gaps γ_0 and γ_1 of the clocking scheme π. Given an intended clocking scheme $\pi' = \langle \phi'_0, \gamma'_0, \phi'_1, \gamma'_1 \rangle$, our algorithms operate on the ideal model assuming a clocking scheme $\pi = \langle \phi'_0 - S, \gamma'_0 + S, \phi'_1 - S, \gamma'_1 + S \rangle$, where S is the setup time of a latch. In order to avoid data contamination due to zero minimum propagation delays, the gaps γ'_0 and γ'_1 are required to exceed the hold time H of a latch.

Another phenomenon that arises in practice is that clock phases are never truly square waves; the waveform rises or falls over an interval of time. To cope with this effect in TIM, one can ensure that the gaps between clock phases are sufficiently large that consecutive latches clocked on opposite phases are not high simultaneously. Similarly, the effect of clock skew can be handled by adjusting the gap widths.

3.4 System Operation

In this section we describe the algorithms implemented in TIM. We first describe the shortest-paths algorithms that we implemented. We then describe the operations of our tool. For each of these operations, the user may use a technology library to determine the propagation delays of the gates, or he can specify all gates to have unit propagation delays.

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 [5]. 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 [5]. 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 [11] 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.

3.4.1 Verification of Proper Timing

The timing verification operation in TIM is a direct implementation of Algorithm TV that is described in Section 2.3 and runs in $O(VE)$ time. In our implementation, we have adapted Inequalities (2.11), (2.12), and (2.8) to take into account the propagation delays of the latches. Specifically, for $v \in V$ and $i = 0, 1, \ldots, 3|V| - 3$, we check the constraints

$$d_{out} + i \cdot d_{thru} + D(v,i) \leq \pi\left(\frac{1+i}{2}\right) + \phi_{1-\chi(v)}$$

if i is odd, and

$$d_{out} + i \cdot d_{thru} + D(v,i) \leq \pi\left(\frac{2+i}{2}\right) - \gamma_{1-\chi(v)}$$

if i is even; and for every simple cycle c

$$d(c) + w(c) \cdot d_{thru} \leq \pi\left(\frac{w(c)}{2}\right) \ .$$

3.4.2 Sensitivity Analysis

The current version of TIM performs both noncritical and critical sensitivity analysis.

The noncritical sensitivity analysis is performed for all gates in the circuit, and it is a direct implementation of Algorithm ALLNCSA. The asymptotic running time of our implementation, however, is $O(VE \lg V)$ time, because we employ a binary heap as a priority queue.

The critical sensitivity analysis in TIM is not a direct implementation of Algorithm CSA. Given a circuit G and a feasible clocking scheme π, TIM runs Algorithm ALLNCSA to identify the critical gates in the circuit, that is, the gates with zero slack. Then, for each critical gate u, TIM sets $d(u) = 0$ and runs Algorithm NCSA(G, π', u), where the clocking scheme π' is computed by tuning the clock of G when $d(u) = 0$. The user can restrict the clocking scheme π' to have the same duty-ratio as the clocking scheme π. Alternatively, he may choose to use a π' that is computed by clock tuning over all possible duty-ratios. Our implementation of Algorithm NCSA is based on the $O(VE)$-time shortest-paths algorithm by Bellman and Ford. Thus, the critical sensitivity analysis operation in TIM runs in a total of $O(VE \lg V + cVE)$ steps, where c denotes the number of critical gates for the clocking scheme π.

3.4.3 Clock Tuning

TIM can perform clock tuning with either fixed or unconstrained duty-ratio. Our implementation is based directly on the algorithm described in Section 2.5 and terminates in $O(VE)$ time. The clock tuning algorithm in TIM computes the $O(V^2)$ values $D(v,i)$ and then solves a linear program in two dimensions in order to determine the optimal point of the ensuing constraints.

3.4.4 Retiming for Speed

Included among TIM's features are algorithms for the retiming problem and fully-polynomial approximation algorithms for the various retiming and tuning problems. The basic subroutine of all these algorithms is Algorithm RETIME which operates on the constraints (2.26), (2.27) and (2.28). In our implementation, we have adapted these constraints to account for the propagation delays of the latches. Specifically, TIM computes a retiming such that for every edge $u \xrightarrow{e} v \in E$, we have

$$r(u) - r(v) \leq w(e) \ ,$$

and for every path $u \overset{p}{\rightsquigarrow} v$, we have

$$\begin{aligned} d_{out} + (w(p) + r(v) - r(u)) \cdot d_{thru} + d(p) \quad \leq \quad & \pi\left(\frac{1 + w(p)}{2}\right) + \phi_{\chi(u)} \\ & + \pi\left\lfloor\frac{r(v)}{2}\right\rfloor + (r(v) \bmod 2)\left(\gamma_{\chi(u)} + \phi_{1-\chi(u)}\right) \\ & - \pi\left\lfloor\frac{r(u)}{2}\right\rfloor - (r(u) \bmod 2)\left(\phi_{\chi(u)} + \gamma_{\chi(u)}\right) \ , \end{aligned}$$

if $\chi(u) \neq \chi(v)$, and

$$\begin{aligned} d_{out} + (w(p) + r(v) - r(u)) \cdot d_{thru} + d(p) \quad \leq \quad & \pi\left(\frac{2 + w(p)}{2}\right) - \gamma_{1-\chi(u)} \\ & + \pi\left\lfloor\frac{r(v)}{2}\right\rfloor + (r(v) \bmod 2)\left(\gamma_{1-\chi(u)} + \phi_{\chi(u)}\right) \\ & - \pi\left\lfloor\frac{r(u)}{2}\right\rfloor - (r(u) \bmod 2)\left(\phi_{\chi(u)} + \gamma_{\chi(u)}\right) \ , \end{aligned}$$

if $\chi(u) = \chi(v)$. It is straightforward to verify that these constraints can be brought into the form $f(r(v)) \geq g(r(u))$, where f and g are monotonic functions. Using a binary heap as a priority queue, TIM generates the retiming constraints in $O(VE \lg V)$ time by an all-pairs shortest-paths computation. Therefore, our implementation of Algorithm RETIME terminates in $O(V^3 + VE \lg V)$ steps.

Retiming in TIM does not relocate I/O latches. This constraint can be easily enforced by introducing the pair of inequalities

$$\begin{aligned} r(u) - r(v) &\geq 0 \\ r(u) - r(v) &\leq 0 \end{aligned}$$

for every pair of vertices $u, v \in V$ such that $u \xrightarrow{e} v$ and there exists an I/O latch on the wire e.

3.4.5 Retiming for Minimum Latch Count

TIM's retiming algorithm for minimum latch count is an implementation of Orlin's capacity scaling algorithm for computing minimum-cost flows [38], and it terminates in $O(V^3)$ steps.

Our implementation works on a modified circuit graph that takes into account latch sharing. The construction of this graph is identical to the construction presented in [31] for retiming of edge-triggered circuitry. TIM takes into account the propagation delays of the latches by finding a solution to the constraint

$$r(u) - r(v) \leq w(e)$$

for every edge $u \xrightarrow{e} v$ in G, and

$$d_{out} + (w(p) + r(v) - r(u)) \cdot d_{thru} + d(p) \leq \pi(w(p) + r(v) - r(u))/2 + \pi - \gamma$$

for every path $u \overset{p}{\leadsto} v$ in G. It is straightforward to verify that these constraints can be rewritten in a form that is the dual of an uncapacitated minimum-cost flow problem. As with retiming for speed, retiming for minimum latch-count does not relocate I/O latches.

3.5 System Performance

Our primary concern during TIM's implementation was to achieve correct functionality. Speed, although important, was a secondary concern. Nevertheless, TIM operates reasonably fast on reasonably large inputs. We have used extensively the current version of TIM on a Sun SPARCstation 2 with 64MB of main memory. For circuits with 1,500 gates after technology mapping, the timing analysis and clock tuning operations typically require a couple of minutes. The retiming operations, however, require approximately 35 minutes. For large inputs, our tool is almost always slowed down because of paging. The adverse effect of paging on the running time of our algorithms is particularly evident when retiming, due to the $O(V^2)$ space requirements of the retiming algorithms.

There are several straightforward ways to speed up TIM's retiming operations. The practical efficiency of the latch count minimization algorithm will possibly improve by using a simplex-based algorithm to solve the minimum-cost flow problem. Even though simplex-based algorithms for minimum-cost flow are not guaranteed to run in polynomial time, they have been shown to perform particularly well in practice [15].

Another operation in TIM with potential for easy improvement is retiming with symmetric clocking schemes. The current implementation of TIM employs the general retiming algorithm RETIME even when the clocking schemes are symmetric. We believe that the practical efficiency of this operation can be substantially improved by implementing Algorithm RWSCS that runs on a sparse graph representation of the problem.

Chapter 4

Edge-Triggering vs. Level-Clocking

4.1 Introduction

Level-clocking is becoming an increasingly popular alternative to edge-triggering as a clocking methodology for high-performance designs. 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 computations are allowed to extend beyond a single clock period during the transparent phase of the level-clocked latches, in constrast to edge-triggered circuitry, in which computations along every path must complete within a clock period. Advocates of edge-triggering, on the other hand, present simplicity and implementation ease as major advantages of edge-triggering, since edge-triggered latches 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 latches.

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 used TIM to 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 urge the reader not to interpret our results in a narrow quantitative manner, however, since our tool may have introduced round-off errors. The reader should rather focus on the qualitative conclusions that can be drawn from our comparison.

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

Parts of this chapter represent joint research with Keith Randall.

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 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.

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 hope to also obtain interesting level-clocked designs and remedy this situation before the full paper is published.)

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 latch by a pair of back-to-back level-clocked latches that are clocked by a two-phase, nonoverlapping clocking scheme, as shown in Figure 4-1. (In fact, it is common in VLSI to implement edge-triggered latches by a pair of back-to-back level-clocked latches [14, 58].) The two-phase circuit that results after this conversion has the same clock period and the same number of storage elements as the original edge-triggered circuit, assuming that each edge-triggered latch counts as two level-clocked 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

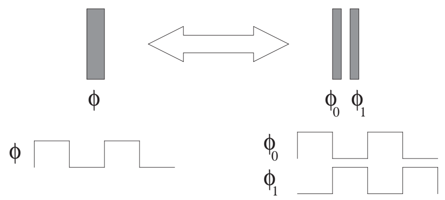


Figure 4-1: Replacement of an edge-triggered latch by a pair of level-clocked latches. The edge-triggered latch is clocked by a single clock ϕ, and the two level-clocked latches are clocked on the phases ϕ_0, ϕ_1 of a two-phase, nonoverlapping clocking scheme.

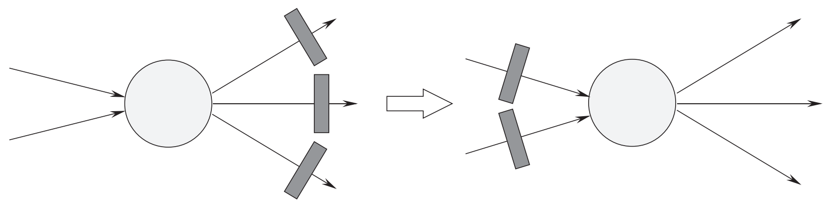


Figure 4-2: This figure illustrates retiming of a gate with lag 1. In this case, one storage element is removed from each output wire of the gate, and one storage element is inserted at each input wire of the gate. The total number of storage elements is reduced by 1. The critical paths in the circuit may also change as a result of the relocation of the storage elements.

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.

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 [28, 29, 31]. 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 [29]. Figure 4-2 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.

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 edge-triggered latches. 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 latch counts as two level-clocked latches.

Timing verification and optimization of synchronous circuitry has been the subject of extensive study [3, 6, 17, 20, 21, 23, 28, 31, 32, 34, 51, 52, 54, 56, 57]. The concept of replacing each edge-triggered latch by a pair of back-to-back level-clocked latches, and then using retiming for speed optimization has been mentioned in [3, 21, 32]. The potential of level-clocking for reducing the number of storage elements has been mentioned in [32]. The idea of using latches instead of edge-triggered latches has been also used in [55]. Retiming for speed has been studied in the context of single-phase level-clocked circuits in [52]. 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.

The remainder of this chapter has three sections. Section 4.2 describes our experimental methodology and reports our results on the relative speed of the two implementation approaches. In Section 4.3 we present our experimental results on latch-minimization. Section 4.4 concludes with a discussion of our results and directions for further research. Appendix A.2 presents an upper bound on the relative speedup that can be achieved by two-phase clocking over edge-triggering and explains intuitively how two circuit characteristics, the maximum gate delay $d_{\max}$ and the maximum delay-to-register ratio R of the circuit, determine the speedups that can be achieved by two-phase clocking.

4.2 Speedup Experiments

In this section, 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. Our test circuits were MCNC benchmark circuits and AT&T communication circuits, all of which were originally designed with edge-triggered latches. The largest among these circuits had 290 gates.

4.2.1 Experimental Methodology

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 original number of latches 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. 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 original delay assigned to v. For each gate in a circuit G, this original delay was the worst-case propagation delay between an input pin and an output pin of the gate, based on the technology library that was used with the circuit. 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.

4.2.2 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 maximum delay-to-register ratio R, which is defined as the maximum ratio of total delay over total latch count around the cycles in the edge-triggered circuit.

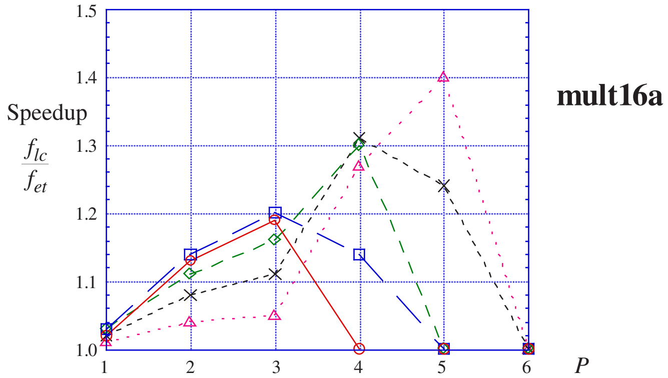
Our experimental results for SP3 are illustrated in the plots of Figure 4-3. Each plot gives data for an original test circuit $G_{1.0}$ and its four delay-modified versions G_i for $i = 0.8, 0.6, 0.4, 0.2$. For each of the five delay configurations of a test circuit and for degrees of pipelining up 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 maximum delay-to-register 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 maximum delay-to-register ratio R is a lower bound on the clock period of both the edge-triggered and the level-clocked circuit [21, 43]. Consequently, the longest combinational delay in the circuits is at least R under any transformation that does not change the number of latches around the cycles in the circuit. Retiming distributes the latches, 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.

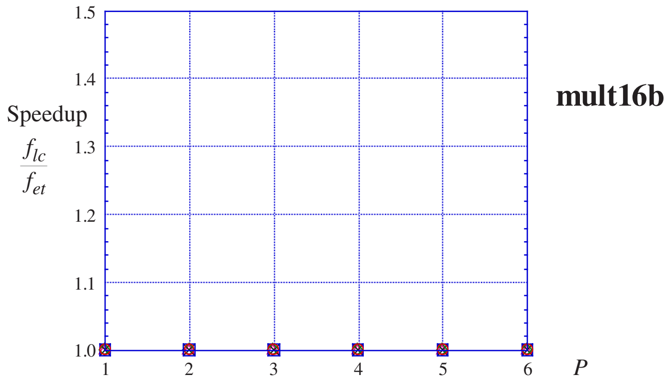
Let us examine more closely some characteristic graphs in Figure 4-3. Our initially surprising results from experiments SP1 and SP2 can be explained by looking at the ratios $d_{\max}/R$ of the original circuits, which correspond to $P = 1$ and $i = 1.0$. For every such circuit, the ratio $d_{\max}/R$ is smaller than 0.67. The only exceptions are `mult16b`, `ampseq2`, and `ampseq1`. `mult16b` has a ratio greater than 1, and consequently, 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 4-3 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 4-3 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.

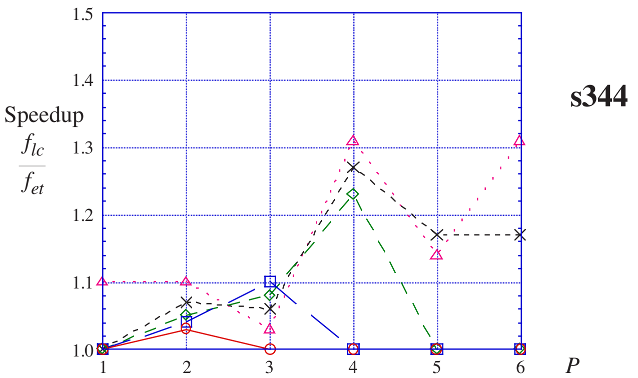


P	1	2	3	4	5	6	
i=0.2	0.20	0.39	0.59	0.79	**0.98**	1.17	$\frac{d_{max}}{R}$
i=0.4	0.23	0.46	0.68	**0.91**	1.14	1.37	
i=0.6	0.26	0.53	0.79	**1.05**	1.31	1.58	
i=0.8	0.31	0.60	**0.90**	1.20	1.50	1.80	
i=1.0	0.34	0.67	**1.00**	1.34	1.67	2.01	

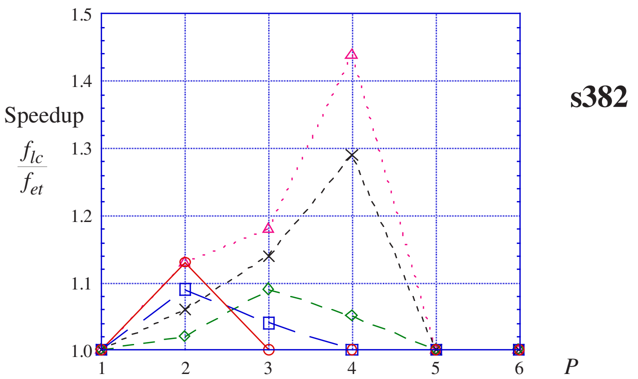


P	1	2	3	4	5	6	
i=0.2	0.59	**1.18**	1.77	2.34	2.95	3.54	$\frac{d_{max}}{R}$
i=0.4	**0.68**	1.36	2.04	2.68	3.40	4.02	
i=0.6	**0.76**	1.51	2.28	3.05	3.78	4.50	
i=0.8	**0.83**	1.65	2.48	3.33	4.16	4.96	
i=1.0	**1.02**	2.04	3.06	4.11	5.14	6.13	

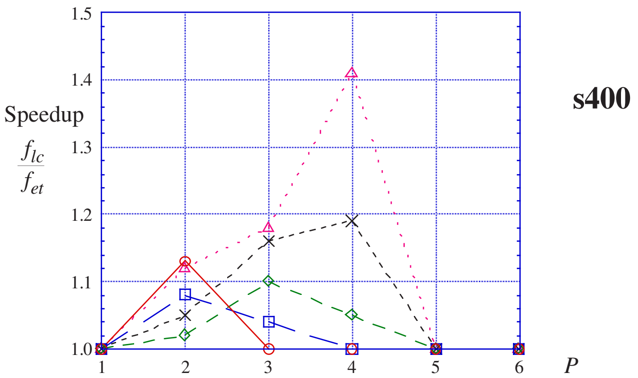
Figure 4-3: Results of experiment SP3 on the MCNC benchmark and the AT&T circuits. Each plot corresponds to a test circuit. The first row of the horizontal axis gives the pipelining degree P. Each of the next five rows corresponds to a circuit G_i for $i = 0.2, 0.4, 0.6, 0.8, 1.0$, and it gives the ratio $d_{\max}/R$ 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.



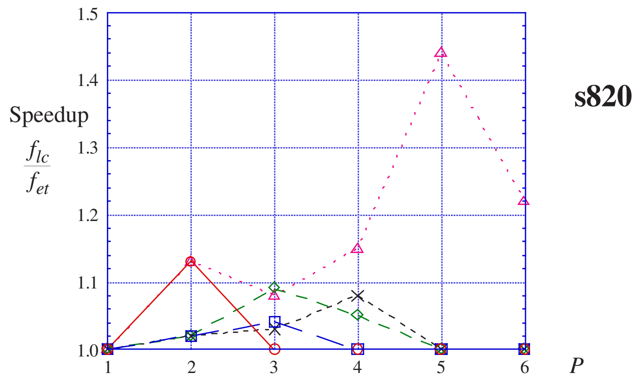
P	1	2	3	4	5	6	
i=0.2	0.15	0.29	0.44	0.59	0.73	**0.88**	$\frac{d_{max}}{R}$
i=0.4	0.19	0.39	0.58	0.77	**0.96**	1.16	
i=0.6	0.25	0.50	0.75	**1.00**	1.25	1.50	
i=0.8	0.32	0.65	**0.97**	1.29	1.61	1.93	
i=1.0	0.41	**0.83**	1.24	1.65	2.07	2.48	



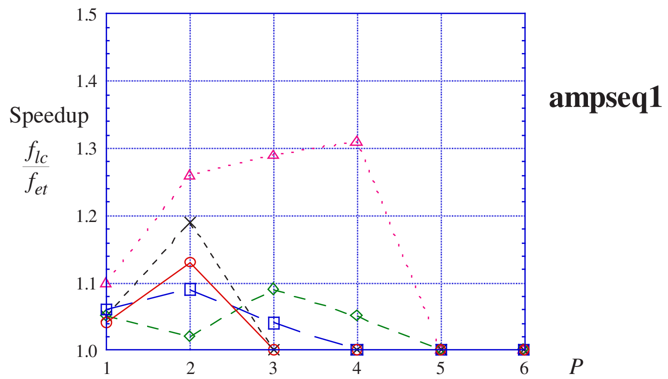
P	1	2	3	4	5	6	
i=0.2	0.23	0.47	0.70	**0.93**	1.16	1.39	$\frac{d_{max}}{R}$
i=0.4	0.27	0.54	0.81	**1.08**	1.35	1.62	
i=0.6	0.31	0.62	**0.93**	1.24	1.55	1.86	
i=0.8	0.35	0.70	**1.05**	1.40	1.75	2.10	
i=1.0	0.39	0.78	**1.16**	1.55	1.94	2.33	



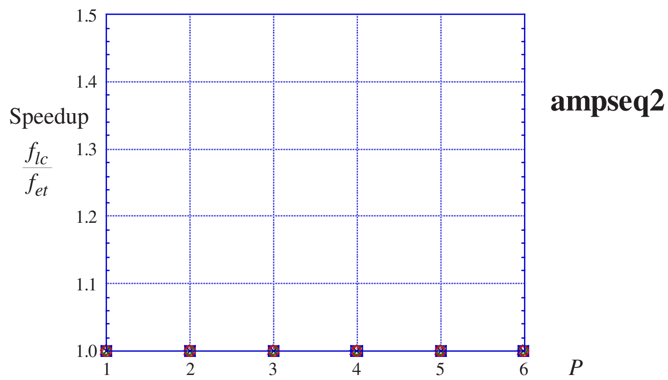
P	1	2	3	4	5	6	
i=0.2	0.24	0.47	0.71	**0.95**	1.19	1.43	$\frac{d_{max}}{R}$
i=0.4	0.28	0.56	0.83	**1.11**	1.39	1.67	
i=0.6	0.32	0.65	**0.97**	1.30	1.62	1.94	
i=0.8	0.37	0.74	**1.11**	1.49	1.86	2.22	
i=1.0	0.42	**0.84**	1.26	1.68	2.10	2.52	



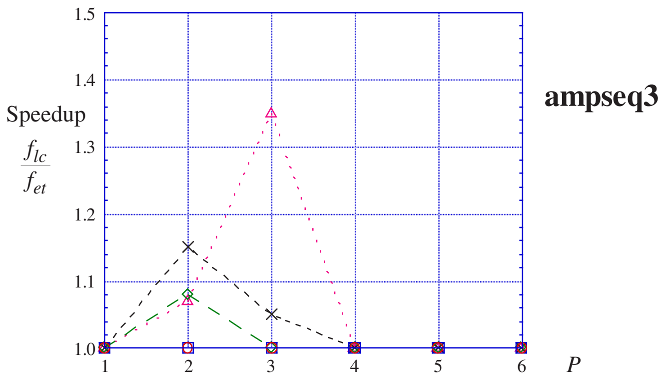
							$\frac{d_{max}}{R}$
i=0.2	0.17	0.33	0.50	0.67	0.84	**1.01**	
i=0.4	0.24	0.48	0.72	**0.97**	1.20	1.44	
i=0.6	0.33	0.66	**1.00**	1.33	1.66	2.00	
i=0.8	0.43	**0.87**	1.30	1.73	2.16	2.59	
i=1.0	0.53	**1.07**	1.61	2.14	2.68	3.21	



							$\frac{d_{max}}{R}$
i=0.2	0.32	0.65	**0.97**	1.29	1.61	1.93	
i=0.4	0.45	**0.90**	1.35	1.80	2.20	2.65	
i=0.6	0.61	**1.21**	1.82	2.43	3.03	3.64	
i=0.8	**0.80**	1.60	2.40	3.19	4.00	4.80	
i=1.0	**0.97**	1.93	2.90	3.87	4.83	5.80	

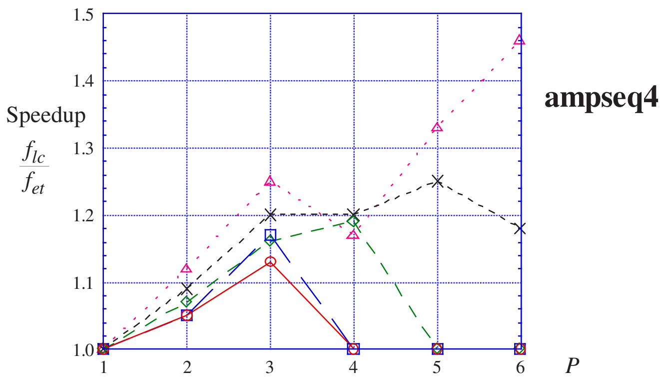


							$\frac{d_{max}}{R}$
i=0.2	0.59	**1.17**	1.76	2.32	2.93	3.51	
i=0.4	0.66	**1.34**	1.99	2.65	3.36	3.97	
i=0.6	**0.74**	1.47	2.20	2.94	3.70	4.40	
i=0.8	**0.80**	1.59	2.39	3.20	3.98	4.73	
i=1.0	**0.85**	1.69	2.54	3.40	4.23	5.07	



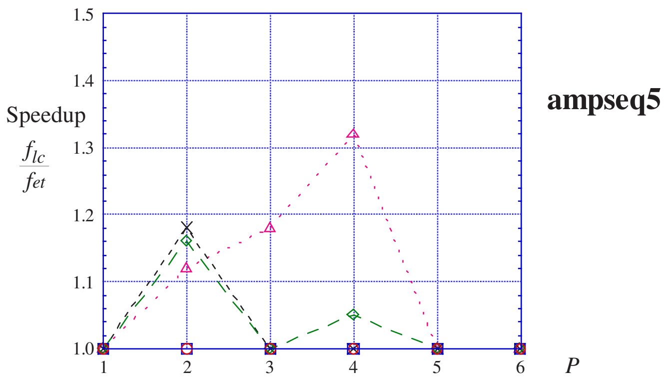
P	1	2	3	4	5	6
i=0.2	0.33	0.65	**0.97**	1.30	1.62	1.96
i=0.4	0.41	**0.82**	1.23	1.64	2.05	2.46
i=0.6	0.51	**1.02**	1.54	2.04	2.56	3.07
i=0.8	0.59	**1.19**	1.78	2.38	2.97	3.55
i=1.0	**0.67**	1.35	2.03	2.69	3.38	4.04

$\frac{d_{max}}{R}$



P	1	2	3	4	5	6
i=0.2	0.17	0.35	0.52	0.70	0.86	**1.05**
i=0.4	0.21	0.42	0.63	0.84	**1.05**	1.26
i=0.6	0.26	0.51	0.77	**1.02**	1.28	1.54
i=0.8	0.31	0.61	**0.92**	1.23	1.53	1.84
i=1.0	0.37	0.73	**1.10**	1.46	1.83	2.19

$\frac{d_{max}}{R}$



P	1	2	3	4	5	6
i=0.2	0.29	0.59	**0.88**	1.17	1.46	1.76
i=0.4	0.41	**0.82**	1.23	1.64	2.05	2.46
i=0.6	0.50	**1.00**	1.50	2.00	2.50	3.00
i=0.8	0.59	**1.17**	1.76	2.35	2.94	3.52
i=1.0	**0.67**	1.34	2.02	2.68	3.36	4.03

$\frac{d_{max}}{R}$

4.3 Latch Count Minimization Experiments

In this section 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.

4.3.1 Experimental Methodology

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 edge-triggered latches. Then, we retimed the corresponding two-phase circuit in order to achieve the same period with the minimum number of level-clocked latches. We compared the number of level-clocked latches in the two optimal circuits, where each edge-triggered latch counted as two level-clocked latches.

LM2 We retimed the original edge-triggered circuit, in order to achieve its original clock period specification with the minimum number of edge-triggered latches. Then, we retimed the corresponding two-phase circuit, in order to achieve the same period with the minimum number of level-clocked latches. We compared the number of level-clocked latches in the two optimal circuits, where each edge-triggered latch counted as two level-clocked latches.

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 count 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.

Circuit	clock period	min count in edge-triggered	min count in two-phase	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 4-4: 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 level-clocked latches in the edge-triggered implementation after retiming (this number equals twice the number of edge-triggered latches in the circuit), the minimum number of level-clocked latches in the two-phase, level-clocked implementation after retiming, and the reduction in the number of storage elements with respect to edge-triggering.

Circuit	clock period	min count in edge-triggered	min count in two-phase
`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 4-5: 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 level-clocked latches in the edge-triggered implementation after retiming (this number equals twice the number of edge-triggered latches in the circuit), and the minimum number of level-clocked latches in the two-phase implementation after retiming. Note that the level-clocked latch count decreases only for `ampseq3`.

4.3.2 Experimental Results

Our experimental results for the two sets of experiments are shown in Figures 4-4 and 4-5. There is a striking difference betwen 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.

In the experimental results of Figure 4-4 the greatest reductions were achieved for two controller circuits. The number of level-clocked 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 4-5 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 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 in Figures 4-4 and 4-5 that give the latch counts in the edge-triggered designs. Two-phase clocking can decrease this number without degrading circuit performance. In fact, as it is evident from the columns in Figures 4-4 and 4-5 that give the latch counts in the level-clocked designs, the number of level-clocked latches in more than half of the aggressive two-phase implementations is not more than 15% higher than the number of level-clocked latches in the low-performance implementations.

4.4 Conclusion

In this chapter we presented an empirical comparison of edge-triggering and two-phase level-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 solely 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 level-clocking, except when the average propagation 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, our experiments indicate that symmetric clocking schemes seem to perform as well as tuned clocking schemes.

In terms of number of storage elements, two-phase level-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.

We urge the reader not to interpret our results in a narrow quantitative way, since our tool may have introduced round-off errors. There are two qualitative conclusions that should be drawn from our timing experiments, though. First, under our assumed timing model, one should not expect to achieve automatic speedups simply by switching from edge-triggering to level-clocking and then by retiming for maximum speed. Second, when the clock period of the designs is determined by the constraints around cycles rather than the constraints along paths, level-clocking can automatically achieve greater speedups with respect to edge-triggering, because it can accommodate more effectively whatever little slack there is for the computations along paths. Edge-triggering doesn't have this flexibility.

We believe that a more extensive experimentation is essential to obtain more conclusive results regarding the relative merits of the two clocking methodologies. First, it is necessary to experiment with a wider variety of circuits. Some recent results that we obtained with proprietary circuits from Digital Equipment Corporation are in accord with the results we presented in this chapter. Those circuits, however, were no bigger than our benchmark circuits. We believe it is important to experiment with larger circuits. It is also importnat to experiment with circuits that were originally designed as two-phase circuits. Another interesting question that should be further investigated involves asymmetric clocking schemes. What design methodology would really favor the use of asymmetric clocking schemes? Can asymmetric clocking schemes decrease the number of storage elements even further than symmetric ones? We believe that the best way to answer these questions and address many other practical concerns regarding level-clocking is to implement actual circuits using our tool and explore the existing possibilities in practice.

Directions for Further Research

There are many interesting, important and fruitful research directions in the area of timing analysis and optimization of synchronous systems. Some of these directions lead to problems that are direct extensions of the work presented in this thesis. Others lead to new and unexplored terrain. In this chapter we will discuss some of these more challenging directions.

A rich and largely unexplored field is the area of algorithms for interactive analysis of circuit timing. TIM incorporates some elements of interactive analysis by means of its sensitivity analysis functions. Can we offer similar functions for retiming? Given that it may not always be possible to retime every part of a circuit, are there efficient algorithms that would allow us to identify the parts of a circuit that are most promising to retime? Are there efficient schemes that would allow us to break a big design into smaller parts, retime each of these parts separately, and then combine them again in a single faster design? Algorithms that perform these tasks will have enormous impact on the development of high-level interactive tools for the design of large circuits.

Several important issues need to be resolved before retiming becomes a widely used timing optimization technique. We believe that verification and modeling are the two most crucial among these issues. Verification is used extensively at every stage of circuit design. Retiming changes the circuit architecture, and the retimed circuits must be verified once again. The challenging task in this case is to compute for a specific set of circuit inputs, the new values that must be stored in the latches at any time during the circuit's operation. It is not clear how to perform this computation, when retiming moves latches across logic that can generate the same output with different input vectors. Modeling is another important issue with retiming. The delay model that is used in the retiming literature assigns a fixed worst-case propagation delay to each logic gate. As latches are relocated, however, the loads of the logic gates change, and consequently their propagation delays change. These changes are not accounted for in the model. Proponents of retiming argue that one could always size the latch transistors to maintain the same delay characteristics in the circuit. But this solution may not be applicable in designs with standard cells that can only use latches from a fixed pool. It is essential, therefore, to investigate retiming using more realistic delay models and to identify properties in these models that can lead to efficient, polynomial-time algorithms.

Two topics that have not been explored yet in the context of multiphase circuits are power dissipation and clock skew optimization. It is possible to retime edge-triggered circuitry in a way that reduces its power dissipation. Since there are more latches available in corresponding multiphase circuitry, retiming may be more effective for such circuitry. Note that for multiphase circuitry, the argument about increased power dissipation due to multiple clock lines is not always an issue, because we can generate a multiphase clocking scheme by distributing a single clock and then inverting the clocking waveform locally to generate the desired waveforms. Clock skew optimization remains a difficult problem even for conventional edge-triggered circuits with a single clock. Naturally, the problem is more pronounced in multiphase circuits. At the same time, however, multiphase circuits offer more ways to tolerate clock skew than conventional edge-triggered circuits. Since the successful implementation of high-performance systems depends heavily on the accurate timing, we believe it is important to investigate the potential of multiphase clocking in this direction.

Another important issue that remains unresolved in timing is the smooth and efficient transition between different levels of abstraction. As circuits increase in size and density, it is essential to develop high-level tools that will harness the ensuing complexity. At the same time, however, it is essential to have efficient and reliable low-level tools, such as a transistor-level timing analyzer and optimizer for level-clocked circuits. When both high-level and low-level tools are combined, the designer must be able to move freely across the different levels of abstraction. What are good models that simplify problems without sacrificing accuracy? How could a designer zoom in and out between the different levels of abstraction in order to examine design choices more closely? Where should one draw the line between the different abstractions? How could one provide effectively the additional computational power required for supporting solutions to these problems? The significance of these questions is not restricted to timing. These questions are fundamental in computer-aided design, and their answer requires a concerted effort from researchers who are knowledgeable both in computational issues and in circuit design issues.

Appendix A

A.1 Constraints for Proper Timing

In this appendix we prove Proposition 13, the fundamental premise of the algorithms in this thesis. The proof relies on [20], and a complete understanding of the proof requires familiarity with the material in that paper. For convenience, however, we give a brief description of the notions of "computational expansions," "proper timing," and "Δ-constraints" that are the basis of the proof.

In this appendix, we adopt the circuit representation of [20]. In that representation, a circuit is a graph $G = \langle V, E \rangle$, where each vertex $v \in V$ represents either a functional element or a level-clocked latch, and each functional element and level-clocked latch is represented by a distinct element of V. Edges in E represent only direct component-to-component interconnections and have no weights associated with them. Though each element of V continues to have associated with it a propagation delay (equal to zero for latches) and a phase (the controlling clock for latches), the functions d and χ are not explicitly included as part of the graph G.

The *computational expansion* G_{CX} of a circuit $G = \langle V, E \rangle$ is a circuit that performs the same computation as G, but in a "combinational" fashion. Construction of G_{CX} essentially requires making multiple copies of the components in G and connecting them together in such a way that for every change in the output of some component in G, there exists a distinct copy of the component, in the computational expansion, which computes the new value of the output. Those familiar with optimizing compilers may find it helpful to think of the computational expansion as an "unrolling" of the cycles, or "loops," in a level-clocked circuit. We generally denote by v_t a copy of $v \in V$ that exists due to a change in the output of v, that is caused by a clock transition that occurs at time t. The results of [20] are based on the observation that there exists a strong correlation between the operation of G and the operation of the corresponding computational expansion G_{CX}.

The computational expansion of a two-phase, level-clocked circuit is defined as the circuit $G_{CX} = \langle V_{CX}, E_{CX} \rangle$, where

$$\begin{aligned} V_{CX} &= \{v_t : v \in V \text{ and either } \phi_{\chi(v)} \text{ rises at time } t, \text{ or } t = -\infty\} \\ E_{CX} &= \{u_t \to v_{t'} : u \to v \in E; u_t, v_{t'} \in V_{CX}; t \le t', \text{ and no clock rises during } (t, t')\}, \end{aligned}$$

and the delays of the components are defined in the natural fashion. The definition presumes that there exists a "start-time" t_0 before which all clock waveforms maintain a constant value, and thus, the circuit is essentially "idle" over the interval $(-\infty, t_0)$. The existence of a copy $v_{-\infty}$ for each $v \in V$ reflects the assumption that the circuit is initialized at time $-\infty$ into a well-defined "ground" state before it begins to compute. Furthermore, each $v_t \in V_{CX}$ has associated with it an *up-time* of t and a *down-time* of $t + \phi_{\chi(v)}$. This definition of G_{CX} differs from the definition that appears in [20]. The two definitions are equivalent for two-phase circuits, however, as we show later in Lemma 56.

Intuitively, a level-clocked circuit is *properly timed* if whenever a latch holds a value (*i.e.*, whenever its clock input is LOW), it holds the same value it would in an identical circuit in which all functional elements have zero propagation delay. Ishii and Leiserson [20] show that a level-clocked circuit is properly timed if and only if its computational expansion is properly timed. Moreover, they show that if, for any latch-to-latch path (possibly itself containing several latches) in the computational expansion, the difference between the up-time associated with the starting latch and the down-time associated with the ending latch is shorter than the propagation delay between the two latches, then the expansion is not properly timed. Conversely, they show that if, for all paths between latches in the computational expansion, the propagation time does not exceed this "up-to-down" time, then the circuit *is* properly timed (see Theorem 4.1 of [20]). The infinite set of linear inequalities that compare up-to-down times with propagation delays is called the set of *Δ-constraints* for the circuit. Formally, the set of Δ-constraints for a two-phase, level-clocked circuit G can be defined as

$$\Delta = \{d(\sigma) \leq t'' - t : \begin{array}{l} v_{t'} \text{ has down-time } t'', u_t \text{ has up-time } t, \text{ and} \\ \sigma \text{ is a path in } G_{CX} \text{ from } u_t \text{ to } v_{t'}\}, \end{array}$$

where $d(\sigma)$ equals the total propagation delay of all components in the path σ.

We can now prove the following lemma.

Lemma 55 *Let u_t and $v_{t'}$ be latches in the computational expansion G_{CX} of a two-phase, level-clocked circuit G. Moreover, let $\sigma = v_0, v_1, \ldots, v_k$ be a path from u_t to $v_{t'}$ in G_{CX}, and let $\sigma' = v'_0, v'_1, \ldots, v'_k$ be a path from u to v in G such that for $i = 0, 1, \ldots, k$, if $v_i = w_{t'''} \in G_{CX}$ then $v'_i = w \in V$. Then the following statements hold.*

(i) The up-to-down time $t'' - t$ satisfies

$$t'' - t = \begin{cases} \pi\left(\frac{l(\sigma')-1}{2}\right) + \phi_{\chi(u)} & \text{if } \chi(u) = \chi(v), \\ \\ \pi\left(\frac{l(\sigma')}{2}\right) - \gamma_{1-\chi(u)} & \text{if } \chi(u) \neq \chi(v); \end{cases} \tag{A.1}$$

where $l(\sigma')$ denotes the number of latches in the path σ'.

(ii) The Δ-constraint $d(\sigma) \leq t'' - t$ holds if and only if

$$d(\sigma') \leq \begin{cases} \pi\left(\frac{l(\sigma')-1}{2}\right) + \phi_{\chi(u)} & \text{if } \chi(u) = \chi(v), \\ \\ \pi\left(\frac{l(\sigma')}{2}\right) - \gamma_{1-\chi(u)} & \text{if } \chi(u) \neq \chi(v). \end{cases} \tag{A.2}$$

Proof. First, we show that statement (i) holds by induction on the number of latches $l(\sigma)$. Let us consider the basis of the induction for $l(\sigma) = 2$. In this case we have $\chi(v_0) \neq \chi(v_k)$. By definition of the fall-time t'' we obtain

$$\begin{aligned} t'' &= t' + \phi_{\chi(v_k)} \\ &= t' + \phi_{\chi(v'_k)} \ , \end{aligned} \tag{A.3}$$

since $\chi(v_k) = \chi(v'_k)$. By definition of G_{CX} we have

$$\begin{aligned} t' &= t + (\phi_{\chi(v_0)} + \gamma_{\chi(v_0)}) \\ &= t + (\phi_{\chi(v'_0)} + \gamma_{\chi(v'_0)}) \ , \end{aligned} \tag{A.4}$$

since $\chi(v_0) = \chi(v'_0)$. Thus, by substitution of Equations (A.3) and (A.4), and the definition of a two-phase clocking scheme, we obtain

$$\begin{aligned} t'' - t &= \left(t' + \phi_{\chi(v'_k)}\right) - t \\ &= \left(t + \phi_{\chi(v'_0)} + \gamma_{\chi(v'_0)}\right) + \phi_{\chi(v'_k)} - t \\ &= \phi_{\chi(v'_0)} + \gamma_{\chi(v'_0)} + \phi_{\chi(v'_k)} \\ &= \phi_{\chi(v'_0)} + \gamma_{\chi(v'_0)} + \phi_{1-\chi(v'_0)} \\ &= \pi - \gamma_{1-\chi(v'_0)} \ , \end{aligned}$$

and, since $l(\sigma) = l(\sigma')$, Equation (A.1) is satisfied. The base case for $l(\sigma) = 3$ can be shown similarly. For the inductive step, we assume that the lemma holds for all paths σ such that $l(\sigma) < m$, and in a way similar to the base case we can show that Equation (A.1) holds for all σ such that $l(\sigma) = m$.

Statement (ii) follows immediately from Equation (A.1) and the fact that $d(\sigma') = d(\sigma)$ by construction of σ'. □

Assuming that the definition of G_{CX} specifies a proper computational expansion, the proposition now follows immediately.

Proposition 1 *A two-phase, level-clocked circuit is properly timed if and only if for all latches A and B in the circuit, the propagation delay $d(p)$ along any path p from A to B is no greater than the rise-to-fall time $\tau(p)$ of the path.*

Proof. Since there is a one-to-one correspondence between circuit components in a two-phase, level-clocked circuit and vertices in the graph representation from [20], and $\tau(p)$ is exactly equal to the value denoted by the expression "$t'' - t$" in Lemma 55, the proposition follows from Lemma 55 and Theorem 4.1 from [20]. □

All that remains to be shown is that the definition of G_{CX} is, in fact, equivalent to the definition of the computational expansion that appears in [20].

Unlike the simplified definition presented above, the definition of G_{CX} from [20] makes reference to a *base-step function* $\widehat{B}$ that maps pairs (v, k), where $v \in V$ and $k = -1, 0, 1, 2, \ldots$, to the integers $[-1, 0, 1, 2, \ldots]$. The integer argument k and the integer result are indexes into the infinite sequence of maximal time intervals over which the clocks of the circuit maintain a constant value. For example, in a two-phase, nonoverlapping clocking scheme, the index -1 corresponds by convention to the interval $(-\infty, t_0)$, the index 0 corresponds to the first high-pulse of ϕ_0, the index 1 corresponds to the first gap after the high-pulse of ϕ_0, the index 2 corresponds to the first high-pulse of ϕ_1, and so forth. Each such maximal time interval is called a *step*. Intuitively, if $\widehat{B}(v,i) = i$, then the clock transition at the beginning of the ith step directly causes a change in the value output by v. If $\widehat{B}(v,i) < i$, then the value output by v changes because of a clock transition that occured at a step earlier than i. Given a circuit $G = \langle V, E\rangle$, the computational expansion [20] is defined to be $\langle V_{CX}, E_{CX}\rangle$, where

$$\begin{array}{rcl} V_{CX} & = & \{v_k : v \in V \text{ and } \widehat{B}(v,k) = k\}, \\ E_{CX} & = & \{u_l \to v_k : u \to v \in E, \widehat{B}(u,k) = l, \text{ and } \widehat{B}(v,k) = k, \}, \end{array}$$

and

$$\widehat{B}(v,k) = \begin{cases} \max_{(u,v)\in E} \widehat{B}(u,k) & \text{if } k \neq -1 \text{, and } v \text{ is a functional element;} \\ \widehat{B}(v,k-1) & \text{if } k \neq -1 \text{, and } v \text{ is a latch whose clock is} \\ & \text{LOW during step } k; \\ \widehat{B}(v,k-1) & \text{if } k \neq -1,\ u \to v \in E,\ v \text{ is a latch whose} \\ & \text{clock is HIGH during step } k \text{, and} \\ & \widehat{B}(v,k-1) \geq \widehat{B}(u,k); \\ k & \text{if } k \neq -1,\ u \to v \in E,\ v \text{ is a latch whose} \\ & \text{clock is HIGH during step } k \text{, and} \\ & \widehat{B}(v,k-1) < \widehat{B}(u,k); \\ k & \text{if } k \neq -1 \text{, and } v \text{ is a latch whose clock is} \\ & \text{HIGH during step } k \text{ and LOW during steps} \\ & -1 \text{ through } k-1; \\ -1 & \text{if } k = -1. \end{cases}$$

The definition is somewhat complex, due to the fact that the definition from [20] applies to a more general class of circuits. The following lemma shows, however, that for two-phase, level-clocked circuits, this definition is equivalent to the definition presented in the beginning of this appendix.

Lemma 56 *Let $G = \langle V, E\rangle$ be a two-phase, level-clocked circuit that employs a clocking scheme $\pi = \langle \phi_0, \gamma_0, \phi_1, \gamma_1\rangle$, and let G_{CX} be its computational expansion with base-step-function $\widehat{B}$. Then, the following statements hold.*

(i) For every vertex $v \in V$, we have $\widehat{B}(v,k) = k$ if and only if either the input phase of v makes a LOW*-to-*HIGH *transition at the start of the step denoted by k or the step denoted by k is $(-\infty, t_0)$.*

(ii) For every edge $u \to v \in E$, if $\widehat{B}(v,k) = k$, then $0 \leq \widehat{B}(v,k) - \widehat{B}(u,k) \leq 2$, that is, step $\widehat{B}(u,k)$ never preceeds step $\widehat{B}(v,k)$, and no clock rises between steps $\widehat{B}(u,k)$ and $\widehat{B}(v,k)$.

Proof. We first show that statement (i) holds.

($\Rightarrow$) The proof is a straightforward case analysis on the possible values of k, and it is based on the definition of $\widehat{B}$ and the fact that the two phases in π are not overlapping.

For $k = -1$, the last branch in the definition of $\widehat{B}$ specifies that $\widehat{B}(v, -1) = -1$ for all $v \in V$, which by convention denotes the interval $(-\infty, t_0)$.

For $k > -1$, we consider latches and functional elements separately. If v is a latch, $\widehat{B}(u, k) = k$ only if either the fourth or the fifth branch in the definition of $\widehat{B}$ apply. In both cases, the phase that controls v must be HIGH. Since ϕ_0 and ϕ_1 are nonoverlapping, we conclude that the controlling phase is LOW during step $k-1$, and thus, it makes a LOW-to-HIGH transition at step k. If v is a functional element, we can always find a latch v' that leads to v along a latch-free path, since there are no latch-free cycles in the circuit. In this case, the first branch in the definition of $\widehat{B}$ applies for every functional element on the path, and we obtain $\widehat{B}(v', k) = \widehat{B}(v, k)$. Thus, the problem reduces to the case $\widehat{B}(v', k) = k$, where v' is a latch, and statement (i) holds in the forward direction.

($\Leftarrow$) The proof of the backward direction is by induction on the number of steps k. For simplicity, we give the proof only for latches. As in the forward direction, the proof for functional elements is a straightforward reduction to the proof for latches.

The basis of the induction for $k = -1$ follows from the last branch of $\widehat{B}$'s definition. The basis for $k \leq 3$ follows from the fifth branch of $\widehat{B}$'s definition.

For the inductive step, consider $k \geq 4$, and let v be a latch whose input phase is ϕ_0. (The case for ϕ_1 is similar.) According to the definition of the base-step function, the value $\widehat{B}(v, k)$ depends on $\widehat{B}(v, k-1)$ and $\widehat{B}(u, k)$, where $u \to v$ is an edge in G. The phase ϕ_0 is LOW during steps $k-1$ through $k-3$, since ϕ_0 and ϕ_1 are not overlapping, and ϕ_0 rises at the start of step k. By the second branch in the definition of $\widehat{B}$, it follows that $\widehat{B}(v, k-1) = \widehat{B}(v, k-2) = \widehat{B}(v, k-3) = \widehat{B}(v, k-4)$. By the inductive hypothesis, the lemma holds for all integers smaller than k, and thus, $\widehat{B}(v, k-4) = k-4$, since ϕ_0 rises at the start of step $k-4$. Moreover, since the input phase of u must be ϕ_1, we infer using a similar reasoning that $\widehat{B}(u, k) = \widehat{B}(u, k-1) = \widehat{B}(u, k-2) = k-2$. Since $k-4 < k-2$, the fourth branch of the definition applies, and it follows that $\widehat{B}(v, k) = k$. Therefore, statement (i) also holds in the backward direction.

Now, we show that statement (ii) holds.

In order to show that $\widehat{B}(u, k) \leq \widehat{B}(v, k)$, it suffices to prove that $\widehat{B}(u, i) \leq i$, for every vertex $u \in V$ and for every integer $i \geq -1$. This proof is a straightforward induction on i that directly applies the definition of $\widehat{B}$.

In order to show that $\widehat{B}(v, k) \leq \widehat{B}(u, k) + 2$, we consider functional elements and latches separately. If v is a functional element, then $\widehat{B}(u, k) = \widehat{B}(v, k)$, and the inequality holds. If v is a latch, then the clocking input of u is LOW, since $\widehat{B}(v, k) = k$ implies that the clocking input of v is HIGH, and the two phases are nonoverlapping. If u is a latch, then $\widehat{B}(u, k) = \widehat{B}(u, k-1) = \widehat{B}(u, k-2)$ from the second branch in the definition of $\widehat{B}$. (If u is a functional element, then these equalities still hold by virtue of the first branch in $\widehat{B}$ and the fact that there are no latch-free cycles in the circuit.) Statement (i) and the fact that the clocking input of u rises at $k-2$ yield $\widehat{B}(u, k-2) = k-2$. Thus, the inequality is satisfied with equality in this case, and therefore, statement (ii) holds. $\square$

A.2 Upper Bound on Relative Speedup

In this section we prove an upper bound on the relative speedup that can be achieved by two-phase level-clocking over edge-triggering. This bound is expressed in terms of the maximum gate delay $d_{\max}$ and the maximum delay-to-register ratio R around the cycles of the edge-triggered circuit.

The following two theorems give bounds on the clock period that can be achieved by retiming and tuning. The first theorem is a restatement of Theorem 3 which provides a lower and an upper bound for retiming edge-triggered circuitry. The only difference between the two statements is on the left-hand side of the theorem's inequality which now includes the obvious lower bound $d_{\max}$ on the clock period of the circuit.

Theorem 57 ([45]) *Let $G_{et} = \langle V, E, d, w \rangle$ be an edge-triggered circuit, with delay $d(v)$ for each gate $v \in V$ and $w(e)$ latches on each wire $e \in E$. Let*

$$R = \max_{C \in G_{et}} \frac{\sum_{v \in C} d(v)}{\sum_{e \in C} w(e)}$$

be the maximum ratio of total delay over total number of edge-triggered latches in the circuit G_{et}. Moreover, let d_{max} denote the maximum gate delay in G_{et}, and let $\Phi_{\min}(G_{et})$ denote the minimum clock period that we can obtain by retiming G_{et}. Then

$$\max\{d_{max}, R\} \leq \Phi_{\min}(G_{et}) \leq R + d_{max} \ . \qquad \square$$

The second theorem provides a lower bound for retiming and tuning two-phase circuitry.

Theorem 58 ([21]) *Let $G_{lc} = \langle V, E, d, w \rangle$ be a two-phase circuit, with delay $d(v)$ for each gate $v \in V$ and $w(e)$ latches on each wire $e \in E$. Let*

$$S = 2 \cdot \max_{C \in G_{lc}} \frac{\sum_{v \in C} d(v)}{\sum_{e \in C} w(e)}$$

be the maximum ratio of delay over number of latches around the cycles in G_{lc}. Then the minimum clock period $\Phi_{\min}(G_{lc})$ we can obtain by retiming and tuning G_{lc} satisfies

$$\max\{d_{max}, S\} \leq \Phi_{\min}(G_{lc}) \ . \qquad \square$$

Note that for a given edge-triggered circuit and its corresponding two-phase, level-clocked implementation that is obtained by replacing each edge-triggered latch with a pair of level-clocked latches, the lower bounds R and S are equal.

We use the bounds in Theorems 57 and 58 to prove the following upper bound on the speedup that can be achieved by switching from edge-triggering to level-clocking.

Lemma 59 *Let G_{et} be an edge-triggered circuit, and let G_{lc} be a two-phase circuit that is obtained by replacing each edge-triggered latch in G_{et} by a pair of level-clocked latches. Let $\Phi_{\min}(G_{et})$ be the minimum clock period that we can achieve by retiming G_{et}, and let $\Phi_{\min}(G_{lc})$ be the minimum clock period that we can achieve by retiming G_{lc} and simultaneously tuning its clocking scheme. Then*

$$\frac{\Phi_{\min}(G_{et})}{\Phi_{\min}(G_{lc})} \leq \frac{R + d_{\max}}{\max\{R, d_{\max}\}} \ . \qquad \square$$

The ratio $(R + d_{\max}) / \max\{R, d_{\max}\}$ can be used as a predictor of the speedup that may be achieved by two-phase clocking. The simple intuition behind this heuristic is that for greater values of the upper bound we have more space for improvement. There is no guarantee, however, on the actual improvement, since we have no lower bounds. The upper bound in maximized for $R = d_{\max}$. As we see in the experimental results of Section 4.2, it is when R approaches $d_{\max}$ that two-phase clocking becomes faster than edge-triggering.

Bibliography

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

[2] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

[3] 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.

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

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

[6] 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.

[7] G. B. Dantzig, W. O. Blattner, and M. R. Rao. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. *Theory of Graphs*, 1967. P. Rosenstiehl, editor, Dunod, Paris, and Gordon and Breach, New York, pp. 77–84.

[8] S. Even and A. Litman. On the capabilities of systolic systems. *3rd ACM Symposium on Parallel Algorithms and Architectures*, July 1991.

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

[10] R. W. Floyd and R. L. Rivest. Expected time bounds for selection. *Communications of the ACM*, 18(3):165–172, 1975.

[11] 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.

[12] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Computing*, October 1989.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., San Francisco, 1979.

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

[15] M. D. Grigoriadis. An efficient implementation of the network simplex method. *Mathematical Programming Study*, 26:83–111, 1986.

[16] 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, October 1991.

[17] R. B. Hitchcock. Timing verification and the timing analysis program. In *Proc. 19th Design Automation Conference*, pages 594–604, June 1982.

[18] C. A. R. Hoare. Algorithm 63 (partition) and algorithm 65 (find). *Communications of the ACM*, 4(7):321–322, 1961.

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

[20] 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.

[21] 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.

[22] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. Polynomial algorithms for optimizing two-phase, level-clocked circuitry. 1992. Unpublished Manuscript.

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

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

[25] E. L. Lawler. *Combinatorial Optimization, Networks and Matroids.* Holt, Rinehart and Winston, New York, 1976.

[26] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees and Hypercubes.* Morgan Kaufman, 1992.

[27] C. E. Leiserson. *Area-efficient VLSI Computation.* PhD thesis, Carnegie-Mellon University, 1981. Published in book form by the MIT Press, Cambridge, Massachusetts, 1983.

[28] 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.

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

[30] C. E. Leiserson and J. B. Saxe. A mixed-integer programming problem which is efficiently solvable. *Journal of Algorithms*, 9:114–128, 1988.

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

[32] 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.

[33] S. Malik, E. Sentovich, R. K. Brayton, and A. Sangiovanni-Vincentelli. Retiming and resynthesis: Optimizing sequential networks with combinational techniques. In *Proc. of the Hawaii International Conference on System Sciences*, 1990.

[34] T. M. McWilliams. Verification of timing constraints on large digital systems. In *Proc. 17th Design Automation Conference*, pages 139–147, June 1980.

[35] C. A. Mead and L. A. Conway. *Introduction to VLSI Systems*. Addison-Wesley, Reading, Massachusetts, 1980.

[36] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the Association for Computing Machinery*, 31(1):114 – 127, 1984.

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

[38] 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.

[39] J. B. Orlin and R. K. Ahuja. New scaling algorithms for the assignment and minimum cycle mean problem. Technical Report 2019-88, MIT Sloan School of Management, 1988.

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

[41] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity*. Prentice-Hall, Inc., 1982.

[42] M. C. Papaefthymiou. On retiming synchronous circuitry and mixed-integer optimization. Master's thesis, Massachusetts Institute of Technology, September 1990. Available as MIT/LCS/TR-486.

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

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

[45] M. C. Papaefthymiou. Understanding retiming through maximum average-delay cycles. *Mathematical Systems Theory*, 1993. To appear. A preliminary version of the paper appeared in the proceedings of the 3rd ACM Symposium on Parallel Algorithms and Architectures, July 1991.

[46] M. C. Papaefthymiou, C. E. Leiserson, and A. T. Ishii. Optimizing two-phase, level-clocked circuitry. In *Proceedings of the 1991 MIT Student Workshop on VLSI and Parallel Systems*, July 1991.

[47] 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.

[48] 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.

[49] N. Park and A. C. Parker. Theory of clocking for maximum execution overlap of high-speed digital systems. *IEEE Transactions on Computers*, 37(6):678–690, June 1988.

[50] K. H. Randall. Edge-triggering vs. level-clocking. Bachelor's thesis, Massachusetts Institute of Technology., 1993.

[51] 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.

[52] 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.

[53] N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Graph algorithms for clock schedule optimization. In *Digest of Technical Papers of the 1992 IEEE/ACM International Conference on CAD*, November 1992.

[54] 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.

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

[56] 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.

[57] 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.

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