LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-486

ON RETIMING SYNCHRONOUS CIRCUITRY AND MIXED-INTEGER OPTIMIZATION

Marios Christos Papaefthymiou

September 1990

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

MIT/LCS/TR-486

ON REVERSIBLE TRANSDUCER CIRCUITS

Mario Christos Papaefthymiou

September 1990

On
Retiming Synchronous Circuitry
and
Mixed-Integer Optimization

by

Marios Christos Papaefthymiou

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

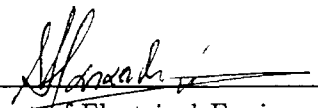
Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

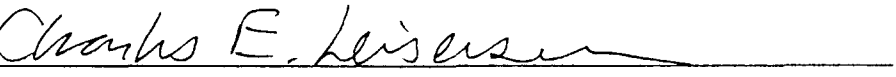
Master of Science in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology
August 1990

© Massachusetts Institute of Technology 1990

Signature of Author 
Department of Electrical Engineering and Computer Science
August 31, 1990

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

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Students

On Retiming Synchronous Circuitry and Mixed-Integer Optimization

by

Marios Christos Papaefthymiou

Submitted to the
Department of Electrical Engineering and Computer Science
on August 31, 1990
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

In this paper we investigate properties of *retiming*, a circuit transformation which preserves the behavior of the circuit as a whole. We present an algorithm which transforms a given combinational circuit into a functionally equivalent pipelined circuit with minimum latency and clock-period no greater than a given upper bound c. The algorithm runs in $O(E)$ steps, where E is the number of interconnections in the circuit, and is optimal within a constant factor. We give a novel and concise characterization of the minimum clock-period of a circuit in terms of the maximum delay-to-register ratio cycle in the circuit. We show that this ratio does not exceed the minimum feasible clock-period by more than the maximum delay D of the elements in the circuit. This characterization leads to an $O(E \lg D)$ algorithm for minimum clock-period pipelining of combinational circuitry with latency no greater than a given upper bound l, an $O(\min\{V^{1/2}E \lg(VD), VE\})$ algorithm for minimum clock-period retiming of unit-delay circuitry, an $O(VE \lg D)$ algorithm for minimum clock-period retiming of general circuitry and an $O(\min\{V^{1/2}E \lg(VW) \lg(VD), VE \lg(VD)\})$ algorithm for approximately minimum clock-period retiming, where V is the number of processing elements in the circuit. We demonstrate the closed semiring structure of retiming on unit-delay circuits under a given clock-period constraint. Finally, we give an $O(V^3 \lg V)$ algorithm for a mixed-integer optimization problem which arises in the linear programming framework of retiming.

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

Keywords: digital circuitry, systolic systems, parallel computation, computer-aided design, retiming, pipelining, propagation delay, group theory, semirings, mixed-integer optimization, network flow.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-87-K-0825

Acknowledgements

I would like to thank my advisor Charles Leiserson for his encouragement and support during the course of this thesis. In short time he taught me a lot about research and teaching.

The Theory of Computation Group at MIT's Laboratory for Computer Science provided an ideal environment for carrying out this work, both in terms of human resources as well as in terms of equipment support and organization.

I would like to thank Demetris Bertsimas of MIT's Sloan School of Management for his helpful pointers to the Operations Research literature.

Above all, I would like to thank my family for always being by my side.

Contents

1 Introduction **9**

2 Minimum Latency Pipelining **12**
2.1 Preliminaries 12
2.2 Difference Constraints and Shortest-Paths 15
2.3 The Algorithm 17

3 Minimum Clock-Period Characterization **23**
3.1 Preliminaries 24
3.2 Minimum Period for Unit-Delay Circuits 24
3.3 Minimum Period for General Circuits 25
3.4 Algorithmic Implications 30
3.4.1 Minimum clock-period pipelining 30
3.4.2 Minimum clock-period retiming 34
3.4.3 Approximately minimum clock-period retiming 35

4 The Closed Semiring Structure of Retiming **38**
4.1 Preliminaries 38
4.2 The Closed Semiring Construction 39
4.3 An Algorithm for Unit-Delay Circuitry Retiming 44

5 A Mixed-Integer Optimization Problem **50**
5.1 Preliminaries 51
5.2 Mixed-Integer Dual Minimum-Cost Flow 53
5.3 Feasibility and Optimality Conditions 54
5.4 The Algorithm 59
5.5 An Application to State Minimization 59

6 Conclusion **63**

Chapter 1

Introduction

Speed of design is essential in building large-scale, highly-complex systems. This issue becomes more apparent, since emerging VLSI technologies lead to systems of increasing size and complexity. Design automation accelerates the design process by providing tools that improve the quality of a quickly designed circuit. *Retiming*, which was introduced in [13, 14, 15] and treated in [17], is a well-known design automation technique which aims at speeding the design process, without sacrificing the quality of the implementation. Retiming optimizes clocked circuits by relocating registers so as to reduce combinational rippling. In this thesis we further investigate retiming and provide results of practical as well as theoretical interest. We present optimal algorithms for optimization of combinational circuitry. We give a novel characterization of the minimum clock-period of a circuit in terms of the maximum register-to-delay ratio cycle in the circuit, which leads to improved algorithms for minimum clock-period and approximately minimum clock-period retiming. We exhibit the group theoretical structure of retiming on circuits with unit-delay components. Finally, we give an efficient algorithm for a mixed-integer optimization problem, which arises in the linear programming framework of retiming.

In Chapter 2 we introduce the basic concepts of retiming. We define the notations and terminology and review the graph-theoretic model of digital circuits from [15, 17]. We give an algorithm that transforms a given combinational circuit into a functionally equivalent pipelined circuit with minimum latency and clock-period no greater than a given upper bound c. The algorithm runs in $O(E)$ steps, where E is the number of interconnections in the circuit, and is optimal within a constant factor. The operation

of the algorithm is based on the notion of *accumulated delay* along a path in the circuit.

In Chapter 3 we give a novel and concise characterization of the minimum feasible clock-period of a circuit in terms of the maximum *delay-to-register ratio cycle* in the circuit graph. We prove that this ratio does not exceed the minimum feasible clock-period by more than an additive factor of D, where D is the maximum delay of the processing elements in the circuit. This observation establishes a range of possible values for the minimum clock-period, that is independent of the size of the circuit. The range depends solely on the delays of the individual components used.

Based on the maximum ratio cycle characterization of the minimum clock-period we approach a variety of retiming problems. For combinational circuits we give an optimal $O(E)$ algorithm, that transforms a unit-delay combinational circuit into a pipelined circuit with minimum clock-period and latency no greater than a given upper bound l. We also give a more general $O(E \lg D)$ algorithm for the same problem on combinational circuits with arbitrary delays. We show how to obtain a minimum clock-period retiming of a unit-delay circuit in $O(\min\{V^{1/2}E \lg(VW), VE\})$ steps, where V is the number of processing elements in the circuit and W is the maximum number of registers on a wire in the circuit, by direct application of graph-theoretic algorithms for finding the minimum cycle mean in a graph [11, 20]. We demonstrate how to obtain a minimum clock-period retiming of a circuit with arbitrary delays in $O(VE \lg D)$ steps. The best previously known strongly polynomial algorithm for minimum clock-period retiming of synchronous circuitry, unit-delay or arbitrary-delay, required $O(VE \lg V)$ steps [17]. Finally, if the retimed circuit is allowed a clock-period which does not exceed the minimum possible by more than D we show how to obtain it in $O(\min\{V^{1/2}E \lg(VW) \lg(VD), VE \lg(VD)\})$ steps. The running times of the algorithms in Chapters 2 and 3 are summarized in the table of Figure 1.1.

In Chapter 4 we investigate group-theoretic properties of retiming. We demonstrate the closed semiring structure of retiming on unit-delay circuits and we give a Bellman-Ford type algorithm, with redefined additive and multiplicative operations, for unit-delay circuitry retiming. Its running time is $O(VE)$ and matches the best previously known strongly polynomial algorithm for the same problem [17].

In Chapter 5 we investigate a mixed-integer optimization problem, which arises in the linear programming framework of retiming. We give a polynomial time algorithm

Circuit Type	Transformation	Running Time
Combinational	Min Latency Pipelining	$O(E)$
UD Combinational	Min Clock-Period	$O(E)$
Combinational	Min Clock-Period	$O(E \lg D)$
UD Sequential	Min Clock-Period	$O\left(\min\left\{\begin{array}{l} V^{1/2}E\lg(VW) \\ VE \end{array}\right\}\right)$
Sequential	Min Clock-Period	$O(VE \lg D)$
Sequential	Approx Min Clock-Period	$O\left(\min\left\{\begin{array}{l} V^{1/2}E\lg(VW)\lg(VD) \\ VE\lg(VD) \end{array}\right\}\right)$

Figure 1.1: Summary of problems and running times of corresponding algorithms. For the sake of simplicity we denote a set S and its cardinality $|S|$ by the same symbol. The initials UD denote unit-delay circuitry.

for a generic mixed-integer optimization problem, that we call *restricted mixed-integer dual of an uncapacitated minimum-cost flow.* The polynomial running time is achieved by introducing a set of additional, appropriately chosen constraints. The same idea was used for the solution of a similar mixed-integer problem in [22, 16], which did not involve, however, an objective to be optimized. The technique of introducing additional constraints, or *cuts* as they are known in the literature, in order to solve mixed-integer optimization problems, is known in general to require an exponential number of steps [21, 23, 3, 18]. Aharoni, Erdös and Linial [1] pose the question whether a clever choice of cuts can yield polynomial time algorithms. We show that this is possible for the problem we consider, by choosing the cuts in a way that reduces the original mixed-integer problem to a network flow problem.

Chapter 2

Minimum Latency Pipelining

In this chapter we review the basic concepts of retiming and describe an $O(E)$ algorithm for minimum latency pipelining of combinational circuitry. The running time of the algorithm is optimal within a constant factor. The chapter is organized as follows. Section 2.1 defines the terminology used in the rest of the paper and presents a mathematical framework of retiming. Section 2.2 gives a brief overview of the relation between the problem of satisfying a given set of difference constraints and the problem of finding single-source shortest-paths in a graph. This relation serves as a basis for proving the correctness of our algorithm for minimum latency pipelining of combinational circuitry. Both the algorithm and its correctness proof are given in Section 2.3.

2.1 Preliminaries

In this section we define the notations and terminology needed in the rest of the paper and present the graph-theoretic model of digital circuits assumed. We also describe the operation of retiming and present a mathematical framework for it. The entire framework presented in this section was introduced in [13, 14, 15] and was treated thoroughly in [17].

We view a circuit abstractly as a network of *functional elements* and globally clocked *registers*. The functional elements provide the computational power of the circuit and the registers act as storage elements. Each functional element has an associated *propagation delay*. The outputs of a functional element at any time are

defined as a specified function of its inputs, provided that all the inputs have been stable for a time at least equal to the element's propagation delay.

We model a circuit as a finite, vertex-weighted, edge-weighted, directed multigraph $G = (V, E, d, w)$. The vertices of the graph model the functional elements of the circuit. Each vertex v is weighted with its numerical propagation delay $d(v)$. The directed edges E of the graph model interconnections between functional elements. Each edge $u \xrightarrow{e} v \in E$ connects an output of some functional element represented by vertex u to an input of some functional element represented by vertex v. Each edge e is labeled with a *register count* $w(e)$, which equals the number of registers along the connection. We impose the restriction that there be no directed cycles in G of zero edge-weight, thereby ensuring that no race conditions can arise. We define the ***clock-period*** $\Phi(G)$ for any synchronous circuit G as the maximum amount of propagation delay through which any signal must ripple between clock ticks.

We shall view a simple path $p = u \overset{p}{\rightsquigarrow} v$ in G as a sequence of vertices and edges, with no repetitions, that starts from a vertex u and ends at a vertex v. For any path $p = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{k-1}} v_k$, we define the *path weight* as the sum of the weights of the edges of the path:

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

We also define the *path delay* as the sum of the delays of the vertices of the path:

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

In order that a graph $G = (V, E, d, w)$ have well-defined physical meaning as a circuit, we place the restriction that the propagation delays $d(v)$ and the register counts $w(e)$ are nonnegative integers for each vertex $v \in V$ and for each edge $e \in E$.

Retiming transformations alter the clock-period of a circuit by inserting and deleting registers, but without otherwise affecting the circuit's structure. The new circuit is functionally *equivalent*, as seen by the external world, to the original. Such a proof can be found in [15], which also contains a technical definition of the term "equivalent". A *retiming* of a circuit $G = (V, E, d, w)$ is an integer-valued vertex-labeling $r : V \to \mathbf{Z}$. The retiming specifies a transformation of the original circuit in which registers are added and removed so as to change the graph G into a new graph $G_r = (V, E, d, w_r)$.

The edge-weighting w_r is defined for an edge $u \xrightarrow{e} v$ by the equation

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

and the label $r(v)$ is referred to as the *lead* of vertex v. A retiming r of a circuit is *legal* if the register counts w_r of the retimed circuit G_r are nonnegative, thus ensuring that no edge may have a negative register count.

In order to characterize the clock-period of a retimed circuit we define two quantities:

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

The quantity $W(u,v)$ is the minimum number of registers on any path from vertex u to vertex v. We call a path $u \overset{p}{\rightsquigarrow} v$ such that $w(p) = W(u,v)$ a *critical path* from u to v and we denote it by $u \overset{p_{cr}}{\rightsquigarrow} v$. The quantity $D(u,v)$ is the maximum total propagation delay on any critical path from u to v.

The following two statements about D are important:

F1 *$D(u,v)$ can take on $O(V^2)$ values.*

F2 *Given a synchronous circuit G and a retiming r of G, the clock-period $\Phi(G_r)$ is equal to $D(u,v)$ for some $u, v \in V$.*

Statements F1 and F2 are easily justified by the fact that there are $O(V^2)$ pairs of vertices in the graph and that retiming does not change the propagation delay along a critical path between any two vertices in the graph.

We can compute W and D by solving an all-pairs shortest-paths problem in G. Common ways of solving this problem are the Floyd-Warshall method [12, page 86], which runs in $O(V^3)$ and Johnson's algorithm [10], which runs in $O(VE + V^2 \lg V)$ time using the Fibonacci heap data structure due to Fredman and Tarjan [6].

The following theorem, which is proven in [17], characterizes the conditions under which a retiming produces a circuit whose clock-period is no greater than a given constant.

Theorem 2.1 *Let $G = (V, E, d, w)$ 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*

$$r(v) - r(u) \leq w(e) \tag{2.1}$$

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

$$r(v) - r(u) \leq W(u, v) - 1 \tag{2.2}$$

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

This theorem provides the basic tool needed to solve the retiming problem for a given clock-period. Notice that the constraints on the unknowns $r(v)$ in the theorem are linear inequalities involving only differences of unknowns. Using the Bellman-Ford algorithm [12, page 74] we can test whether there exists a retimed circuit with clock-period less than some constant c in $O(V^3)$ steps, since there can be $O(V^2)$ inequalities of the form (2.1). Leiserson and Saxe [17] give an asymptotically faster algorithm, which runs in $O(VE)$ steps.

2.2 Difference Constraints and Shortest-Paths

In this section we exhibit the relation between the problem of satisfying a given set of difference constraints and the problem of finding single-source shortest-paths in a graph generated by the given set of constraints. We also give without proof an important property of the single-source shortest-paths solution [12, 4]. The framework, that we develop in this section, will be used extensively in the rest of this thesis.

We consider the problem of solving the following system of difference constraints.

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

$$x_j - x_i \leq a_{ij} \tag{S}$$

on the n unknowns $x_1, x_2, \ldots, x_n$, where a_{ij} are given real constants. Determine a set of feasible values for the unknowns x_i or determine that no such set exists. □

The given system S *induces* an edge-weighted graph $G = (V, E, w)$. The vertex set V is defined as

$$V = \{v \ : \ x_v \text{ is an unknown of } S\}.$$

The edge set E is defined as

$$E = \{u \rightarrow v \ : \ x_v - x_u \leq a_{uv} \text{ is a constraint of } S\}.$$

Finally, for every edge $u \xrightarrow{e} v \in E$ we have

$$w(e) = a_{uv}.$$

Now, we define the single-source shortest-paths problem on an edge-weighted graph $G = (V, E, w)$ from a source-vertex $s \in V$.

Problem SSSP *(Single-Source Shortest-Paths)* Let $G = (V, E, w)$ be an edge-weighted graph and let s be a vertex in V. Determine a value $l(v)$ for each vertex $v \in V$ such that

$$l(v) = \min\{w(p) \ : \ s \overset{p}{\rightsquigarrow} v\}. \qquad \square$$

We give without proof three important lemmata [12].

Lemma 2.2 *Problem DC is solvable if and only if Problem SSSP is solvable.* $\square$

Lemma 2.3 *Problem SSSP is solvable if and only if there exist no directed cycles C in G with weight $w(C) < 0$.* $\square$

Lemma 2.4 *Let S be a system of m difference constraints of the form*

$$x_j - x_i \leq a_{ij}$$

on the n unknowns $x_1, x_2, \ldots, x_n$, where a_{ij} are given real constants. Let $G = (V, E, w)$ be the graph induced by S, and let $l(v)$ be the length of the shortest path in G from the source $s \in V$ to vertex v. Then the assignment $x_v = l(v)$ for each vertex $v \in V$ satisfies the constraints in S and maximizes $x_v - x_s$ for every vertex $v \in V$. $\square$

These three lemmata will be used extensively in the correctness proofs of the algorithms that we present in the rest of the thesis.

2.3 The Algorithm

This section introduces the problem of minimum latency pipelining of combinational circuitry and presents an efficient algorithm for its solution. The algorithm terminates in $O(E)$ steps, and its performance is optimal within a constant factor. Its running time is a significant improvement over the $O(VE)$ running time of the previously known techniques for the general retiming problem.

In a *combinational* circuit all register counts are zero and thus the circuit graph is acyclic. We consider the circuit to have one input interface v_I and one output interface v_O. By retiming a combinational circuit G, we can produce a *pipelined* circuit G_r which achieves a shorter clock-period at the cost of introducing a *latency* of $r(v_I) - r(v_O)$ clock ticks for signals to propagate from the input interface v_I to the output interface v_O.

The problem of *minimum latency pipelining* is defined as follows: *Given a combinational circuit $G = (V, E, d, 0)$ with input interface v_I and output interface v_O, and a positive integer c, find a legal retiming r of G such that $\Phi(G_r) \leq c$ and the latency $r(v_I) - r(v_O)$ of the retimed circuit is as small as possible.* Stated in mathematical terms, we want to solve the following problem:

Problem MLP *(Minimum Latency Pipelining)* Given a combinational circuit $G = (V, E, d, 0)$ with input interface v_I and output interface v_O, determine a value $r(v)$ for each vertex $v \in V$ that minimizes $r(v_I) - r(v_O)$ subject to

$$r(v) - r(u) \leq 0 \tag{2.3}$$

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

$$r(v) - r(u) \leq -1 \tag{2.4}$$

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

According to Section 2.2, Problem MLP can be viewed as a single-source shortest-paths problem on the *constraint* graph $G_c = (V_c, E_c, w_c)$, which is defined in the following manner.

$$V_c = V,$$

$$E_c = \{u \rightarrow v : r(v) - r(u) \text{ is constrained by (2.3) or (2.4)}\},$$
$$w_c(u \rightarrow v) = \begin{cases} 0 & \text{if } r(v) - r(u) \text{ is constrained by (2.3),} \\ -1 & \text{if } r(v) - r(u) \text{ is constrained by (2.3).} \end{cases}$$

A feasible assignment of values to the unknowns of Problem MLP can be obtained in $O(VE)$ steps by using the general techniques described in Section 2.1.

We present Algorithm MLP, which yields a solution to Problem MLP in $O(E)$ steps. The running time of the algorithm is optimal within a constant factor. For each vertex v in the graph, Algorithm MLP maintains its *stage* $|r(v)|$ and its *accumulated delay* $\delta(v)$. The stage of a vertex v is the number of registers along any path from the input interface v_I to the vertex v. The accumulated delay of a vertex v is the longest delay of a signal coming into that vertex from a preceding register. The algorithm operates as follows:

Algorithm MLP *(Minimum Latency Pipelining)* Given a combinational circuit G and a desired clock-period c, this algorithm determines a pipelined combinational circuit G_r with clock-period $\Phi(G_r) \leq c$ and minimum latency.

1. For each vertex $v \in V$, set $r(v) \leftarrow 0$ and $\delta(v) \leftarrow d(v)$.

2. Visit the edges $u \rightarrow v$ in topological sort order. For each edge $u \rightarrow v$ do:

   2.1. If $r(v) > r(u)$, then $r(v) \leftarrow r(u)$.

   2.2. If $\delta(u) + d(v) > c$ and $r(v) \geq r(u)$, then $r(v) \leftarrow r(u) - 1$.

   2.3. If $\delta(u) + d(v) > \delta(v)$ and $r(u) = r(v)$, then $\delta(v) \leftarrow \delta(u) + d(v)$.

3. For each edge $u \xrightarrow{e} v \in E$, set $w_r(e) = w(e) + r(u) - r(v)$. □

The idea behind Algorithm MLP is to visit the vertices of the graph keeping track of the longest propagation delay up to the vertex currently visited. New registers are introduced according to a greedy criterion: whenever the longest propagation delay exceeds the desired clock-period c, a pipeline stage is introduced. Visiting the edges in topological sort order ensures that whenever an edge is considered all the preceding edges in the graph have been taken into account. Step 2.1 of the algorithm ensures that no succeeding vertex belongs to a higher pipeline stage. Step 2.2 ensures

that whenever the longest propagation delay along a register-free path leading from a preceding vertex to the currently visited vertex exceeds the desired clock-period c, a new pipeline stage is introduced. Finally, step 2.3 ensures that once all the incoming edges of a vertex v have been processed, the maximum propagation delay along a register-free path leading from a preceding vertex to vertex v is maintained.

Algorithm MLP terminates, since the number of edges is finite and it executes a finite number of operations per edge. In fact the algorithm runs quickly, as is shown by the following lemma.

Lemma 2.5 *Algorithm MLP terminates in $O(E)$ steps on a circuit $G = (V, E, d, 0)$.*

Proof: Steps 1 and 3 require $\Theta(E + V)$ steps. Sorting the edges of a directed acyclic graph in topological order requires $O(E)$ time [4]. In step 2 each edge is visited exactly once and the number of operations is bounded by a constant. By the time the algorithm terminates, therefore, it has executed $O(E)$ steps, assuming $V \leq E - 1$. □

In order to demonstrate the correctness of Algorithm MLP we proceed in two stages. First we show that Algorithm MLP yields a set of values for $r(v)$ that satisfies (2.3) and (2.4). Then, we show that this set is a single-source shortest-paths solution in the constraint graph G_c, thereby ensuring, according to Lemma 2.4, that $r(v_O) - r(v_I)$ is maximized. It follows directly that the set of values $r(v)$ is a legal retiming that minimizes the latency $r(v_I) - r(v_O)$.

Lemma 2.6 *Algorithm MLP yields a solution that satisfies (2.3).*

Proof: Steps 2.1 and 2.2 of the algorithm change $r(v)$. Both steps ensure that $r(v)$ is only decreasing for every edge $u \rightarrow v$ in E. □

Lemma 2.7 *Algorithm MLP yields a solution that satisfies (2.4).*

Proof: Assume for the sake of contradiction that for some pair of vertices (u_0, u_k), there exists a path $p = u_0 \stackrel{0}{\rightarrow} u_1 \stackrel{1}{\rightarrow} \ldots \stackrel{k-2}{\rightarrow} u_{k-1} \stackrel{k-1}{\rightarrow} u_k$ in G with propagation delay $d(p) > c$ such that $r(u_k) - r(u_0) > -1$ or, equivalently, $r(u_0) \leq r(u_k)$. The inequality $r(u_0) \leq r(u_k)$ and transitive application of inequality (2.3) imply that $r(u_i) = r(u_j)$ for all vertices $u_i, u_j \in p$. In this case step 2.3 of the algorithm ensures

that $\delta(u_{k-1}) + d(u_k) \geq d(p)$ which in turn implies that $\delta(u_{k-1}) + d(u_k) > c$. When visiting vertex u_k the algorithm detects this condition in step 2.2 and enforces $r(u_k) > r(u_{k-1})$, which contradicts the fact that $r(u_i) = r(u_j)$ for all vertices $u_i, u_j \in p$. □

In order to show that the values $r(v)$ given by Algorithm MLP are a single-source shortest-paths solution in the constraint graph G_c, we must prove two basic lemmata first.

Lemma 2.8 *At any point of the algorithm, we have* $d(v) \leq \delta(v) \leq c$.

Proof: The relation $d(v) \leq \delta(v)$ clearly holds at any point of the algorithm, since initially $d(v) = \delta(v)$ and $\delta(v)$ is never decreased.

Now, for the second part of the inequality, observe that $\delta(v)$ increases in step 2.3 only. For the sake of contradiction assume that for some edge $u \rightarrow v$ the relation $\delta(v) > c$ holds after the execution of step 2.3. It follows that the preconditions $\delta(v) = \delta(u) + d(v) > c$ and $r(u) = r(v)$ of step 2.3 must have been satisfied prior to its execution. But, from the immediately previous step 2.2 we have that $r(u) > r(v)$, since $\delta(u) + d(v) > c$, which contradicts the fact that $r(u) = r(v)$. □

Lemma 2.9 *For every vertex v that has had all its incoming edges visited by the algorithm we have* $\Delta(v) \leq c|r(v)| + \delta(v)$, *where* $\Delta(v)$ *denotes the maximum possible delay from* v_I *to* v *along any path in* G.

Proof: The proof is by induction on the vertices that have had all their incoming edges visited by the algorithm. Initially, vertex v_I has had all its incoming edges trivially visited by the algorithm, since the indegree of v_I is zero, and $r(v_I) = 0$. Since the longest path from v_I to itself is the trivial path with no edges, we infer that $\Delta(v_I) = d(v_I)$. Also, we have that $\delta(v_I) = d(v_I)$. Therefore $\Delta(v_I) \leq c|r(v_I)| + \delta(v_I)$ holds.

Now, consider the inductive step. Since the edges are visited in topological sort order, whenever all the incoming edges of a vertex v have been visited all the incoming edges of the vertices u_i with edges $u_i \rightarrow v$ have been visited as well. Assume for the sake of contradiction that $\Delta(v) > c|r(v)| + \delta(v)$ holds after having visited all the incoming edges $u_i \rightarrow v$ of vertex v. Then, we have:

$$d(v) + \max\{\Delta(u_i) \ : \ u_i \rightarrow v \ \in E\} > c|r(v)| + \delta(v), \tag{2.5}$$

which implies

$$d(v) + \max\{c|r(u_i)| + \delta(u_i) \ : \ u_i \rightarrow v \in E\} > c|r(v)| + \delta(v), \tag{2.6}$$

since $\Delta(u_i) \leq c|r(u_i)| + \delta(u_i)$ by the inductive assumption. Let the maximum in the left hand side of (2.5) occur for $i = i'$. Now, consider the three possible orderings of $r(u_{i'})$ and $r(v)$:

Case 1: $r(u_{i'}) < r(v)$. This case is impossible, because steps 2.1 and 2.2 of the algorithm ensure that $r(v)$ can only decrease.

Case 2: $r(u_{i'}) = r(v)$. In this case we have from (2.6):

$$\begin{aligned} c|r(v)| + \delta(v) &< d(v) + \max\{c|r(u_i)| + \delta(u_i) \ : \ u_i \rightarrow v \in E\} \\ &= d(v) + c|r(u_{i'})| + \delta(u_{i'}) \\ &= d(v) + c|r(v)| + \delta(u_{i'}) \\ &\leq d(v) + c|r(v)| + \max\{\delta(u_i) \ : \ u_i \rightarrow v \in E\} \end{aligned}$$

which implies that

$$\delta(v) < d(v) + \max\{\delta(u_i) \ : \ u_i \rightarrow v \in E\}.$$

But from step 2.3 we have $d(v) + \max\{\delta(u_i) \ : \ u_i \rightarrow v \in E\} = \delta(v)$, which is a contradiction.

Case 3: $r(u_{i'}) > r(v)$. In this case we have $|r(v)| \geq |r(u_{i'})| + 1$, which implies

$$\begin{aligned} c|r(v)| &\geq c|r(u_{i'})| + c \\ &\geq c|r(u_{i'})| + \delta(u_{i'}) \\ &= \max\{c|r(u_i)| + \delta(u_i) \ : \ u_i \rightarrow v \in E\}. \end{aligned}$$

Since $\delta(v) \geq d(v)$ from Lemma 2.8, the last inequality implies

$$c|r(v)| + \delta(v) \geq d(v) + \max\{c|r(u_i)| + \delta(u_i) \ : \ u_i \rightarrow v \in E\},$$

which contradicts inequality (2.6). □

Now, using Lemma 2.9 we can prove that the values $r(v)$ given by Algorithm MLP are the lengths of the shortest-paths in the constraint graph G_c from the input interface v_I.

Lemma 2.10 *Let $l(v)$ be the length of the shortest path in G_c from v_I to v. Then Algorithm MLP sets $r(v) = l(v)$.*

Proof: Assume for the sake of contradiction that the length $l(v)$ of the actual shortest path p in G_c satisfies $l(v) < r(v) \leq 0$. This inequality implies that p traverses *at least* one -1 edge more than the shortest path indicated by Algorithm MLP. Consequently, there exists a path p from v_I to v in G with propagation delay $d(p)$ such that $d(p) \geq c|l(v)| + 1$, since $d(v) \geq 1$ for every vertex $v \in V$. From Lemma 2.9, however, the maximum possible delay $\Delta(v)$ from v_I to v along any path in G satisfies

$$\begin{aligned}
\Delta(v) &\leq c|r(v)| + \delta(v) \\
&\leq c|r(v)| + c \\
&\leq c(|l(v)| - 1) + c \\
&= c|l(v)| \\
&< c|l(v)| + 1,
\end{aligned}$$

implying $\Delta(v) < d(p)$, which is a contradiction. □

Combining lemmata 2.4, 2.6, 2.7 and 2.10, we obtain the following theorem.

Theorem 2.11 *Algorithm MLP correctly solves Problem MLP.* □

This theorem completes the correctness proof of Algorithm MLP.

In summary, in this chapter we presented and proved the correctness of a greedy strategy for pipelining combinational circuitry. The clock-period of the pipelined circuit is guaranteed not to exceed a specified upper bound c and its latency is guaranteed to be minimal under the given clock-period constraint. The running time of the algorithm is directly proportional to the number of interconnections in the circuit and is optimal within a constant factor. The given procedure is used extensively as a subroutine of the algorithms in the following section.

Chapter 3

Minimum Clock-Period Characterization

In this chapter we give a concise characterization of the minimum feasible clock-period of a circuit in terms of the maximum *delay-to-register ratio* of the directed cycles in the circuit graph. This characterization leads to improved algorithms for various retiming problems.

The chapter is structured as follows. Section 3.1 introduces basic definitions that are used throughout the chapter. Section 3.2 gives an *exact* characterization of the minimum feasible clock-period for unit-delay circuits and Section 3.3 gives a range of D values for the minimum feasible clock-period of general circuits, where D is the maximum propagation delay of the circuit components. The previous ranges known for both cases had $\Theta(V^2)$ values.

The algorithmic implications of the minimum feasible clock-period characterizations are given in the three subsections of Section 3.4. Section 3.4.1 gives an $O(E)$ algorithm for minimum clock-period pipelining of unit-delay combinational circuitry. The running time of this algorithm is optimal within a constant. An $O(E \lg D)$ algorithm for minimum clock-period pipelining of general combinational circuitry is also presented in this section. Section 3.4.2 presents an $O(\min\{V^{1/2}E \lg(VD), VE\})$ algorithm for minimum clock-period retiming of unit-delay circuitry, and an $O(VE \lg D)$ algorithm for minimum clock-period retiming of general circuitry. Finally, Section 3.4.3 gives an $O(\min\{V^{1/2}E \lg(VW) \lg(VD), VE \lg(VD)\})$ algorithm for determining a retiming of a general circuit such that the clock-period is approximately minimized.

3.1 Preliminaries

In this section we give some basic definitions that we will use throughout the rest of the chapter.

Let $G = (V, E, d, w)$ be a circuit graph. We denote by D the maximum propagation delay of the circuit components:

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

We define the *delay-to-register* ratio $R(C)$ of a cycle $C = v_0 \stackrel{e_0}{\rightarrow} v_1 \stackrel{e_1}{\rightarrow} \ldots \stackrel{e_{k-2}}{\rightarrow} v_{k-1} \stackrel{e_{k-1}}{\rightarrow} v_0$ 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 $C \in G$.

A clock-period c is called *feasible* for the circuit G if and only if there exists a retiming r of G such that $\Phi(G_r) \leq c$. Finally, we denote by $\Phi_{min}(G)$ the clock-period of the retimed circuit G_r with the smallest possible clock-period:

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

3.2 Minimum Period for Unit-Delay Circuits

In this section we relate the minimum clock-period $\Phi_{min}(G)$, that we can obtain by retiming a given unit-delay circuit $G = (V, E, 1, w)$, with the maximum delay-to-register ratio $R(C^*(G))$ of the cycles C in the circuit graph G. Specifically, we show that $\Phi_{min}(G) = \lceil R(C^*(G)) \rceil$.

The result presented in this section relies on a retiming theorem in [17], which gives a characterization of when a unit-delay circuit has a clock-period less than or equal to c. The theorem is phrased in terms of the graph $G - 1/c$, which is defined as $G - 1/c = (V, E, d, w')$ where $w'(e) = w(e) - 1/c$ for every edge $e \in E$. Thus, $G - 1/c$ is the graph obtained from G by subtracting $1/c$ from the weight of each edge in G.

Theorem 3.1 *Let $G = (V, E, 1, w)$ be a unit-delay synchronous circuit, and let c be any positive integer. Then there is a retiming r of G such that $\Phi(G_r) \leq c$ if and only if $G - 1/c$ contains no cycles having negative edge-weight.* □

We can use Theorem 3.1 to characterize the minimum clock period $\Phi_{min}(G)$ in terms of the maximum delay-to-register ratio $R(C^*(G))$.

Theorem 3.2 *Let $G = (V, E, 1, w)$ be a unit-delay synchronous circuit with maximum delay-to-register ratio $R(C^*(G))$. Let $\Phi_{min}(G)$ denote the minimum clock-period that can be obtained by retiming G. Then*

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

Proof: According to Theorem 3.1, a clock-period c is feasible if and only if $G - 1/c$ has no negative-weight cycles. Thus, for every cycle $C = v_0 \stackrel{e_0}{\rightarrow} v_1 \stackrel{e_1}{\rightarrow} \ldots \stackrel{e_{k-2}}{\rightarrow} v_{k-1} \stackrel{e_{k-1}}{\rightarrow} v_0$ in G and any feasible clock-period c we have:

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

Equivalently:

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

The right hand side of the last inequality equals $R(C)$, by definition, and since this inequality holds for every cycle $C \in G$ it must also hold for $C^*(G)$. Thus $c \geq R(C^*(G))$. Now, the integrality of c implies that $c \geq \lceil R(C^*(G)) \rceil$, and since $c \geq \Phi_{min}(G) \geq \lceil R(C^*(G)) \rceil$ for every feasible period c, we have that $\Phi_{min}(G) = \lceil R(C^*(G)) \rceil$. □

3.3 Minimum Period for General Circuits

In this section we relate the minimum clock-period $\Phi_{min}(G)$, that we can obtain by retiming a given general circuit $G = (V, E, d, w)$, with the delay-to-register ratios of the cycles in the circuit graph G and the propagation delays of the circuit components. Specifically, we show that

$$\lceil R(C^*(G)) \rceil \leq \Phi_{min}(G) \leq \lceil R(C^*(G)) \rceil + D,$$

where D denotes the maximum propagation delay of the elements in the circuit, and $C^*(G)$ denotes the cycle in G with maximum delay-to-register ratio $R(C^*(G))$. Observe that both the lower and the upper bound are independent of the size of the circuit.

There is no counterpart of Theorem 3.1 known for general circuits. This is the reason why we cannot obtain an exact characterization of $\Phi_{min}(G)$ for general circuits in a manner similar to that of the previous section for unit-delay circuits. However, we are still able to give tight bounds for $\Phi_{min}(G)$, which are independent of the size of the circuit.

The next theorem gives a necessary condition for a circuit to have a clock-period less than or equal to c, and will be used to derive a lower bound for $\Phi_{min}(G)$. The theorem is phrased in terms of the graph $G - d/c$, which is defined as $G - d/c = (V, E, d, w')$, where $w'(e) = w(e) - d(v)/c$ for every edge $u \xrightarrow{e} v \in E$.

Theorem 3.3 *Let $G = (V, E, d, w)$ be a synchronous circuit, and let c be any positive integer. If there is a retiming r of G such that $\Phi(G_r) \leq c$ then $G - d/c$ contains no cycles having negative edge-weight.*

Proof: Assume there exists a retiming r of G such that $\Phi(G_r) \leq c$. Consider any cycle $C \in G_r$. For every register-free path $p = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{k-2}} v_{k-1} \xrightarrow{e_{k-1}} v_k$ in the cycle we have $\sum_{i=0}^{k} d(v_i) \leq c$. Let $w_r(C) = \sum_{e_i \in C} w_r(e_i)$ be the number of registers in C. Then, by adding the contributions from the $w_r(C)$ register-free paths in C, we get

$$\sum_{v_i \in C} d(v_i) \;\leq\; c \sum_{e_i \in C} w_r(e_i)$$

or, equivalently,

$$\sum_{e_i \in C} w_r(e_i) \;-\; \sum_{v_i \in C} d(v_i)/c \geq 0.$$

Now, recall that $w_r(e) = w(e) + r(u) - r(v)$ for every edge $u \xrightarrow{e} v \in C$. Consequently, the sum over the edges in C telescopes, yielding

$$\sum_{e_i \in C} w(e_i) - \sum_{v_i \in C} d(v_i)/c \geq 0.$$

Since this statement is true for every cycle $C \in G$, we conclude that $G - d/c$ contains no cycles with negative edge-weight. □

As a direct consequence of Theorem 3.3, we have the following lower bound on the minimum feasible clock-period of a general circuit:

Corollary 3.4 *Let $G = (V, E, d, w)$ 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*

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

Proof: For *any* feasible clock-period c, Theorem 3.3 implies

$$\sum_{u \xrightarrow{e} v \in C} (w(e) - d(v)/c) \geq 0$$

for every cycle $C \in G$. Equivalently, $c \geq R(C)$ for every cycle $C \in G$, which yields $c \geq R(C^*(G))$ for $C = C^*(G)$. Since this lower bound holds for every feasible clock-period, we have $R(C^*(G)) \leq \Phi_{min}(G)$. Given that the propagation delays of the circuit components are integers we infer that $\Phi_{min}(G)$ must be an integer as well. Therefore $\lceil R(C^*(G)) \rceil \leq \Phi_{min}(G)$. □

Observe that the converse of Theorem 3.3 is *not* true. Specifically, given a circuit $G = (V, E, d, w)$, if $G - d/c$ has no negative weight cycles it does not follow that there exists a retiming r of the circuit such that $\Phi(G_r) \leq c$. The validity of this statement can be demonstrated most easily with the help of an example. Consider the circuit of Figure 3.1, which is configured as a ring with three registers and four computational elements. It is impossible to get a retiming with clock-period $c = 3$, even though $R(C^*(G)) = 3$, since there is only one register available to be placed among the three elements of delay 2.

Even though the converse of Theorem 3.3 is not true, we can still find an upper bound for $\Phi_{min}(G)$ in terms of the maximum delay-to-register ratio $R(C^*(G))$ in the circuit and the maximum propagation delay D of the circuit components.

Lemma 3.5 *Let $G = (V, E, d, w)$ 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*

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

Figure 3.1: A synchronous circuit G with three registers and four computational elements. The propagation delay of each element is indicated in the vertex which represents it. The circuit cannot be retimed to have period $c = 3$, even though G-d/c has no negative weight cycles.

Proof: We will prove that $\Phi_{min}(G) \leq \lceil R(C^*(G)) \rceil + D$ by showing that G can be retimed to have clock-period $c = \lceil R(C^*(G)) \rceil + D$.

According to the mathematical programming formulation of retiming, which was given in Theorem 2.1, the circuit G can be retimed to achieve period c if we can find a set of values $r(v)$ such that

$$r(v) - r(u) \leq w(u \rightarrow v) \tag{3.1}$$

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

$$r(v) - r(u) \leq W(u, v) - 1 \tag{3.2}$$

for all vertices u, v such that $D(u, v) > c$. Let

$$E_W = \{u \rightarrow v \ : \ u, v \in V, \ r(v) - r(u) \text{ is constrained by (3.2)}\}.$$

The constraint sets (3.1) and (3.2) induce the constraint graph $G_c = (V, E \cup E_W, w_c)$, where

$$w_c(u \rightarrow v) = \begin{cases} w(u \rightarrow v) & u \rightarrow v \in E, \\ W(u, v) - 1 & u \rightarrow v \in E_W. \end{cases}$$

According to Lemma 2.2 and Lemma 2.3, the circuit G can be retimed to achieve clock-period $\Phi(G_r) \leq c$ exactly when G_c has no negative weight cycles. Let us assume for the sake of contradiction that G_c *does* have a negative weight cycle $C^- \in G_c$, which consists of two sets of edges E_1' and E_2', with $E_1' \subseteq E$, $E_2' \subseteq E_W$, $|E_1'| = n_1$ and $|E_2'| = n_2$. Since the edge-weights are integral we have

$$\sum_{e \in E_1'} w_c(e) + \sum_{e \in E_2'} w_c(e) \leq -1. \tag{3.3}$$

Let

$$E_2'' = \{v_1 \to v_2 \ : \ v_1 \to v_2 \in E,\ v_1 \to v_2 \in u \overset{p_{cr}}{\rightsquigarrow} v,\ u \to v \in E_2'\},$$

where $u \overset{p_{cr}}{\rightsquigarrow} v$ denotes the critical path in G from u to v. Then, according to inequality (3.3), we have:

$$\begin{aligned}
\sum_{e\in E_1'} w(e) + \sum_{e\in E_2''} w(e) &= \sum_{e\in E_1'} w(e) + \sum_{e\in E_2''} w(e) - n_2 + n_2 \\
&= \sum_{e\in E_1'} w(e) + \sum_{e\in E_2'} w_c(e) + n_2 \\
&= \sum_{e\in E_1'} w_c(e) + \sum_{e\in E_2'} w_c(e) + n_2 \\
&\leq n_2 - 1.
\end{aligned}$$

Now, for the delay-to-register ratio of the cycle which consists of the edges $E_1' \cup E_2''$ in G we have :

$$\begin{aligned}
\frac{\sum_{u\overset{e}{\to}v\in E_1'} d(u) + \sum_{u\overset{e}{\to}v\in E_2''} d(u)}{\sum_{u\overset{e}{\to}v\in E_1'} w(e) + \sum_{u\overset{e}{\to}v\in E_2''} w(e)} &\geq \frac{\sum_{u\overset{e}{\to}v\in E_1'} d(u) + \sum_{u\overset{e}{\to}v\in E_2''} d(u)}{n_2 - 1} \\
&\geq \frac{\sum_{u\overset{e}{\to}v\in E_2''} d(u)}{n_2 - 1} \\
&> \frac{n_2(c - D)}{n_2 - 1} \\
&= \frac{n_2}{n_2 - 1} \lceil R(C^*(G))\rceil \\
&\geq \frac{n_2}{n_2 - 1} R(C^*(G)).
\end{aligned}$$

Since $n_2/(n_2 - 1) > 1$, 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 weight cycles and $c = \lceil R(C^*(G))\rceil + D$ is a feasible clock-period. Consequently $\Phi_{min}(G) \leq \lceil R(C^*(G))\rceil + D$. □

Corollary 3.4 and Lemma 3.5 imply the following.

Theorem 3.6 *Let $G = (V, E, d, w)$ 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*

$$\lceil R(C^*(G))\rceil \leq \Phi_{min}(G) \leq \lceil R(C^*(G))\rceil + D.$$ □

Circuit Type	Transformation	Running Time
UD Combinational	Min Clock-Period	$O(E)$
Combinational	Min Clock-Period	$O(E \lg D)$
UD Sequential	Min Clock-Period	$O\left(\min\left\{\begin{array}{l} V^{1/2}E\lg(VW) \\ VE \end{array}\right\}\right)$
Sequential	Min Clock-Period	$O(VE \lg D)$
Sequential	Approx Min Clock-Period	$O\left(\min\left\{\begin{array}{l} V^{1/2}E\lg(VW)\lg(VD) \\ VE\lg(VD) \end{array}\right\}\right)$

Figure 3.2: Summary of problems and running times of corresponding algorithms. The initials UD denote unit-delay circuitry.

3.4 Algorithmic Implications

In this section we study the algorithmic implications of the minimum clock-period characterization for a variety of retiming problems. We use the ideas of the previous sections to develop fast algorithms for minimum clock-period pipelining of combinational circuitry. We show how to obtain improved running times for clock-period minimization of sequential circuits, using known graph-theoretic algorithms. Finally, we give a faster algorithm for *approximate* clock-period minimization of general sequential circuits. The problems listed in this section along with the running times of the corresponding algorithms are illustrated in Figure 3.2.

3.4.1 Minimum clock-period pipelining

We use the ideas of the previous sections to develop fast algorithms for minimum clock-period pipelining of combinational circuitry. Specifically, we give an $O(E)$ optimal algorithm for minimum clock-period pipelining of unit-delay combinational circuitry and an $O(E \lg D)$ algorithm for minimum clock-period pipelining of general combinational circuitry.

Let us consider unit-delay circuitry first. The problem of minimum clock-period pipelining is defined as follows: *Given a unit-delay combinational circuit $G = (V, E, 1, 0)$ and a positive integer l, determine a retiming r such that G_r is a pipelined combinational circuit with latency no greater than l and with minimum clock-period.* The following lemma characterizes the minimum feasible clock-period in terms of the longest propagation delay Δ of a path in the circuit and the latency l.

Lemma 3.7 *Let $G = (V, E, 1, 0)$ be a unit-delay combinational circuit with input interface v_I and output interface v_O. Let Δ be the number of vertices in the longest path $p_\Delta = v_I \rightsquigarrow v_O$ in G, and let l be a positive integer. Then the minimum clock-period $\Phi_{min}(G)$ for any pipelined version of G with latency l is*

$$\Phi_{min}(G) = \left\lceil \frac{\Delta}{l+1} \right\rceil .$$

Proof: Any retiming r of the circuit that gives a pipelined version of the circuit with latency l satisfies constraints (2.1) and (2.2) as well as a latency constraint. Specifically, it satisfies

$$r(v) - r(u) \leq 0$$

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

$$r(v) - r(u) \leq -1$$

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

$$r(v_I) - r(v_O) \leq l.$$

This set of inequalities induces the constraint graph $G_c = (V_c, E_c, w_c)$ and according to Lemma 2.2 and Lemma 2.3 it is feasible if and only if there exists no negative-weight cycle in G_c. We shall use this statement to show that $\Phi_{min}(G) = \lceil \Delta/(l+1) \rceil$.

First we show that $\lceil \Delta/(l+1) \rceil$ is a lower bound for $\Phi_{min}(G)$. Let r be a feasible retiming of the circuit with latency l and clock-period c. Every path in G_r from v_I to v_O has $l+1$ register-free parts. Consider the longest such path p_Δ with delay Δ. Adding up all the contributions yields $\Delta \leq c(l+1)$, which implies $c \geq \Delta/(l+1)$. Therefore, $\Phi_{min}(G) \geq \Delta/(l+1)$, or $\Phi_{min}(G) \geq \lceil \Delta/(l+1) \rceil$, since $\Phi_{min}(G)$ must be an integer.

Now, we prove that $\lceil \Delta/(l+1) \rceil$, the lower bound of $\Phi_{min}(G)$, is a feasible clock-period, thus establishing the desired equality. In order to prove feasibility of the lower bound it suffices to show that G_c has no negative-weight cycles for $c = \lceil \Delta/(l+1) \rceil$. Equivalently, since the maximum number of -1 edges in any path is

$$\left\lfloor \frac{\Delta - 1}{\lceil \Delta/(l+1) \rceil} \right\rfloor ,$$

it suffices to show that

$$l - \left\lfloor \frac{\Delta - 1}{\lceil \Delta/(l+1) \rceil} \right\rfloor \geq 0.$$

We have

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

since $(l+1)/\Delta \leq 1$. Therefore, G_c has no negative-weight cycles, which implies that $\lceil \Delta/(l+1) \rceil$ is a feasible clock-period in addition to being a lower bound for $\Phi_{min}(G)$. Therefore $\Phi_{min}(G) = \lceil \Delta/(l+1) \rceil$. □

Now, we give the following algorithm for the problem of minimum clock-period pipelining. The correctness of the algorithm follows from Theorem 2.11 and Lemma 3.7.

Algorithm UDMPP *(Unit-Delay Minimum Period Pipelining)* Given a unit-delay combinational circuit $G = (V, E, 1, 0)$ with input interface v_I and output interface v_O, and a positive integer l, determine a retiming r such that G_r is a pipelined combinational circuit with latency l and minimum clock-period.

1. Determine the number of vertices Δ in the longest path p_Δ in G from v_I to v_O.
2. Run Algorithm MLP on G with clock-period $\lceil \Delta/(l+1) \rceil$. □

The algorithm terminates in $O(E)$ steps, since step 1 is a depth-first-search in the graph and Algorithm MLP runs in $O(E)$ steps.

Now, we consider the case of general combinational circuitry. The problem of minimum clock-period pipelining is defined in an analogous way: *Given a unit-delay combinational circuit $G = (V, E, d, 0)$ and a positive integer l, determine a retiming r such that G_r is a pipelined combinational circuit with latency no greater than l and minimum clock-period.* The following lemma characterizes the minimum feasible clock-period in terms of the delay Δ of the longest path in the circuit, the latency l, and the longest component delay D.

Lemma 3.8 *Let $G = (V, E, d, 0)$ be a combinational circuit with input interface v_I and output interface v_O. Let Δ be the delay of the path $p_\Delta = v_I \rightsquigarrow v_O$ 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:

$$\left\lceil \frac{\Delta}{l+1} \right\rceil \leq \Phi_{min}(G) \leq \left\lceil \frac{\Delta}{l+1} \right\rceil + D,$$

where D is the longest component delay in the circuit.

Proof: Any retiming r of the circuit that gives a pipelined version of the circuit with latency l satisfies constraints (2.3) and (2.4) as well as a latency constraint. Specifically, it satisfies

$$r(v) - r(u) \leq 0$$

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

$$r(v) - r(u) \leq -1$$

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

$$r(v_I) - r(v_O) \leq l.$$

First, we derive the lower bound of the inequality. Consider the constraint graph G_c induced by the above constraints. Let r be a feasible retiming of the circuit with latency l and clock-period c. Every path in G_r from v_I to v_O has $l+1$ register-free parts. Adding up all the delays of the register-free parts along the longest such path p_Δ yields $\Delta \leq c(l+1)$, which implies $c \geq \Delta/(l+1)$. Therefore, $\Phi_{min}(G) \geq \Delta/(l+1)$, or $\Phi_{min}(G) \geq \lceil \Delta/(l+1) \rceil$, since $\Phi_{min}(G)$ must be an integer as a consequence of the fact that $d(v) \in \mathbf{Z}$ for every vertex $v \in V$.

Now, we establish the upper bound of the inequality by proving that $\lceil \Delta/(l+1) \rceil + D$ is a feasible clock-period. In order to achieve this it suffices to show that G_c has no negative-weight cycles for $c = \lceil \Delta/(l+1) \rceil$. The maximum number of -1 edges in any path is

$$\left\lfloor \frac{\Delta - 1}{(\lceil \Delta/(l+1) \rceil + D) - D} \right\rfloor = \left\lfloor \frac{\Delta - 1}{\lceil \Delta/(l+1) \rceil} \right\rfloor.$$

But we have already shown in Lemma 3.7 that

$$l \geq \left\lfloor \frac{\Delta - 1}{\lceil \Delta/(l+1) \rceil} \right\rfloor.$$

Hence G_c has no negative-weight cycles. □

We give an $O(E \lg D)$ algorithm for minimum period pipelining of combinational circuitry. Its correctness follows from Theorem 2.11 and Lemma 3.8.

Algorithm MPP *(Minimum Period Pipelining)* Given a combinational circuit $G = (V, E, d, 0)$ with input interface v_I and output interface v_O, and a positive integer l, determine a retiming r such that G_r is a pipelined combinational circuit with latency l and minimum clock-period.

1. Determine the delay Δ of the longest path p_Δ in G from v_I to v_O.

2. Binary search among the D possible values of $\Phi_{min}(G)$ applying Algorithm MLP on G. □

Step 1 is a depth-first search in G and Step 2 performs $O(\lg D)$ applications of Algorithm MLP. Therefore, Algorithm MPP terminates in $O(E \lg D)$ steps.

3.4.2 Minimum clock-period retiming

In this section we study the implications of the minimum period characterization for retiming of sequential circuitry. Specifically, we consider the problem: Given a sequential circuit $G = (V, E, d, w)$, determine a retiming r such that $\Phi(G_r)$ is minimum.

We consider unit-delay circuitry first. In order to compute the minimum feasible period of the circuit we can use Karp's $O(VE)$ algorithm for finding minimum mean cycles in a graph [11]. Then, using Bellman-Ford's shortest-paths algorithm on $G - 1/c$ we can find a retiming r such that $\Phi(G_r)$ is minimum, according to Theorem 3.1. The overall running time is $O(VE)$, which is an improvement over the best previously known strongly polynomial algorithm by a $\lg V$ factor, since it eliminates the need for binary search. Using scaling we obtain an $O(V^{1/2}E \lg(VW))$ algorithm for the same problem, where W is the maximum register count among the edges. This algorithm utilizes Orlin-Ahuja's $O(V^{1/2}E \lg(VW))$ algorithm for minimum mean cycles [20], followed by Gabow-Tarjan's $O(V^{1/2}E \lg(VW))$ scaling algorithm for shortest-paths [8].

For general circuits we obtain an $O(VE \lg D)$ running time by binary searching with the general retiming algorithm described in [17] the range of the D possible values for the clock -period of the circuit. An interesting open question is whether we can obtain a better running time by using scaling.

3.4.3 Approximately minimum clock-period retiming

In this section we give an algorithm for determining a retiming of a general circuit such that the clock-period is approximately minimized. Specifically, we consider the following problem: Given a sequential circuit $G = (V, E, d, w)$ determine a retiming r such that $\Phi(G_r) \leq \Phi_{min}(G) + D \leq 2\Phi_{min}(G)$, where D is the maximum propagation delay of the circuit components. We show that using scaling this problem can be solved faster than minimum clock-period retiming by a factor of $V^{1/2}/(\lg(VW)\lg(VD))$.

The algorithm for approximately minimum clock-period retiming is based on the lemma that follows. We denote by $G - d/c$ the graph with vertex set V, edge set E and edge weight $w(e) - d(v)/c$ on each edge $u \xrightarrow{e} v \in E$.

Lemma 3.9 *Let $G = (V, E, d, w)$ 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. Moreover, let $n = \lceil R(C^*(G)) \rceil$, and let $l(v)$ be the solutions of a single-source shortest-paths problem on $G - d/n$. Then, the assignment $r(v) = \lceil l(v) \rceil$ for each vertex $v \in V$ is a retiming of G such that*

$$\Phi(G_r) \leq \Phi_{min}(G) + D. \tag{3.4}$$

Proof: Note that the shortest-paths lengths $l(v)$ are well-defined, since $G - d/\lceil R(C^*(G)) \rceil$ has no negative-weight cycles. In order to prove that $r(v) = \lceil l(v) \rceil$ is a legal retiming with clock-period $\Phi(G_r) \leq \Phi_{min}(G) + D$, we show that it satisfies constraints (2.1) and (2.2) with $c = \lceil R(C^*(G)) \rceil + D$. Then, we conclude inequality (3.4) directly from Corollary 3.4.

First, we prove that $r(v) = \lceil l(v) \rceil$ for each $v \in V$ satisfies constraints (2.1). For every edge $u \xrightarrow{e} v$ we have :

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

since $\lceil x - y \rceil \leq \lceil x \rceil - \lceil y \rceil$ for every real x, y, and $w(e)$ is an integer. Therefore, $\lceil l(v) \rceil$ satisfies (2.1).

Now, we prove that the assignment $r(v) = \lceil l(v) \rceil$ for each $v \in V$ satisfies constraints (2.2). Consider any path $p = u_0 \stackrel{e_0}{\rightarrow} u_1 \stackrel{e_1}{\rightarrow} \ldots \stackrel{e_{k-2}}{\rightarrow} u_{k-1} \stackrel{e_{k-1}}{\rightarrow} u_k$ with delay $\sum_{i=0}^{k} d(u_i) > c$. For this path we have:

$$\begin{aligned} l(u_k) - l(u_0) &\leq \left(\sum_{i=0}^{k-1} w(e_i)\right) - \left(\sum_{i=1}^{k} \frac{d(u_i)}{n}\right) \\ &= \left(\sum_{i=0}^{k-1} w(e_i)\right) - \left(\sum_{i=0}^{k} \frac{d(u_i)}{n}\right) + \frac{d(u_0)}{n} \\ &\leq \left(\sum_{i=0}^{k-1} w(e_i)\right) - \frac{(c+1) - d(u_0)}{n} \\ &\leq \left(\sum_{i=0}^{k-1} w(e_i)\right) - \frac{\lceil R(C^*(G)) \rceil + D + 1 - d(u_0)}{\lceil R(C^*(G)) \rceil} \\ &\leq \left(\sum_{i=0}^{k-1} w(e_i)\right) - 1, \end{aligned}$$

since $D + 1 - d(u_0) \geq 1$. Therefore,

$$\begin{aligned} \lceil l(u_k) \rceil - \lceil l(u_0) \rceil &\leq \lceil l(u_k) - l(u_0) \rceil \\ &= \left\lceil \left(\sum_{i=0}^{k-1} w(e_i)\right) - 1 \right\rceil \\ &= \left(\sum_{i=0}^{k-1} w(e_i)\right) - 1, \end{aligned}$$

which implies that $\lceil l(v) \rceil$ satisfies constraints (2.2).

Therefore, the assignment of lead $\lceil l(v) \rceil$ to each vertex $v \in V$ yields a legal retiming with clock-period $\Phi(G_r) \leq \lceil R(C^*(G)) \rceil + D$. From Corollary 3.4 it follows that $\Phi(G_r) \leq \Phi_{min}(G) + D$. □

The algorithm for approximately minimum clock-period is based on Lemma 3.9 and it proceeds as follows:

Algorithm ApproxCPM *(Approximate Clock-Period Minimization)* Given a circuit $G = (V, E, d, w)$ with maximum delay-to-register ratio $R(C^*(G))$, minimum feasible clock-period $\Phi_{min}(G)$ and maximum component delay D, determine a retiming r of the circuit, such that $\Phi(G_r) \leq \lceil R(C^*(G)) \rceil + D \leq \Phi_{min}(G) + D$.

1. Compute $n = \lceil R(C^*(G)) \rceil$ by binary searching in the range $[1, \ldots, VD]$.

2. Let $l(v)$ be the lengths of the shortest-paths in $G - d/n$ from some source vertex $s \in V$.

3. Set $r(v) = \lceil l(v) \rceil$, for every vertex $v \in V$. □

Step 1 of the algorithm binary searches for the smallest integer n, which exceeds the maximum delay-to-register ratio $R(C^*(G))$. This ratio is positive and cannot exceed VD, since the maximum propagation delay of the circuit components is D and since the longest simple path in the circuit has at most V vertices. Each one of the $O(\lg(VD))$ iterations of the binary search checks for negative-weight cycles in $G - d/n$. The value of $\lceil R(C^*(G)) \rceil$ equals the smallest integer n in the range that induces no negative-weight cycles in $G - d/n$ [12]. Negative-weight cycles can be detected in $O(VE)$ steps, using Bellman-Ford's algorithm [12], or in $O(V^{1/2}E\lg(VW))$ steps, where W is the maximum register count along any connection in the circuit, using Gabow-Tarjan's shortest-paths algorithm [8]. Step 2 requires a single-source shortest-paths algorithm and Step 3 terminates in $O(V)$ steps. Step 1 of the algorithm dominates the total running time yielding an $O(\min\{V^{1/2}E\lg(VW)\lg(VD), VE\lg(VD)\})$ running time overall.

In summary, in this chapter we presented a novel and concise characterization of the minimum clock-period, that can be obtained by retiming a synchronous circuit G, in terms of the maximum delay-to-register ratio of the cycles in the circuit graph and the maximum propagation delay of the circuit components. Based on the ideas behind this characterization, we gave an optimal algorithm for optimal pipelining of unit-delay combinational circuitry and an efficient algorithm for optimal pipelining of general combinational circuitry. We also gave improved algorithms for minimum clock-period retiming of unit-delay and general circuitry. Finally, we described a technique which yields a retiming with clock-period that does not exceed the minimum by more than a factor of 2 and is asymptotically faster than the known algorithms for minimum clock-period retiming.

Chapter 4

The Closed Semiring Structure of Retiming

This chapter investigates group-theoretic properties of retiming on unit-delay circuitry. Specifically, we show that retiming of unit-delay circuitry can be described in terms of a closed semiring. The three sections of this chapter are organized as follows. In Section 4.1 we review the notion of a closed semiring. In Section 4.2 we construct the closed semiring, that captures the structure of unit-delay circuitry retiming. Finally, in Section 4.3 we utilize the additive and multiplicative operations of the closed semiring in order to design an $O(VE)$ algorithm for unit-delay circuitry retiming.

4.1 Preliminaries

In this section we review the notion of a closed semiring. A more detailed exposition can be found in [4].

Let S be a set of elements, and let $\oplus$ and $\otimes$ be binary operations on S. A system $(S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ is a closed semiring if it satisfies the following properties:

1. $(S, \oplus, \mathbf{0})$ is a *monoid*:
   - S is *closed* under $\oplus$: $a \oplus b \in S$ for all $a, b \in S$.
   - $\oplus$ is *associative*: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ for all $a, b, c \in S$.
   - $\mathbf{0}$ is an *identity* element for $\oplus$: $a \oplus \mathbf{0} = a$ for all $a \in S$.
2. $(S, \otimes, \mathbf{1})$ is a *monoid*:

- S is *closed* under $\otimes$: $a \otimes b \in S$ for all $a, b \in S$.
- $\otimes$ is *associative*: $(a \otimes b) \otimes c = a \otimes (b \otimes c)$ for all $a, b, c \in S$.
- $\mathbf{1}$ is an *identity* element for $\otimes$: $a \otimes \mathbf{1} = a$ for all $a \in S$.

3. $\mathbf{0}$ is an *annihilator*: $\mathbf{0} \otimes a = \mathbf{0}$ for all $a \in S$.
4. $\oplus$ is *commutative*: $a \oplus b = b \oplus a$ for all $a, b \in S$.
5. $\oplus$ is *idempotent*: $a \oplus a = a$ for all $a \in S$.
6. $\otimes$ *distributes* over $\oplus$: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ for all $a, b, c \in S$.
7. For any infinite, countable sequence $a_1, \ldots, a_i, \ldots$ the sum $a_1 \oplus \ldots \oplus a_i \oplus \ldots$ exists and is unique. Associativity, commutativity, idempotence applies to finite as well as infinite sums.
8. $\otimes$ distributes over countably infinite sums.

4.2 The Closed Semiring Construction

In this section we present the closed semiring construction which captures unit-delay retiming on the original circuit graph.

We define the set S as follows:

$$S = \{ (r, d) \ : \ r \in \mathbf{N} \ , \ d \in \{0, 1, \ldots, c-1\} \} \cup \{\infty, \infty\}.$$

We denote the additive operation by MIN and define it as follows:

$$(r_1, d_1) \ \mathrm{MIN} \ (r_2, d_2) = (\min\{r_1, r_2\}, I(d_1, d_2 \ ; \ r_1, r_2 \ )),$$

where

$$\min\{r_1, r_2\} = \begin{cases} r_1 & \text{if } r_1 \leq r_2, \\ r_2 & \text{if } r_1 > r_2; \end{cases}$$

and

$$I(d_1, d_2 \ ; \ r_1, r_2 \ ) = \begin{cases} d_1 & \text{if } r_1 < r_2, \\ \max\{d_1, d_2\} & \text{if } r_1 = r_2, \\ d_2 & \text{if } r_1 > r_2. \end{cases}$$

We denote the multiplicative operation by $\odot$ and define it as follows:

$$(r_1, d_1) \odot (r_2, d_2) = (r_1 + r_2 - (d_1 + d_2) \operatorname{div} c, d_1 +_c d_2),$$

where

$$(d_1 + d_2) \operatorname{div} c = \left\lfloor \frac{d_1 + d_2}{c} \right\rfloor ;$$

$$d_1 +_c d_2 = \begin{cases} (d_1 + d_2) \bmod c & \text{if } d_1 \text{ and } d_2 \text{ are finite,} \\ \infty & \text{if } d_1 \text{ or } d_2 \text{ is infinite;} \end{cases}$$

and $+$ is the ordinary addition between integers.

The identity element for the additive operation is $\mathbf{0} \stackrel{\text{def}}{=} (\infty, \infty)$.

The identity element for the multiplicative operation is $\mathbf{1} \stackrel{\text{def}}{=} (0, 0)$.

Theorem 4.1 *The system* $(S, MIN, \odot, \mathbf{0}, \mathbf{1})$ *is a closed semiring.*

Proof: We prove the theorem by showing that the system satisfies all the properties of a closed semiring.

1. $(S, \text{MIN}, \mathbf{0})$ is a monoid. The following properties hold:

   - Closedness under MIN. Obvious.

   - Associativity of $\odot$. We must show that

   $$((r_1, d_1) \text{ MIN } (r_2, d_2)) \text{ MIN } (r_3, d_3) = (r_1, d_1) \text{ MIN } ((r_2, d_2) \text{ MIN } (r_3, d_3)). \tag{4.1}$$

   The left hand side of equation (4.1) can be rewritten as

   $$(\min\{\{r_1, r_2\}, r_3\},\ I(\ I(d_1, d_2\ ;\ r_1, r_2), d_3\ ;\ \min\{r_1, r_2\}, r_3)\ ).$$

   The right hand side of equation (4.1) can be rewritten as

   $$(\min\{r_1, \{r_2, r_3\}\},\ I(\ d_1,\ I(d_2, d_3\ ;\ r_2, r_3)\ ;\ r_1, \min\{r_2, r_3\})\ ).$$

   Since $\min\{\{r_1, r_2\}, r_3\} = \min\{r_1, \{r_2, r_3\}\}$, the first coordinates of the two sides are clearly equal. For associativity to hold, it remains to show that

   $$I(I(d_1, d_2\ ;\ r_1, r_2), d_3\ ;\ \min\{r_1, r_2\}, r_3) = I(d_1, I(d_2, d_3\ ;\ r_2, r_3)\ ;\ r_1, \min\{r_2, r_3\}). \tag{4.2}$$

   Applying the definitions of the operations to both sides of equation (4.2) we obtain the same expression:

   $$\begin{cases} d_i & \text{if } r_i < r_j, r_k, \\ \max\{d_i, d_j\} & \text{if } r_i = r_j < r_k, \\ \max\{d_1, d_2, d_3\} & \text{if } r_1 = r_2 = r_3, \end{cases}$$

   for distinct $i, j, k \in \{1, 2, 3\}$. Therefore, MIN is associative.

- **0** is an identity element. We have

$$\begin{aligned}(r,d)\ \text{MIN}\ (\infty,\infty) &= (\infty,\infty)\ \text{MIN}\ (r,d)\\ &= (\min\{r,\infty\}, I(d,\infty\,;\ r,\infty))\\ &= (r,d).\end{aligned}$$

2. $(S, \odot, \mathbf{1})$ is a monoid. The following properties hold:

- Closedness under $\odot$. Obvious.

- Associativity of $\odot$. We must show that

$$((a,b)\odot(r,d))\odot(e,f) = (a,b)\odot((r,d)\odot(e,f)). \tag{4.3}$$

If one of the pairs equals (∞,∞) the relation holds. In general, now, the left hand side of equation (4.3) equals

$$(a+r+e-\ (b+d)\operatorname{div} c\ -\ (f+(b+d)\bmod c)\operatorname{div} c\ ,(b+d+f)\bmod c)$$

and the right hand side of equation (4.3) equals

$$(a+r+e-\ (f+d)\operatorname{div} c\ -\ (b+(f+d)\bmod c)\operatorname{div} c\ ,(b+d+f)\bmod c).$$

In order to prove associativity it remains to show that

$$(b+d)\operatorname{div} c+(f+(b+d)\bmod c)\operatorname{div} c = (f+d)\operatorname{div} c+(b+(f+d)\bmod c)\operatorname{div} c. \tag{4.4}$$

The left hand side of equation (4.4) can be written as:

$$\begin{aligned}&(b+d)\ \text{div}\ c+(f+(b+d)\ \text{mod}\ c)\ \text{div}\ c\\ &= \left\lfloor\frac{b+d}{c}\right\rfloor+\left\lfloor\frac{f+(b+d)\ \text{mod}\ c}{c}\right\rfloor\\ &= \frac{b+d-(b+d)\ \text{mod}\ c}{c}+\left\lfloor\frac{f+(b+d)\ \text{mod}\ c}{c}\right\rfloor\\ &= \left\lfloor\frac{b+d-(b+d)\ \text{mod}\ c}{c}+\frac{f+(b+d)\ \text{mod}\ c}{c}\right\rfloor\\ &= \left\lfloor\frac{b+d+f}{c}\right\rfloor.\end{aligned}$$

Similarly, the right hand side of equation (4.4) can be shown to be equal to $\lfloor(b+d+f)/c\rfloor$. Therefore, $\odot$ is associative.

- **1** is an identity element. We have

$$\begin{aligned}(r,d)\odot(0,0) &= (0,0)\odot(r,d)\\ &= (r+0-\lfloor\frac{d}{c}\rfloor, d+_c 0)\\ &= (r,d).\end{aligned}$$

3. **0** is an annihilator: We have

$$\begin{aligned}(\infty,\infty)\odot(r,d) &= (r,d)\odot(\infty,\infty)\\ &= (\infty+r, \infty+_c d)\\ &= (\infty,\infty).\end{aligned}$$

4. MIN is commutative: We have

$$\begin{aligned}(a,b)\,\mathrm{MIN}\,(r,d) &= (\min\{a,r\}, I(b,d\,;\,a,r))\\ &= (\min\{r,a\}, I(d,b\,;\,r,a))\\ &= (r,d)\,\mathrm{MIN}\,(a,b)\end{aligned}$$

since it is clear that $I(b,d\,;\,a,r) = I(d,b\,;\,r,a)$.

5. MIN is idempotent: By a simple application of the definition

$$\begin{aligned}(r,d)\,\mathrm{MIN}\,(r,d) &= (\min\{r,r\}, I(d,d;\,r,r))\\ &= (r,d).\end{aligned}$$

6. $\odot$ distributes over MIN: We must show that

$$(a,b)\odot((r,d)\,\mathrm{MIN}\,(e,f)) = ((a,b)\odot(r,d))\,\mathrm{MIN}\,((a,b)\odot(e,f)). \tag{4.5}$$

For convenience, let $(L_1,L_2) = (a,b)\odot((r,d)\,\mathrm{MIN}\,(e,f))$ and $(R_1,R_2) = ((a,b)\odot(r,d))\,\mathrm{MIN}\,((a,b)\odot(e,f))$. Applying the definitions we have that

$$\begin{aligned}L_1 &= \min\{a+r, a+e\} - I(b+d, b+f\,;\,r,e)\ \mathrm{div}\ c,\\ L_2 &= (b+I(d,f\,;\,r,e))\ \mathrm{mod}\ c\,);\end{aligned}$$

and

$$\begin{aligned}R_1 &= \min\{a+r-(b+d)\ \mathrm{div}\ c,\ a+e-(b+f)\ \mathrm{div}\ c\},\\ R_2 &= I(\,(b+d)\ \mathrm{mod}\ c, (b+f)\ \mathrm{mod}\ c\,;\ a+r-(b+d)\ \mathrm{div}\ c,\ a+e-(b+f)\ \mathrm{div}\ c\,).\end{aligned}$$

First we show that $L_1 = R_1$.

$$\begin{aligned} L_1 &= \min\{a+r, a+e\} - I(b+d, b+f\ ;\ r, e) \text{ div } c \\ &= \min\{a+r, a+e\} - I((b+d) \text{ div } c,\ (b+f) \text{ div } c\ ;\ r, e) \\ &= \begin{cases} a+r-(b+d) \text{ div } c & \text{if } r < e, \\ a+r-\max\{(b+d) \text{ div } c,\ (b+f) \text{ div } c\} & \text{if } r = e, \\ a+e-(b+f) \text{ div } c & \text{if } r > e \end{cases} \\ &= \min\{a+r-(b+d) \text{ div } c,\ a+e-(b+f) \text{ div } c\} \\ &= R_1. \end{aligned}$$

Now, we show that $L_2 = R_2$.

$$\begin{aligned} L_2 &= (b + I(d, f\ ;\ r, e)) \bmod c \\ &= (I(b+d, b+f\ ;\ r, e)) \bmod c. \end{aligned}$$

Consider the following three possible combinations of values for $(b+d)$ div c and $(b+f)$ div c.

Case 1: $(b+d)$ div $c = (b+f)$ div c. Then:

$$\begin{aligned} L_2 &= I((b+d) \bmod c,\ (b+f) \bmod c\ ;\ r, e) \\ &= I((b+d) \bmod c,\ (b+f) \bmod c\ ;\ a+r-(b+d) \text{ div } c,\ a+e-(b+f) \text{ div } c) \\ &= R_2. \end{aligned}$$

Case 2: $(b+d)$ div $c = 1$ and $(b+f)$ div $c = 0$. In this case $f < d$ and $(b+d) \bmod c < (b+f) \bmod c$, since $(b+d) \bmod c = b+d-c < b+f = (b+f) \bmod c$ and $f, d < c$. Now, consider the two possible relations between r and e.

$r \leq e$: In this case $a+r-(b+d)$ div $c < a+e-(b+f)$ div c and consequently

$$\begin{aligned} L_2 &= R_2 \\ &= (b+d) \bmod c. \end{aligned}$$

$r > e$: In this case

$$\begin{aligned} L_2 &= (b+f) \bmod c \\ &= \max\{(b+d) \bmod c,\ (b+f) \bmod c\} \\ &= R_2. \end{aligned}$$

Case 3: $(b+d)$ div $c = 0$ and $(b+f)$ div $c = 1$. Symmetric to Case 2.

From Cases 1, 2, and 3, we conclude that $L_2 = R_2$. Therefore, equality (4.5) holds, since $(L_1, L_2) = (R_1, R_2)$, and consequently $\odot$ distributes over MIN.

7. MIN gives a unique result when operating on countably infinite sequences of arguments. Also, associativity, commutativity and idempotence applies to finite as well as infinite sums, as it can be readily seen from the definitions of the operations.

8. The multiplicative operation $\odot$ distributes over countably infinite sums, as we can easily demonstrate by a simple induction.

Items 1 through 8 demonstrate the correctness of the theorem. □

4.3 An Algorithm for Unit-Delay Circuitry Retiming

In this section we give a Bellman-Ford type algorithm for retiming of unit-delay circuitry, which operates on the original circuit graph. Specifically, given a unit-delay circuit $G = (V, E, 1, w)$ and a positive integer c, we determine a retiming r of G such that the clock-period $\Phi(G_r)$ of the retimed circuit G_r satisfies $\Phi(G_r) \leq c$. Our algorithm terminates in $O(VE)$ steps and it matches the best previously known strongly polynomial algorithm for the same problem [17], which, according to Theorem 3.1 is obtained by running Bellman-Ford on $G - 1/c$.

For the reader's convenience, we give here, without proof of correctness, the Bellman-Ford algorithm on $G - 1/c$ for unit-delay circuitry retiming.

Algorithm BF This algorithm, given a unit-delay circuit graph $G = (V, E, 1, w)$ and an upper bound c for the clock-period, determines a function $\rho : V \to \mathbf{R}$, such that the retimed circuit $G_{\lceil \rho \rceil}$ satisfies the clock-period constraint $\Phi(G_{\lceil \rho \rceil}) \leq c$.

1. For some vertex $s \in V$ set $\rho(s) = 0$. For all vertices v in $V - \{s\}$ set $\rho(v) = \infty$.

2. Repeat $V - 1$ times:
   For each edge $u \xrightarrow{e} v \in E$ set $\rho(v) = \min\{\rho(v), \rho(u) + w(e) - 1/c\}$.

3. For each edge $u \xrightarrow{e} v \in E$ set $w_r(e) = w(e) + \lceil \rho(u) \rceil - \lceil \rho(v) \rceil$. □

In our algorithm, the additive and multiplicative operations utilized are the MIN and $\odot$ operations introduced in the previous section. The elements of the set S of the

semiring are labels $h(v) = (r(v), d(v))$ associated with each vertex v of the graph. The algorithm proceeds as follows:

Algorithm R This algorithm, given a unit-delay circuit graph $G = (V, E, 1, w)$ and an upper bound c for the clock-period, determines a retiming r, such that the retimed circuit satisifes the clock period constraint $\Phi(G_r) \leq c$.

1. For some vertex $s \in V$ set $h(s) = (0, 0)$. For all vertices v in $V - \{s\}$ set $h(v) = (\infty, \infty)$.

2. Repeat $V - 1$ times:
   For each edge $u \xrightarrow{e} v \in E$ set $h(v) = \text{MIN}(h(v), h(u) \odot (w(e), 1))$.

3. For each edge $u \xrightarrow{e} v \in E$ set $w_r(e) = w(e) + r(u) - r(v)$. □

The correctness of Algorithm R is ensured by the following lemma, which shows that the operation of Algorithm R on G simulates the operation of Algorithm BF on $G - 1/c$.

Lemma 4.1 *Let $G = (V, E, 1, w)$ be a unit-delay circuit graph. Let $\rho(v)$ be the variables of Bellman-Ford on $G - 1/c$ for each vertex $v \in V$. Moreover, let $(r(v), d(v))$ be the variables of Algorithm R on G for each vertex $v \in V$. If both Algorithm BF and Algorithm R relax edges in the same order, then after each relaxation of an edge $u \rightarrow v \in E$ we have $\rho(v) = r(v) - d(v)/c$ for every vertex $v \in V$.*

Proof: The proof is by induction on the relaxations. Let $\rho^b(v)$ and $(r^b(v), d^b(v))$ denote the values of the variables of Algorithms BF and R respectively *before* the relaxation of an edge $x \rightarrow y$. Similarly, let $\rho^a(v)$ and $(r^a(v), d^a(v))$ denote the values of the variables *after* the relaxation of an edge $x \rightarrow y$. We shall show that if $\rho^b(v) = r^b(v) - d^b(v)/c$ for every vertex $v \in V$ and both algorithms relax the same edge $x \rightarrow y \in E$, then $\rho^a(v) = r^a(v) - d^a(v)/c$ for every vertex $v \in V$ after the relaxation.

Initially, before any relaxation is performed, the statement holds, assuming $\infty - \infty/c = \infty$ for every vertex $v \in V - \{s\}$ and since $0 - 0/c = 0$ for $v = s$. Now, let $x \xrightarrow{e} y$ be the edge to be relaxed. Then

$$(r^a(v), d^a(v)) = (r^b(v), d^b(v))$$

for every vertex $v \neq y$, and

$$\begin{aligned} r^a(y) &= \min\left\{r^b(y), r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor\right\}, \\ d^a(y) &= I\left(d^b(y), (d^b(x)+1) \bmod c\ ;\ r^b(y), r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor\right). \end{aligned}$$

We consider the following three cases, based on whether $r^b(y)$ is smaller than, greater than or equal to $r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor$.

1. $r^b(y) < r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor$. In this case the relaxation of edge $x \xrightarrow{e} y$ by Algorithm R yields

$$(r^a(y), d^a(y)) = (r^b(y), d^b(y)). \tag{4.6}$$

Now, we want to find $\rho^a(y)$ due to the relaxation of edge $x \xrightarrow{e} y$ by Algorithm BF. Since $r^b(v) \in \mathbf{Z}$ for all vertices $v \in V$, the inequality $r^b(y) < r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor$ implies that:

$$\begin{aligned} r^b(y) &\leq r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor - 1 \\ &< r^b(x) + w(e) - (d^b(x)+1)/c \\ &= \rho^b(x) + w(e) - 1/c, \end{aligned}$$

given that $1 + \left\lfloor (d^b(x)+1)/c \right\rfloor > (d^b(x)+1)/c$, for $d^b(x) \in \{0, \ldots, c-1\}$, and that $r^b(x) - d^b(x)/c = \rho^b(x)$ by the inductive assumption. We also have that

$$\begin{aligned} \rho^b(y) &= r^b(y) - d^b(y)/c \\ &\leq r^b(y). \end{aligned}$$

Therefore $\rho^b(y) < \rho^b(x) + w(e) - 1/c$ and the relaxation of edge $x \xrightarrow{e} y$ by Algorithm BF yields

$$\rho^a(y) = \rho^b(y). \tag{4.7}$$

From equations (4.6) and (4.7) and the inductive assumption we have that $\rho^a(y) = r^a(y) - d^a(y)/c$.

2. $r^b(y) > r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor$. In this case the relaxation of edge $x \xrightarrow{e} y$ by Algorithm R yields

$$(r^a(y), d^a(y)) = \left(r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor, (d^b(x)+1) \bmod c\right). \tag{4.8}$$

Now, we want to find $\rho^a(y)$ due to the relaxation of edge $x \xrightarrow{e} y$ by Algorithm BF. Since $r^b(v) \in \mathbf{Z}$ and $d^b(v)/c < 1$ for all vertices $v \in V$, the inequality $r^b(y) > r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor$ implies that:

$$\begin{aligned} \rho^b(y) &= r^b(y) - d^b(y)/c \\ &> r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor - d^b(v)/c \\ &\geq r^b(x) + w(e) - (d^b(x)+1)/c - d^b(v)/c \\ &= \rho^b(x) + w(e) - 1/c. \end{aligned}$$

Therefore the relaxation of edge $x \xrightarrow{e} y$ by Algorithm BF yields

$$\rho^a(y) = \rho^b(x) + w(e) - 1/c. \tag{4.9}$$

From equations (4.8) and (4.9) and the inductive assumption we have that

$$\begin{aligned} r^a(y) - d^a(y)/c &= r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor - \left((d^b(x)+1) \bmod c\right)/c \\ &= \rho^b(x) + d^b(x)/c + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor - \left((d^b(x)+1) \bmod c\right)/c \\ &= \rho^a(x) + (d^b(x)+1)/c - \left\lfloor (d^b(x)+1)/c \right\rfloor - \left((d^b(x)+1) \bmod c\right)/c \\ &= \rho^a(x), \end{aligned}$$

since

$$(d^b(x)+1)/c = \left\lfloor (d^b(x)+1)/c \right\rfloor + \left((d^b(x)+1) \bmod c\right)/c \tag{4.10}$$

for $d^b(x) \in \{0, \ldots, c-1\}$, as it can be easily verified by checking the cases for $d^b(x) = c-1$ and $d^b(x) < c-1$.

3. $r^b(y) = r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor$. In this case, the relaxation of edge $x \xrightarrow{e} y$ by Algorithm R yields

$$(r^a(y), d^a(y)) = \left(r^b(y), \max\left\{d^b(y), (d^b(x)+1) \bmod c\right\}\right). \tag{4.11}$$

Now, we want to find $\rho^a(y)$ due to the relaxation of edge $x \xrightarrow{e} y$ by Algorithm BF. By the inductive assumption and equation (4.10) the equality $r^b(y) = r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor$ implies that

$$\begin{aligned} \rho^b(y) &= \rho^b(x) + w(e) + (d^b(x) - d^b(y))/c - \left\lfloor (d^b(x)+1)/c \right\rfloor \\ &= \rho^b(x) + w(e) + (d^b(x) - d^b(y))/c - (d^b(x)+1)/c + \left((d^b(x)+1) \bmod c\right)/c \\ &= \rho^b(x) + w(e) - 1/c + \left((d^b(x)+1) \bmod c - d^b(y)\right)/c. \end{aligned}$$

We consider two cases, depending on the ordering of $(d^b(x)+1) \bmod c$ and $d^b(y)$.

Case A: $(d^b(x)+1) \bmod c - d^b(y) > 0$. In this case $\rho^b(y) > \rho^b(x) + w(e) - 1/c$ and Algorithm BF yields

$$\rho^a(y) = \rho^b(x) + w(e) - 1/c. \tag{4.12}$$

From equations (4.11), (4.12), and (4.10), and the inductive assumption we have that

$$\begin{aligned} r^a(y) - d^a(y)/c &= r^b(x) + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor - \left((d^b(x)+1) \bmod c \right)/c \\ &= \rho^b(x) + d^b(x)/c + w(e) - \left\lfloor (d^b(x)+1)/c \right\rfloor - \left((d^b(x)+1) \bmod c \right)/c \\ &= \rho^b(x) + d^b(x)/c + w(e) - (d^b(x)+1)/c \\ &= \rho^b(x) + w(e) - 1/c \\ &= \rho^a(x). \end{aligned}$$

Case B: $(d^b(x)+1) \bmod c - d^b(y) \leq 0$. In this case Algorithm BF yields

$$\rho^a(y) = \rho^b(y). \tag{4.13}$$

From equations (4.11) and (4.13) and the inductive assumption we have that

$$\begin{aligned} r^a(y) - d^a(y)/c &= r^b(y) - d^b(y)/c \\ &= \rho^b(y) \\ &= \rho^a(y). \end{aligned}$$

Therefore, we still have $r^a(y) - d^a(y)/c = \rho^a(y)$.

From cases 1, 2 and 3 we conclude that if both algorithms relax edges in the same order then $\rho(v) = r(v) - d(v)/c$ for every $v \in V$ after each relaxation. □

Now, the following theorem shows that the set of values $r(v)$ computed by Algorithm R yields a legal retiming of the unit-delay circuit G.

Theorem 4.2 *Let $G = (V, E, 1, w)$ be a unit-delay circuit graph and let c be a positive integer. Also, let $(r(v), d(v))$ be the variables of Algorithm R on G for each vertex $v \in V$. Then, after the termination of Algorithm R, $r(v)$ yields a retiming of G such that $\Phi(G_r) \leq c$.*

Proof: Let $\rho(v)$ be the variables of Bellman-Ford on $G - 1/c$ for each vertex $v \in V$. From Theorem 4.1 and the facts that $r(v) \in \mathbf{Z}$ and $d(v) < c$ for every $v \in V$, we have that $\lceil \rho(v) \rceil = r(v)$. This equality and the correctness of Algorithm BF imply that $r(v)$ yields a legal retiming of G, such that $\Phi(G_r) \leq c$. □

In summary, this chapter exhibits the closed semiring structure of retiming on a unit-delay circuit G and demonstrates a Bellman-Ford type algorithm, which uses the additive and multiplicative operations of the semiring in order to compute a legal retiming of the circuit. The algorithm operates only on the original graph G and its running time matches that of the best previously known strongly polynomial algorithm for the same problem.

Chapter 5

A Mixed-Integer Optimization Problem

This chapter investigates a mixed-integer optimization problem which arises in the mixed-integer optimization framework of retiming, as it was introduced in [17]. We present a polynomial time algorithm for the problem, that is based on the technique of introducing additional constraints, known as *cuts*, in such a way that the integrality constraints of the mixed-integer problem are met by the optimum solution of its linear programming relaxation.

The five sections of the chapter are organized as follows. Section 5.1 reviews the problem of finding a minimum-cost flow on a network. It also presents the dual problem of a minimum-cost flow and gives optimality conditions which relate primal and dual solutions. Section 5.2 introduces the mixed-integer optimization problem that we solve in this chapter. The problem is identified as the restricted case of a *mixed-integer dual of an uncapacitated minimum-cost flow*, because the relaxation of its integrality constraints reduces it to the dual of an uncapacitated minimum-cost flow problem. Based on this observation, we develop feasibility and optimality conditions for the mixed-integer problem in Section 5.3. Section 5.4 describes an algorithm that solves the mixed-integer problem in $O(V^3 \lg V)$ steps. Finally, Section 5.5 gives an application of our algorithm by reducing the problem of state minimization of synchronous circuitry to the mixed-integer problem that we solve in this chapter.

5.1 Preliminaries

In this section we give some basic background material on the problem of finding a minimum-cost flow in a network.

A *flow network* $G = (V, E, w, c)$ is an edge-weighted directed graph in which each edge $u \xrightarrow{e} v \in E$ has a weight $w(e)$ and capacity $c(e) \geq 0$. Let each vertex $v \in V$ have an associated real value $b(v)$ such that $\sum_{v \in V} b(v) = 0$. A *flow* in G is a real-valued function $f : E \rightarrow \mathbf{R}$ that satisfies the following two properties:

$$\sum_{v \xrightarrow{e} u \in E} f(e) - \sum_{u \xrightarrow{e} v \in E} f(e) = b(u) \tag{5.1}$$

for all vertices $u \in V$, and

$$0 \leq f(e) \leq c(e) \tag{5.2}$$

for all edges $u \xrightarrow{e} v \in E$.

A flow network $G = (V, E, w, c)$ with $c(e) = \infty$ for all edges $e \in E$ is called *uncapacitated*. For simplicity, in the rest of this paper we shall denote an uncapacitated network by $G = (V, E, w)$. The problem of finding a minimum-cost flow on an uncapacitated network $G = (V, E, w)$ is defined as follows.

Problem UMC-Flow *(Uncapacitated Minimum-Cost Flow)* Let $G = (V, E, w)$ be an uncapacitated flow network. Let each vertex $v \in V$ have an associated real value $b(v)$ such that $\sum_{v \in V} b(v) = 0$. A minimum-cost flow f on G is a flow that minimizes $\sum_{u \xrightarrow{e} v \in E} w(e) f(e)$. □

The linear programming dual of Problem UMC-Flow is defined as follows.

Problem DUMC-Flow *(Dual Uncapacitated Minimum-Cost Flow)* Let $G = (V, E, w)$ be an uncapacitated flow network. Let each vertex $v \in V$ have an associated real value $b(v)$ such that $\sum_{v \in V} b(v) = 0$. Determine a value $x(v)$ for each vertex $v \in V$ that maximizes $\sum_{v \in V} x(v) b(v)$ subject to

$$x(v) - x(u) \leq w(e) \tag{5.3}$$

for all edges $u \xrightarrow{e} v \in E$. □

Note that there are no integrality constraints on the solutions x. The mixed-integer problem that we solve, and that we present in the following section, has the form of Problem DUMC-Flow with the addition of integrality constraints on a subset of the variables in x.

The following theorem is a direct consequence of the primal-dual relation of Problems UMC-Flow and DUMC-Flow.

Theorem 5.1 *Let f^* be a flow that solves Problem UMC-Flow and let $Z_p(f^*) = \sum_{u \xrightarrow{e} v \in E} w(e) f^*(e)$. Similarly, let x^* be a flow that solves Problem DUMC-Flow and let $Z_d(x^*) = \sum_{v \in V} x^*(v) b(v)$. Then $Z_p(f^*) = Z_d(x^*)$.* □

Almost all algorithms for Problems UMC-Flow and DUMC-Flow rely on Theorem 5.1 and they usually yield a solution for both problems at the same time. A basic concept used in these algorithms is that of the *residual* network $G(f)$. The residual network $G(f)$ corresponding to a flow f is defined as follows: we replace each edge $u \rightarrow v \in E$ by two edges $u \xrightarrow{e} v$ and $v \xrightarrow{e'} u$. The edge $u \xrightarrow{e} v$ has cost $w(e)$ and a *residual capacity* $r(e) = u(e) - f(e)$, and the edge $v \xrightarrow{e'} u$ has cost $-w(e)$ and residual capacity $r(e') = f(e)$. The residual network consists only of arcs with positive residual capacity.

The following theorem gives a necessary and sufficient solution for a flow f to be optimum in terms of the residual network $G(f)$.

Theorem 5.2 *Let $G = (V, E, w, u)$ be a flow network. Then a flow f on G is optimum if and only if $G(f)$ contains no negative-weight directed cycles.* □

Finally, the following Lemma [2] demonstrates how to obtain an optimum solution x for Problem DUMC-Flow once an optimum flow f for Problem UMC-Flow is known.

Lemma 5.3 *Let $G = (V, E, w, u)$ be a flow network and let f^* be an optimum flow on G. Moreover, let $l(v)$ denote the length of the shortest-path in $G(f)$ from some source $s \in V$ to vertex $v \in V$. Then the assignment $x(v) = l(v)$ for every vertex $v \in V$ is an optimum solution for Problem DUMC-Flow.* □

5.2 Mixed-Integer Dual Minimum-Cost Flow

In this section we present the mixed-integer optimization problem that we solve in this chapter. We refer to the problem as the *restricted mixed-integer dual of uncapacitated minimum-cost flow* and we identify it as a special case of a general mixed-integer optimization problem.

The restricted mixed-integer dual of uncapacitated minimum-cost flow is defined as follows.

Problem RMI-Dual-Flow *(Restricted Mixed-Integer Dual of Uncapacitated Minimum-Cost Flow)* Given an uncapacitated flow network $G = (V, E, w)$ with $w(e) \in \mathbf{R}$, a set V_I such that $V_I \subseteq V$, and an integer $b(v)$ for each vertex $v \in V_I$ such that $\sum_{v \in V_I} b(v) = 0$ and $b(v) = 0$ for all $v \notin V_I$, find a value $x(v)$ for each vertex $v \in V$ that maximizes $Z(x) = \sum_{v \in V_I} x(v)b(v)$ subject to

$$x(v) - x(u) \leq w(e) \tag{5.4}$$

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

$$x(v) \in \mathbf{Z} \tag{5.5}$$

for every $v \in V_I$. □

Observe that the maximization of the sum is performed over the subset V_I of V, which is required to take on integer values. The reason that we identify the problem as the mixed-integer dual of an uncapacitated minimum-cost flow is that if we relax the integrality constraints (5.5) it reduces to Problem DUMC-Flow, the linear programming dual of an uncapacitated minimum-cost flow problem [12, 21]. Based on this observation, we describe in Section 5.4 an $O(V^3 \lg V)$ time procedure, which solves Problem RMI-DualFlow.

We can generalize Problem RMI-Dual-Flow by extending the set over which the maximization is performed to include the entire vertex set V of the graph.

Problem MI-Dual-Flow *(Mixed-integer Dual of Uncapacitated Minimum-Cost Flow)* Given an uncapacitated network $G = (V, E, w)$, with $w(e) \in \mathbf{R}$, a set V_I such that $V_I \subseteq V$, and an integer $b(v)$ for each vertex $v \in V$ such that $\sum_{v \in V} b(v) = 0$, find a

value $x(v)$ for each vertex $v \in V$ that maximizes $\sum_{v \in V} x(v)b(v)$ subject to

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

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

$$x(v) \in \mathbf{Z}$$

for every $v \in V_I$. □

We conjecture that, contrary to Problem RMI-Dual-Flow, Problem MI-Dual-Flow is not tractable.

Conjecture. *Problem MI-Dual-Flow is NP-Complete.* □

Two facts support our conjecture. First, the feasible vectors of Problem RMI-Dual-Flow do not form a convex set, due to the integrality constraints. Lack of convexity rules out linear programming approaches that lead to polynomial time algorithms [21, 2]. In addition, the solutions to Problem MI-Dual-Flow do not necessarily exhibit the optimal substructure property. There exist instances of Problem MI-Dual-Flow which in order to be solved require a locally suboptimal assignment of values to the unknowns. The lack of optimal substructure rules out dynamic programming approaches that could lead to polynomial time algorithms [4].

5.3 Feasibility and Optimality Conditions

In this section we develop feasibility and optimality conditions for Problem RMI-Dual-Flow. Specifically, we construct an auxiliary problem by augmenting the constraint-set of Problem RMI-Dual-Flow with new constraints, which are derived from the given constraint-set. The auxiliary problem has no explicit integrality constraints and we prove that it is feasible if and only if Problem RMI-Dual-Flow is feasible. Finally, we prove that a solution of the auxiliary problem solves Problem RMI-Dual-Flow as well.

First, let us describe how the additional constraints are obtained. Let $G = (V, E, w)$ be an edge-weighted graph and let $V_I \subseteq V$. We define the ***short-cut graph*** $G_S = (V, E_S, w_S)$ as follows.

$$\begin{aligned} E_S &= \{u \to v \ : \ u, v \in V_I,\ u \overset{p}{\rightsquigarrow} v \in G\}, \\ w_S(u \xrightarrow{e} v) &= \min\{w(p) \ : \ u \overset{p}{\rightsquigarrow} v \in G\}. \end{aligned}$$

We also define the *dense graph* $G_D = (V, E \cup E_S, w_D)$ with edge-weights defined as follows.

$$w_D(u \xrightarrow{e} v) = \begin{cases} w(e) & \text{if } e \in E, \\ \lfloor w_S(e) \rfloor & \text{if } e \in E_S. \end{cases}$$

The edges in E_S impose the additional constraints of Problem AUX. We define the auxiliary problem AUX in terms of the original graph G and its corresponding short-cut graph G_S.

Problem AUX *(Auxiliary Dual of Uncapacitated Minimum-Cost Flow)* Let $G = (V, E, w)$ with $w(e) \in \mathbf{R}$, be an edge-weighted graph and let $G_S = (V_S, E_S, w_S)$ be its corresponding short-cut graph. Given a set V_I such that $V_I \subseteq V$, and an integer $b(v)$ for each vertex $v \in V_I$ such that $\sum_{v \in V_I} b(v) = 0$ and $b(v) = 0$ for all $v \notin V_I$, find a value $x(v)$ for each vertex $v \in V$ that maximizes $Z(x) = \sum_{v \in V_I} x(v)b(v)$ subject to

$$x(v) - x(u) \leq w(e) \tag{5.6}$$

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

$$x(v) - x(u) \leq \lfloor w_S(e) \rfloor \tag{5.7}$$

for every edge $u \xrightarrow{e} v \in E_S$.

First, we shall prove that feasibility of RMI-Dual-Flow implies feasibility of AUX by showing that the set of solutions of RMI-Dual-Flow encompasses all solutions of AUX. We denote by $\mathcal{X}_{RMI}$ the set of feasible vectors for RMI-Dual-Flow and by $\mathcal{X}^*_{RMI}$ the set of optimum vectors for RMI-Dual-Flow. Similarly for AUX, we denote its set of feasible vectors by $\mathcal{X}_{AUX}$ and its set of optimum vectors by $\mathcal{X}^*_{AUX}$.

Lemma 5.4 *If $x \in \mathcal{X}_{RMI}$ then $x \in \mathcal{X}_{AUX}$.*

Proof: Let $x = (x(1), \ldots, x(|V|))$ be any vector in $\mathcal{X}_{RMI}$. Then, from inequality (5.4) we have that $x(v) - x(u) \leq w(e)$ for every edge $u \xrightarrow{e} v \in E$. Therefore, x satisfies inequality (5.6).

Also, for every $u, v \in V_I$ let $p = u \rightsquigarrow v$ be the shortest path in G from u to v. By applying inequality (5.4) along p and the definition of the short-cut graph G_S we have that $x(v) - x(u) \leq w_S(e)$ for $u \xrightarrow{e} v \in E_S$. Since $x(u)$ and $x(v)$ are integers from

constraint (5.5), we can write $x(v) - x(u) \leq \lfloor w_S(e) \rfloor$ for $u \xrightarrow{e} v \in E_S$. Therefore, x satisifies inequality (5.7) as well and consequently $x \in \mathcal{X}_{AUX}$. □

As an immediate consequence we have the following.

Corollary 5.5 *For any $x \in \mathcal{X}^*_{RMI}$ with $Z(x) = \sum_{v \in V_I} x(v)b(v)$ and for any $y \in \mathcal{X}^*_{AUX}$ with $Z(y) = \sum_{v \in V_I} y(v)b(v)$, we have*

$$Z(x) \leq Z(y).$$

Proof: Since $x \in \mathcal{X}^*_{RMI}$ we also have $x \in \mathcal{X}_{RMI}$. From Lemma 5.4 we infer that $x \in \mathcal{X}_{AUX}$ and therefore $Z(x) \leq Z(y)$ for every $y \in \mathcal{X}^*_{AUX}$. □

Now, we shall prove that feasibility of Problem AUX implies feasibility of Problem RMI-Dual-Flow:

Lemma 5.6 *If $\mathcal{X}_{AUX} \neq \emptyset$ then $\mathcal{X}_{RMI} \neq \emptyset$.*

Proof: From Lemma 2.3 and the definition of G_S and G_D we have that $\mathcal{X}_{AUX} \neq \emptyset$ exactly when G_S is well-defined and there exists no negative-weight cycle in G_D. Let $x(v)$ be the length of the shortest path in G_D from some source $s \in V$ to the vertex $v \in V$. Then, from Lemma 2.2, x satisfies

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

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

$$x(v) - x(u) \leq \lfloor w_S(e) \rfloor$$

for every edge $u \xrightarrow{e} v \in E_S$. Therefore, x satisfies inequality (5.4).

Moreover, since for any path $p = u \rightsquigarrow v$ in E with $u, v \in V_I$ there always exists an edge $u \xrightarrow{e} v \in E_D$ with $w_D(e) \leq w(p)$ and $w_D(e) \in \mathbf{Z}$, the shortest path in E_D from the source s to any vertex $v \in V_I$ will be on integer-weight edges only, provided $s \in V_I$. Thus, by setting $x(s) = 0$ we can ensure $x(v) \in \mathbf{Z}$ for all vertices $v \in V_I$. Therefore, x satisfies inequality (5.5) as well and consequently $x \in \mathcal{X}_{RMI}$. □

As a consequence of Lemmata 5.4 and 5.6, we have the following corollary.

Corollary 5.7 *Problem RMI-Dual-Flow is feasible if and only if the short-cut graph G_S is well-defined and the dense graph G_D has no negative-weight directed cycles.* □

In the remaining of this section we show how to obtain a solution of Problem AUX that solves Problem RMI-Dual-Flow as well.

First, we shall show that there exists a primal solution of Problem AUX which has a special structure. Then we demonstrate how we can exploit this special structure in order to find a solution for Problem RMI-Dual-Flow. Recall that, according to section 5.1, the primal of Problem AUX is an uncapacitated minimum-cost flow on $G_D = (V_D, E_D, w_D)$.

Lemma 5.8 *Let f be a flow on G_D that solves the primal of Problem AUX. Also, let $E_D^+(f) = \{u \xrightarrow{e} v \in E_D \ : \ f(e) > 0\}$. Then there exists a flow f' on G_D that solves the primal of Problem AUX such that*

$$E_D^+(f') \subseteq E_S.$$

Proof: Consider an optimum f on G_D with $f(e) > 0$ for some edge $u \xrightarrow{e} v \notin E_S$. We show that by rerouting flow we can always convert f to a new flow f' such that $Z_p(f) = Z_p(f')$ and $E_D^+(f') \subseteq E_S$. Since $u \xrightarrow{e} v \notin E_S$ there exists a path $p_1 = u_0 \xrightarrow{e_0} u_1 \xrightarrow{e_1} \ldots u_{k-1} \xrightarrow{e_{k-1}} u$ with $u_0 \in V_I$, $u_1, \ldots, u_{k-1} \notin V_I$, and $f(e_i) > 0$ for $i = 0, 1, \ldots k-1$, and a path $p_2 = v \xrightarrow{e_{l-1}} v_{l-1} \xrightarrow{e_{l-2}} \ldots u_1 \xrightarrow{e_0} v_0$ with $v_0 \in V_I$, $v_1, \ldots, v_{l-1} \notin V_I$, and $f(e_i) > 0$ for $i = 0, 1, \ldots l-1$. Note that as long as there exists an edge $u \xrightarrow{e} v \notin E_I$ with $f(e) > 0$, we can always find paths p_1 and p_2 constructed in the way above. If there were no such paths, then the node-balance constraints (5.1) would have been violated, since $b(v) = 0$ for every $v \notin V_I$.

Now, since $u_0, v_0 \in V_I$, and f is optimum, there exists an edge $u_0 \xrightarrow{e_I} v_0 \in E_S$ with $w_D(e_I) = w_D(p_1; e; p_2)$, where $p_1; e; p_2$ denotes the path formed by concatenating p_1, e, and p_2. Therefore, we can reroute $\min\{f(e_i) \ : \ e_i \in p_1; e; p_2\}$ units of flow through e_I and still maintain an optimum flow. Let f_a be the new optimum flow. Then

$$|E_D^+(f_a)| \leq |E_D^+(f)| - 1.$$

Therefore, repetition of this procedure until $E_D^+(f_a) \cap E = \emptyset$ yields an optimum flow f' such that $E_D^+(f') \subseteq E_S$. □

Now, we show how we can get a solution for Problem AUX that satisfies the integrality constraints of Problem RMI-Dual-Flow. The proof relies on Lemma 5.8 above and on Lemma 5.3 of Section 5.1.

Lemma 5.9 *Let f be a solution for the primal of Problem AUX with $E_D^+(f) \subseteq E_S$. Then there exists $x \in \mathcal{X}^*_{AUX}$ such that:*

$$x(v) \in \mathbf{Z}$$

for all $v \in V_I$.

Proof: Let $d_D(f, v)$ denote the length of a shortest-path in the residual graph $G_D(f)$ from a source $s \in V_I$ to a vertex $v \in V$. From Lemma 5.3 we know that once an optimum flow f for the primal of Problem AUX is known, the assignment $x(v) = d_D(f, v)$ for every vertex $v \in V$ yields a solution x to Problem AUX.

It remains to show that x satisfies $x(v) \in \mathbf{Z}$ for all $v \in V_I$. Let us denote by $l_D(f, s)$ the length of a path s in $G_D(f)$ and let p be a shortest-path in $G_D(f)$ from the source $s \in V_I$ to a vertex $v \in V_I$. We shall prove that $l_D(f, p) = d_D(f, v) \in \mathbf{Z}$. Let $q = v_0 \stackrel{e_0}{\rightarrow} v_1 \stackrel{e_1}{\rightarrow} \ldots \stackrel{e_{k-2}}{\rightarrow} v_{k-1} \stackrel{e_{k-1}}{\rightarrow} v_k$ be a part of p such that $v_0, v_k \in V_I$ and $v_1, \ldots, v_{k-1} \notin V_I$. Since $E_D^+(f) \subseteq E_S$, we have that either $e_i \in E \cup E_S$ for all edges $e_i \in p$, or that $k = 1$ and $v_0 \rightarrow v_1$ is a backward edge of a flow-carrying edge $v_1 \rightarrow v_0 \in E_S$. In the first case $l_D(f, q) \in \mathbf{Z}$, since q has to be a shortest-path from v_0 to v_k and there always exists an edge $e \in E_S$ such that $w_D(e) \leq \lfloor w(q) \rfloor$. In the second case $l_D(f, q) \in \mathbf{Z}$, since $v_1 \rightarrow v_0 \in E_S$ implies $w_D(v_1 \rightarrow v_0) \in \mathbf{Z}$ and $l_D(f, q) = -w_D(v_1 \rightarrow v_0)$ by definition. Therefore, $l_D(f, q) \in \mathbf{Z}$ for every q and consequently $l_D(f, p) \in \mathbf{Z}$. □

Now, we can easily infer that the solution of Problem AUX derived according to the way suggested in Lemma 5.9 is a solution for Problem RMI-Dual-Flow.

Theorem 5.10 *Let f be a solution for the primal of Problem AUX with $E_D^+(f) \subseteq E_S$. Let x be a solution of a single-source shortest-paths problem on $G_D(f)$ from a source $s \in V_I$. Then x is a solution for Problem RMI-Dual-Flow.*

Proof: From Lemma 5.3 we infer that $x \in \mathcal{X}^*_{AUX}$ and consequently x satisfies constraint (5.6):

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

for all $e \in E$. Therefore, x satisfies constraint (5.8) of Problem RMI-Dual-Flow.

From Lemma 5.9 we have that x also satisfies the integrality constraint (5.9) of Problem RMI-Dual-Flow. Therefore, $x \in \mathcal{X}_{RMI}$, which implies that $Z(y) \geq Z(x)$ for every $y \in \mathcal{X}^*_{RMI}$. But from Corollary 5.5 we have that $Z(y) \leq Z(x)$. Therefore, $x \in \mathcal{X}^*_{RMI}$ as well. □

5.4 The Algorithm

In this section we give the $O(V^3 \lg V)$ algorithm that solves Problem RMI-Dual-Flow. Its correctness relies on the theory developed in the previous section.

Algorithm RMI-Dual-Flow This algorithm determines a solution x for Problem RMI-Dual-FLow.

1. Compute the edges in E_S by solving an all-pairs shortest-paths problem on G. Fail if a negative-weight cycle is found.
2. Compute a min-cost flow f on the graph G_D.
3. Transform f into f' by rerouting flow in such a way that if $f'(e) > 0$ then $u \xrightarrow{e} v \in E_S$.
4. Compute the shortest-paths lengths $x(v)$ for each vertex v in $G_D(f')$ from a source $s \in V_I$. □

Step 1 requires V shortest-paths algorithms. The total cost is $O(V(E + V \lg V))$ using Johnson's all-pairs shortest-paths algorithm [10]. Step 2 executes one uncapacitated min-cost flow algorithm, which requires $O(V \lg V(E_D + V \lg V))$ steps, using Orlin's strongly polynomial algorithm [19]. Step 3 runs for $O(VE)$ time, since each rerouting eliminates flow from at least one edge in E and requires $O(V)$ steps. Step 4 can be implemented in $O(VE_D)$ time, using Bellman-Ford's algorithm for shortest-paths. Therefore, the overall running time is $O(V^3 \lg V)$.

5.5 An Application to State Minimization

In this section we present the *state minimization problem* for retiming from the mathematical programming perspective described in [17], and we give a reduction of the

problem to Problem RMI-Dual-Flow. The state minimization problem is defined as follows: For a given circuit $G = (V, E, d, w)$, determine a retiming of the circuit such that the total number of registers $\sum_{e \in E} w_r(e)$ of the retimed circuit is minimized.

First, we give without proof the following theorem from [17]. This theorem describes retiming as a mixed-integer programming problem.

Theorem 5.11 *Let $G = (V, E, d, w)$ be a synchronous circuit, and let c be a positive real number. There exists a retiming r of G such that $\Phi(G_r) \leq c$ 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*

$$\begin{aligned} R(v) - r(v) &\leq -d(v)/c, \\ r(v) - R(v) &\leq 1, \end{aligned}$$

for every vertex $v \in V$, and

$$\begin{aligned} r(v) - r(u) &\leq w(e), \\ R(v) - R(u) &\leq w(e) - d(v)/c, \end{aligned}$$

wherever $u \xrightarrow{e} v$. □

The number of registers $S(G_r)$ in the retimed circuit G_r is

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

where $S(G)$ is the number of registers in the original circuit. Since $S(G)$ is constant, minimizing $S(G_r)$ is equivalent to minimizing the quantity

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

which is a linear combination of the $r(v)$, since $(\text{outdegree}(v) - \text{indegree}(v))$ is constant for each v. Now, using Theorem 5.11 we can state the register minimization problem in its mixed-integer form:

Problem STMIN *(State Minimization)* Given a synchronous circuit $G = (V, E, d, w)$ and a positive number c, determine a retiming r of G such that $\Phi(G_r) \leq c$ and G_r has the minimum number of registers. Equivalently, find an assignment of a real value $R(v)$ and an integer value $r(v)$ to each vertex $v \in V$ that minimizes $\sum_{v \in V} r(v)(\text{outdegree}(v) - \text{indegree}(v))$ subject to

$$R(v) - r(v) \leq -d(v)/c, \tag{5.8}$$

$$r(v) - R(v) \leq 1, \tag{5.9}$$

for every vertex $v \in V$, and

$$r(v) - r(u) \leq w(e), \tag{5.10}$$

$$R(v) - R(u) \leq w(e) - d(v)/c, \tag{5.11}$$

wherever $u \xrightarrow{e} v$. □

The state minimization problem on $G = (V, E, d, w)$ can be seen from the perspective of the mixed-integer problem RMI-Dual-Flow on an uncapacitated network $G' = (V', E', w')$. The graph G' is defined as follows.

$$\begin{aligned} V' &= \{v_i : v \in V,\ i = 1, 2\}, \\ E' &= E'_1 \cup E'_2 \cup E'_3 \cup E'_4, \end{aligned}$$

where

$$\begin{aligned} E'_1 &= \{v_1 \rightarrow v_2 : v_1, v_2 \in V'\}, \\ E'_2 &= \{v_2 \rightarrow v_1 : v_1, v_2 \in V'\}, \\ E'_3 &= \{u_1 \rightarrow v_1 : u \rightarrow v \in E\}, \\ E'_4 &= \{u_2 \rightarrow v_2 : u \rightarrow v \in E\}. \end{aligned}$$

The edge-weight of each edge $e \in E'$ is

$$w'(e) = \begin{cases} -d(v)/c & \text{if } v_1 \xrightarrow{e} v_2 \in E'_1, \\ 1 & \text{if } v_2 \xrightarrow{e} v_1 \in E'_2, \\ w(e) & \text{if } u \xrightarrow{e} v \in E'_3, \\ w(e) - d(v)/c & \text{if } u \xrightarrow{e} v \in E'_4. \end{cases}$$

The unknown $r(v)$ of the state minimization problem corresponds to $x(v_1)$ and the unknown $R(v)$ corresponds to $x(v_2)$. The function b is defined on V' as $b(v_1) =$ (indegree(v) − outdegree(v)) for every vertex $v_1 \in V$, and $b(v_2) = 0$ for every vertex $v_2 \in V$. Finally, $V_I = \{v_1 \ : \ v_1 \in V'\}$.

In summary, in this chapter we gave a solution to a mixed-integer optimization problem. We identified the problem as the restricted mixed-integer dual of an uncapacitated minimum-cost flow by observing that its linear programming relaxation is the dual of an uncapacitated minimum-cost flow problem. Based on this observation we developed a theoretical framework for its solution and we gave a procedure that solves it in $O(V^3 \lg V)$ steps. Finally, we gave an application of our algorithm by reducing the state minimization problem for retiming to the mixed-integer problem that we solved.

Chapter 6

Conclusion

In this paper we have investigated properties of retiming, a synchronous circuitry optimization technique. We presented specialized, fast algorithms for retiming of combinational circuitry. Specifically, we showed that combinational circuitry can be pipelined with minimum latency in $O(E)$ steps, which is optimal within a constant factor. clock-period minimization of combinational circuitry can be achieved in $O(E \lg D)$ steps, where D is the maximum component delay in the circuit. We presented a novel and concise graph theoretic characterization of the minimum clock-period of a circuit. Based on this characterization we gave improved techniques for minimum clock-period retiming of sequential circuitry. We presented an $O(\min\{V^{1/2}E \lg(VD), VE\})$ algorithm for minimum clock-period retiming of unit-delay circuitry, and an $O(VE \lg D)$ algorithm for minimum clock-period retiming of general circuitry. We also showed that a retiming of a general circuit with clock-period that does not exceed the minimum by more than D can be found in $O(\min\{V^{1/2}E \lg(VW) \lg(VD), VE \lg(VD)\})$ steps. Subsequently, we exhibited the closed semiring structure of retiming and we gave an algorithm which operates based on this structure. Finally, we gave an $O(V^3 \lg V)$ time algorithm for a mixed-integer optimization problem, which arises in the linear programming framework of retiming.

There are still open questions of both practical and theoretical interest in the area. It is an interesting question whether there exists an algorithm for minimum clock-period retiming of general circuits that matches the running time of the algorithm for the same problem on unit-delay circuitry. Decoupling the running time of our algorithm for the mixed-integer optimization Problem RMI-Dual-Flow from the number of

the new constraints introduced will also be an interesting extension of the techniques presented in this thesis. Finally, proving the conjecture that Problem MI-Dual-Flow is intractable will fully elucidate the problem of optimizing mixed-integer difference constraints. Our conjecture is supported by the fact that the feasible vectors of Problem MI-Dual-Flow do not form a convex set as well as by the fact that the solutions to Problem MI-Dual-Flow do not necessarily exhibit the optimal substructure property. Lack of convexity and optimal substructure rules out linear programming and dynamic programming approaches, that could lead to polynomial-time algorithms.

Bibliography

[1] R. Aharoni, P. Erdös, and N. Linial. Dual integer linear programs and the relationship between their optima. *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, May 1985.

[2] R. Ahuja, J. Orlin, and T. Magnanti. Network flows. Technical Report OR 185-88, Operations Research Center, MIT, August 1988.

[3] V. Chvátal. *Linear Programming.* W. H. Freeman, New York, 1983.

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

[5] J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency. *Journal of the Association for Computing Machinery*, 19(2):248–264, 1972.

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

[7] H. Gabow. Scaling algorithms for network problems. *Journal of Computer and System Sciences*, 31:148–168, 1985.

[8] H. Gabow and R. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Computing*, October 1989.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., San Francisco, 1979.

[10] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the Association for Computing Machinery*, 24(1):1–13, January 1977.

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

[12] E. L. Lawler. *Combinatorial Optimization, Networks and Matroids.* Holt, Rinehart and Winston, New York, 1976.

[13] C. E. Leiserson. *Area-efficient VLSI Computation.* PhD thesis, Carnegie-Mellon University, 1981. Published in book form by the MIT Press, Cambridge, Massachusetts, 1983.

[14] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing synchronous circuitry by retiming. *3rd Caltech Conference on VLSI*, 1981. R. Bryant, ed., pp. 87-116.

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

[16] C. E. Leiserson and J. B. Saxe. A mixed-integer programming problem which is efficiently solvable. *Journal of Algorithms*, 9:114–128, 1988.

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

[18] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, 1988.

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

[20] J. B. Orlin and R. K. Ahuja. New scaling algorithms for the assignment and minimum cycle mean problem. Technical Report 2019-88, Sloan WP, 1988.

[21] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall, Inc., 1982.

[22] F. M. Rose. Models for VLSI circuits. Master's thesis, Massachusetts Institute of Technology, May 1982. Also available as MIT VLSI Memo No. 82–114.

[23] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1986.

[24] N. Weste and K. Eshraghian. *CMOS VLSI Design*. Addison-Wesley, Reading, Massachusetts, 1985.

This blank page was inserted to preserve pagination.

CS-TR Scanning Project
Document Control Form

Date : 3/10/95

Report # LCS-TR-486

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 67(76 IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☒ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form 2(PGS) ☐ Funding Agent Form ☒ Cover Page 2(COPIES)
☐ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): 2, 4, 6, 8

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:
IMAGE MAP (1-67) PAGES #'ED 1-67 (INCLUDING TITLE PAGE)
(68-73) SCANNING TRGT, COVER(2), PRINTER NOTES, DOD(2-PAGE)
(74-76) TRGTS

Scanning Agent Signoff:
Date Received: 3/10/95 Date Scanned: 3/14/95 Date Returned: 3/16/95

Scanning Agent Signature: Michael W. Cook

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/TR 486	5. MONITORING ORGANIZATION REPORT NUMBER(S) N00014-87-K-0825
6a. NAME OF PERFORMING ORGANIZATION MIT Lab for Computer Science 6b. OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Dept. of Navy
6c. ADDRESS (*City, State, and ZIP Code*) 545 Technology Square Cambridge, MA 02139	7b. ADDRESS (*City, State, and ZIP Code*) Information Systems Program Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*) 1400 Wilson Blvd. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
On Retiming Synchronous Circuitry and Mixed-Integer Optimization

12. PERSONAL AUTHOR(S)
Mario Christos Papaefthymiou

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Technical	FROM _____ TO _____	9/90	68

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Digital circuitry, systolic systems, parallel computation, computer-aided design, retiming, pipeling, propagation delay, group theory, semirings, mixed-integer optimization, network flow.

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

A.

In this paper we investigate properties of *retiming*, a circuit transformation which preserves the behavior of the circuit as a whole. We present an algorithm which transforms a given combinational circuit into a functionally equivalent pipelined circuit with minimum latency and clock-period no greater than a given upper bound c. The algorithm runs in $O(E)$ steps, where E is the number of interconnections in the circuit, and is optimal within a constant factor. We give a novel and concise characterization of the minimum clock-period of a circuit in terms of the maximum delay-to-register ratio cycle in the circuit. We show that this ratio does not exceed the minimum feasible clock-period by more than the maximum delay D of the elements in the circuit. This characterization leads to an $O(E \lg D)$ algorithm for minimum clock-

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Carol Nicolora	22b. TELEPHONE (*Include Area Code*) (617) 253-5894 22c. OFFICE SYMBOL

☆U.S. Government Printing Office: 1985—507-047

B. period pipelining of combinational circuitry with latency no greater than a given upper bound l, an $O(\min\{V^{1/2}E\lg(VD), VE\})$ algorithm for minimum clock-period retiming of unit-delay circuitry, an $O(VE\lg D)$ algorithm for minimum clock-period retiming of general circuitry and an $O(\min\{V^{1/2}E\lg(VW)\lg(VD), VE\lg(VD)\})$ algorithm for approximately minimum clock-period retiming, where V is the number of processing elements in the circuit. We demonstrate the closed semiring structure of retiming on unit-delay circuits under a given clock-period constraint. Finally, we give an $O(V^3\lg V)$ algorithm for a mixed-integer optimization problem which arises in the linear programming framework of retiming.