Retiming and Clock Scheduling for Digital Circuit Optimization

Xun Liu, *Student Member, IEEE,* Marios C. Papaefthymiou, *Member, IEEE,* and Eby G. Friedman, *Fellow, IEEE*

***Abstract*—This paper investigates the application of simultaneous retiming and clock scheduling for optimizing synchronous circuits under setup and hold constraints. Two optimization problems are explored: 1) clock period minimization and 2) tolerance maximization to clock-signal delay variations. Exact mixed-integer linear programming formulations and efficient heuristics are given for both problems. When both long and short paths are considered, circuits optimized by the combined application of retiming and clock scheduling can achieve shorter clock periods or demonstrate greater tolerance to clock-signal delay variations than circuits optimized by retiming or clock scheduling. Experiments with benchmark circuits demonstrate the effectiveness of the combined optimization. In comparison with the best result obtained by either of the two optimizations, the joint application of retiming and clock scheduling increased operating speeds by more than 8% on the average. It also increased tolerance to clock delay variations by an average of 12% over a broad range of target clock frequencies. Larger relative improvements were achieved for shorter clock periods, thus suggesting that simultaneous retiming and clock scheduling can play an important role in high-speed design.**

***Index Terms*—Delay tolerance, digital circuits, optimization delay variations, retiming, scheduling, timing.**

I. Introduction

PERIODIC clock signals play a central role in the design and performance of synchronous digital systems. They provide a synchronization mechanism that enables the straightforward implementation of finite state machines. They also determine the throughput of the computation performed by these machines, since data propagation through a digital system proceeds at the rate specified by the clock period. As clocking rates increase, variations in the propagation delays of the clock signals across a chip are becoming a significant fraction of the clock period and begin posing a fundamental challenge to the design of high-performance systems. Such variations are typically due to process parameter variations, temperature or environmental variations, and power supply variations and become particularly pronounced in submicron design. Consequently, in addition to clock period minimization, tolerance maximization to clock-signal delay variations has become of paramount importance to the design of high-performance digital circuits.

Retiming and clock scheduling are two elegant and powerful approaches to digital circuit performance optimization. Although completely different from an implementation standpoint, they both improve circuit performance by redistributing timing slack among the combinational paths of a circuit. Retiming decreases the critical delays of a circuit by relocating its registers so that combinational path delays are evenly balanced across the entire design. Instead of transforming a circuit, clock scheduling increases the timing slack along critical paths by introducing appropriate clock delays that differentiate, or skew, the arrival times of the clock signals at the circuit registers. Due to their common operating principle, retiming and clock scheduling are often referred to as being "equivalent," although they do not always achieve the same clock period when applied separately. Significant effort has been devoted to the individual study of the two optimizations. The investigation of their combined application has been limited, however.

This paper investigates the simultaneous application of retiming and clock scheduling for digital circuit optimization under setup and hold constraints. When both long and short paths are considered, the combined application of the two optimizations is more powerful than either of the two optimizations by itself, resulting in faster circuits or circuits that are more tolerant to variations in the delays of their clock signals. Moreover, the combined application of retiming and clock scheduling adds flexibility during circuit synthesis, providing alternative ways for achieving a target performance specification.

The effectiveness of simultaneous retiming and clock scheduling in minimizing the clock period of a digital circuit is demonstrated in the example of Fig. 1. In addition to demonstrating the speedup potential of simultaneous retiming and clock scheduling, Fig. 1 shows that the shortest clock period achievable by retiming and clock scheduling may not be obtainable by an "intuitive" application of the two optimizations in sequence, that is, first performing retiming for maximum speed (under zero skew) and then applying clock scheduling. In this figure, digital circuits are represented as directed graphs. Each vertex denotes a block of combinational logic, and each edge between a vertex pair denotes a wire between the corresponding logic blocks. The rectangles represent edge-triggered registers. The pair x/y inside each vertex denotes the minimum and maximum propagation delays of the signals through the corresponding logic block.

Manuscript received August 16, 2000; revised April 27, 2001. This work was supported in part by the National Science Foundation under Grants CCR-9796145, MIP-9610108, and CCR-0082876, a grant from the New York State Science and Technology Foundation, and by grants from the Xerox, IBM, and Intel Corporations. This paper was recommended by Associate Editor L. Stok.

X. Liu and M. C. Papaefthymiou are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: xunliu@eecs.umich.edu, marios@eecs.umich.edu).

E. G. Friedman is with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627 USA (e-mail: friedman@ece.rochester.edu).

Publisher Item Identifier S 0278-0070(02)01054-0.

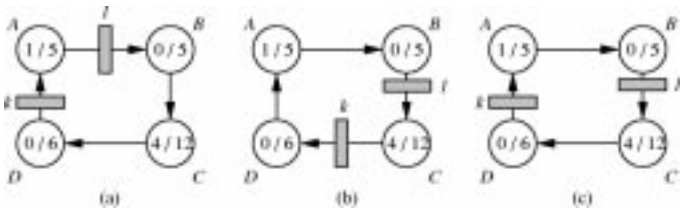

Fig. 1. Clock period improvement by simultaneous retiming and clock scheduling. (a) Original circuit. (b) Fastest retimed circuit with zero skew. (c) Fastest retimed circuit with nonzero skew.

When all clock skews are zero, the minimum period at which the circuit can be clocked is the longest delay among its combinational paths. For example, the original circuit in Fig. 1(a) achieves a clock period of $\pi = 23$ tu (time units) with zero clock skew. If nonzero clock skew is introduced, it is possible to operate the circuit at a shorter clock period that equals the largest difference between the maximum and minimum register-to-register delays over all paths in the circuit. Accordingly, if register l "sees" the clock edge by 4 tu earlier than register k, the circuit can function correctly with a clock period of $\pi = 19$ tu. This clock period is the shortest one that clock scheduling can achieve without introducing any signal races, since the difference in the maximum and minimum propagation delays along the path *BCD* is 19 tu.

The retimed circuit in Fig. 1(b) is obtained by first retiming the original circuit to minimize its clock period with zero skew and subsequently adjusting its clock schedule to further reduce its clock period. The new register locations result from a backward shift of register k across block D and a forward shift of register l across block B. With zero clock skew, this circuit is optimally retimed and achieves a clock period of $\pi = 16$ tu. If the clock edge arrives at register k by 1 tu earlier than at register l, this retimed circuit can achieve an even shorter clock period of $\pi = 15$ tu. No further clock period improvements are possible, however, because the propagation delay difference along path *DAB* is 15 tu.

Fig. 1(c) shows another retimed version of the original circuit that is obtained by shifting register l across block B. For this circuit, the shortest clock period that results in no timing violations under zero clock skew is $\pi = 18$ tu. If the clock edge arrives at register k by 4 tu later than register l, however, this circuit can function correctly with a clock period $\pi = 14$ tu. This clock period is the minimum that can be possibly achieved by applying both retiming and clock scheduling, since the total propagation delay around the cycle is 28 tu and must be distributed between two combinational paths.

The effectiveness of simultaneous retiming and clock scheduling in increasing the tolerance of a circuit to variations in the propagation delays of its clock signals is demonstrated by the example in Fig. 2. In this example, the clock skew between the input/output registers i and k is assumed to be zero. The setup and hold constraints along each of the two combinational paths yield a range $[x, y]$ of permissible clock delays [18] for the arrival of the clock signal at register j. The permissible skew range of j is obtained by intersecting the two possible ranges.

For the original circuit in Fig. 2(a) and a target clock period of 12 tu, the range of possible delays for the clock signal arriving at j is $[-2, 4]$. When clock delay is centered at zero, therefore, the permissible skew range of j is $[-2, 2]$, assuming symmetric clock delay variations, and the tolerance of the circuit is 4 tu. When clock signals are designed to arrive at j with a delay $s(j) = 1$, however, the permissible skew range is $[-2, 4]$, and delay tolerance increases to 6 tu.

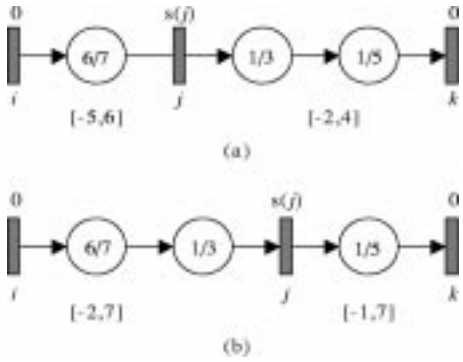

Fig. 2. Tolerance improvement by simultaneous retiming and clock scheduling. (a) Original and (b) retimed circuit.

A retimed version of the original circuit that is obtained by shifting j forward is shown in Fig. 2(b). In this case, the intersection of the two skew ranges is $[-1, 7]$. When clock skew is zero, the permissible range of j is $[-1, 1]$, and tolerance drops to 2 tu. When the arrival time of the clock signal at j is delayed by $s(j) = 3$, however, the permissible range becomes $[-1, 7]$, and tolerance increases to 8 tu. This value is the maximum tolerance that can be achieved by simultaneous retiming and clock scheduling for $\pi = 12$ tu. An interesting observation about the retimed circuit in Fig. 2(b) is that when skews are zero, the delay tolerance of this circuit is smaller than that in the original one. Nevertheless, when clock skews are nonzero, this circuit exhibits maximum tolerance to delay variations. It may thus be impossible to compute a maximum tolerance configuration by simply performing retiming for maximum tolerance, followed by clock scheduling for maximum tolerance.

This paper investigates the simultaneous application of retiming and clock scheduling for increasing the performance of a digital circuit. We first study the *retiming and clock scheduling problem*. In this problem, given a synchronous digital circuit, a target clock period, and a target tolerance on the variation of the clock arrival times at the circuit's registers, we wish to compute a retiming and a clock schedule so that the optimized circuit meets its performance specification. We then turn to the two optimization variants of the basic retiming and clock scheduling problem. In the *minimum-period retiming and clock scheduling problem*, given a synchronous circuit and a target tolerance, we wish to compute a retiming and a clock schedule such that the optimized circuit satisfies the tolerance constraints while achieving the minimum possible clock period. In the *maximum-tolerance retiming and clock scheduling problem*, given a synchronous circuit and a target clock period, we wish to compute a retiming and a clock schedule such that the optimized circuit meets its clock period constraint while achieving the maximum possible tolerance to variations in the clock signal arrival times.

We give an exact mixed-integer linear programming (MILP) formulation for the retiming and clock scheduling problem under setup and hold constraints. Our program has $O(E^2)$ constraints, where E is the number of wires in the circuit and can be solved exactly using general MILP solvers. Although MILP algorithms have worst case exponential complexity, our MILP formulation is valuable from a number of standpoints. Most notably, it gives a mathematically accurate set of necessary and sufficient conditions for the problem, thus providing a way for calculating the optimal solution. For relatively small circuits, the MILP formulation is thus useful in evaluating the effectiveness of heuristics. In addition to the MILP formulation of retiming and clock scheduling, we give practical heuristics for the minimum-period and the maximum-tolerance optimization problems. Our heuristics run faster than MILP-based optimization by several orders of magnitude with little or no sacrifice in accuracy.

To complement our analysis, we present extensive experimental results on modified LGSynth93 and ISCAS89 benchmark circuits that demonstrate the speed and tolerance improvements which can be achieved by combining retiming and clock scheduling. Our results also indicate the circuit structures that are more amenable to speed-up by retiming and clock scheduling. In our experiments, clock periods improve by 8% on the average and by as much as 42% in the best case over separate retiming or clock scheduling. Sequential application of retiming and clock scheduling is found to be effective for most benchmark circuits when optimizing for speed. For about a quarter of the circuits, however, simultaneous retiming and clock scheduling can further improve the clock frequency by up to 28%. With respect to delay variations, our heuristic outperforms the best of retiming, scheduling, or sequential retiming and scheduling by an average of 12% across a broad range of target clock periods. In certain cases, improvements of up to a multiplicative factor of 3 are achieved. In our experiments, the effectiveness of retiming and clock scheduling in improving tolerance to clock signal delay variations generally increases with the target operating frequency. Such a property is especially desirable in high-performance design, since it provides flexibility in achieving high operating speeds with high tolerance to delay variations.

The remainder of this paper has six sections. Background material is given in Section II. In Section III, we present a set of necessary and sufficient conditions for correct timing in a digital circuit that is optimized by retiming and clock scheduling. A novel MILP formulation for the retiming and clock scheduling problem is given in Section IV. In Section V, we describe a heuristic scheme for retiming and clock scheduling that can be adapted to solve the minimum-period and maximum-tolerance variants of the basic retiming and clock scheduling problem. The results of extensive experiments using LGSynth93 and ISCAS89 benchmark circuits are presented in Section VI. A summary of our contributions and directions for future research are given in Section VII.

II. Background

Section II-A presents the circuit and delay models we assume in this paper. Background material for retiming and clock scheduling is given in Section II-B and Section II-C, respectively. Previous research related to retiming and clock scheduling is discussed in Section II-D.

A. Circuit and Delay Model

An edge-triggered circuit is modeled as a directed multigraph $G = \langle V, E, d, \delta, w \rangle$. The vertices V correspond to the combinational logic elements in the circuit. The directed edges E model the interconnections between the combinational blocks. Each edge $e \in E$ corresponds to a wire that connects an output of a combinational block to the input of another combinational block, possibly through one or more globally clocked, edge-triggered registers. For each edge $e \in E$, the register count of the corresponding wire is given by a nonnegative, integer edge-weight $w(e)$. To avoid races, every directed cycle of G includes an edge with strictly positive register count.

Throughout this paper, we assume a data-independent delay model that considers timing constraints on long and short paths. Specifically, each vertex $v \in V$ is associated with a pair of nonnegative weights $d(v)$ and $\delta(v)$ which give the longest and shortest propagation delay, respectively, through the corresponding combinational logic block. All registers are assumed to have the same propagation delay t_d, setup time t_s and hold time t_h, regardless of their positions in the circuit. Given a target clock period π, we say that a circuit G achieves π if all setup and hold constraints in G hold. Intuitively, G contains no setup violations if for every combinational path p from a register k to a register l in G, data have sufficient time to propagate along p before the arrival of the latching clock edge at register l. Moreover, G contains no hold violations (also known as races) if the propagation of clock signals between k and l occurs faster than data propagation along p. A precise mathematical formulation of the setup and hold constraints that must hold to ensure correct timing is given in Section III.

B. Retiming

A *retiming* of an edge-triggered circuit $G = \langle V, E, d, \delta, w \rangle$ is an integer-valued vertex-labeling $r : V \rightarrow \mathbf{Z}$ that denotes a transformation of the original circuit G into a functionally equivalent circuit $G_r = \langle V, E, d, \delta, w_r \rangle$. For each edge $u \xrightarrow{e} v$ in G_r, w_r is defined by

$$w_r(e) = w(e) + r(v) - r(u) \,. \tag{1}$$

The retiming transformation for a vertex u in V is shown in Fig. 3. The output of u's computation in G_r is generated $r(u)$ clock cycles later than in G. The retimed circuit G_r is *well formed* if for all edges $e \in E$, we have

$$w_r(e) \geq 0 \,. \tag{2}$$

Equation (1) implies that for every vertex pair u, v in V, the change in the register count along *any* path $u \overset{p}{\rightsquigarrow} v$ depends solely on its two endpoints

$$w_r(p) = w(p) + r(v) - r(u) \tag{3}$$

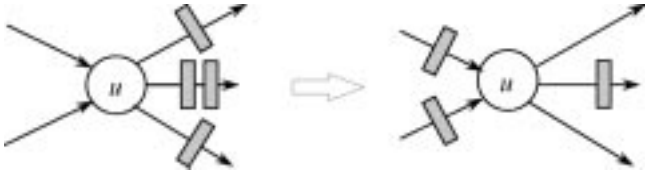

Fig. 3. Retiming of a vertex u by $r(u) = 1$.

where $w(p) = \sum_{e \in p} w(e)$. Thus, the maximum decrease in the register count of any path $u \overset{p}{\rightsquigarrow} v$ is

$$W(u,v) = \min\left\{w(p) : u \overset{p}{\rightsquigarrow} v\right\}. \tag{4}$$

Accordingly, given a retiming r, the number of registers along all paths $u \overset{p}{\rightsquigarrow} v$ with $w(p) = W(u,v)$ is $W_r(u,v) = W(u,v) + r(v) - r(u)$. The only paths $u \overset{p}{\rightsquigarrow} v$ that can become combinational (and possibly lead to a timing violation) in G_r are those for which $w(p) = W(u,v)$ in G. For each of the $O(V^2)$ vertex pairs u, v in V, the quantities

$$D(u,v) = \max\left\{d(p) : u \overset{p}{\rightsquigarrow} v, w(p) = W(u,v)\right\} \tag{5}$$

$$\Delta(u,v) = \min\left\{\delta(p) : u \overset{p}{\rightsquigarrow} v, w(p) = W(u,v)\right\} \tag{6}$$

where $d(p) = \sum_{x \in p} d(x)$, $\delta(p) = \sum_{x \in p} \delta(x)$ represent the longest and shortest propagation delays along the path p. Therefore, with zero skew, the clock period of any retimed circuit G_r is always some element in the $O(V^2)$-size set of $D(u,v)$.

When only setup constraints are considered, a retimed circuit that achieves a given clock period π can be computed in $O(VE)$. A retimed circuit that achieves the minimum possible clock period can be computed in $O(VE + V^2 \lg V)$ steps [11].

C. Clock Scheduling

In synchronous digital circuits, a clock signal provides a global time reference that synchronizes the flow of data between storage elements. Clock signals are delivered throughout the circuit by a clock distribution network which introduces delays in their propagation [8].

A *clock schedule* of an edge-triggered circuit $G = \langle V, E, d, \delta, w\rangle$ is a real-valued edge-labeling $s : E \to \mathbb{R}$. This labeling gives the propagation delay of each clock signal from the global clock source to the corresponding wire e in G. Timing violations can be fixed (or created) by adjusting these delays. For example, consider the path $? \overset{e}{\rightarrow} u \overset{p}{\rightsquigarrow} v \overset{\epsilon}{\rightarrow} ?$, where e and ϵ are wires with registers, and p is a path of combinational logic blocks. The difference $s(\epsilon) - s(e)$ is referred to as the *clock skew* between the registers e and ϵ. If $s(\epsilon) - s(e) \leq 0$, then the time available for the propagation of data from e to ϵ decreases by $s(e) - s(\epsilon)$. This adjustment may cause p to become a critical path in G or eliminate a race condition along p. Conversely, if $s(\epsilon) - s(e) \geq 0$, then the time available for data to propagate from e to ϵ increases by $s(\epsilon) - s(e)$. This increase may remove a long path violation or introduce a short path violation. In the *clock scheduling problem*, we wish to adjust the clock skews by appropriately selecting a clock schedule s so that no setup or hold violations exist in the circuit. Furthermore, in the *clock scheduling optimization problem*, clock skews are adjusted to implement the optimal circuit performance, for example, minimal clock period, without any setup or hold violations.

A variety of factors, including differences in interconnect delay, parasitic impedances, process parameter variations, and temperature variations, affect the delays in the propagation of the clock signals to their registers in the circuit. After fabrication, therefore, the actual delay of a clock signal with nominal propagation delay $s(e)$ varies within a "range" around $s(e)$. For every value of the clock signal delay in this range, a correctly designed circuit should meet all its timing constraints. The minimum and the maximum delays for the propagation of each clock signal to the corresponding wire e in G are denoted by labelings $s_m : E \to \mathbb{R}$ and $s_M : E \to \mathbb{R}$, respectively. For each wire e, the values $s_m(e)$ and $s_M(e)$ define a *range* $[s_m(e), s_M(e)]$ for $s(e)$.

Given a clock period π, we say that a circuit G achieves π with *tolerance* τ to the propagation delays of its clock signals if and only if there exist labelings s_m and s_M such that G is timed correctly with clock period π and for all wires $e \in E$, we have

$$\tau \leq s_M(e) - s_m(e). \tag{7}$$

The *maximum tolerance* $\tau_{\max}$ of G is defined as

$$\tau_{\max} = \max\{\tau : \exists s_M, s_m \text{ such that } G \text{ achieves } \pi \text{ and Inequality (7) holds}\}. \tag{8}$$

Given s_m and s_M, it is possible to compute a clock schedule s with tolerance τ if the distribution of the delay variations is known. Throughout this paper we assume that delay variations are distributed symmetrically. In this case, for each edge e in E, the quantity $s(e)$ can be computed straightforwardly by setting $s(e) = (s_m(e) + s_M(e))/2$.

D. Previous Research

Retiming has been investigated for a variety of clocking disciplines [9], [11], [14], [20], delay models [10], [21], and optimization objectives [2], [6], [17], [19]. In the context of edge-triggered circuits, a linear programming formulation of the clock scheduling problem was first described in [7], and a graph-theoretic approach to clock scheduling was presented in [5]. In both papers, the relative placement of the storage elements was assumed to be fixed. Algorithms for scheduling local clocks to improve the tolerance of an edge-triggered circuit to process parameter variations were described in [18].

The combined application of retiming and clock scheduling was first discussed in [16]. A mathematical framework for retiming and clock scheduling for maximum tolerance to clock delay variations under setup and hold constraints was investigated in [13]. A two-step procedure for minimizing the clock period of a synchronous circuit by combining retiming and clock scheduling was proposed in [3]. That work considered only setup violations. When both setup and hold constraints are considered, however, the solution space expands. The optimization of clock period in this broader solution space is discussed in [12].

Level-clocked circuits have the potential to achieve greater tolerance than edge-triggered ones with the same clock period.

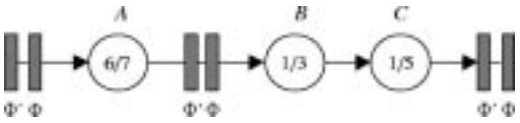

Fig. 4. A two-phase level-clocked implementation of the circuit in Fig. 2.

Algorithms for maximizing the tolerance of level-clocked circuits to clock signal delay variations were investigated in [1] and [15]. The work in [1] investigates clock period minimization and tolerance maximization using clock scheduling with fixed latch locations. In that paper, the timing slacks of all existing datapaths and the tolerance of all clock tree paths are improved by as much as possible, without using retiming to relocate registers and balance the timing slacks of different paths. On the other end of the spectrum, the work in [15] applies retiming with fixed clock skew values that are assumed to be given and are not free parameters to be optimized. Moreover, clocking schemes are constrained to avoid hold violations. In our proposed method, both register locations and clock skew values are variables that are adjusted to optimize circuit performance.

A direct comparison between our approach and the methods described in [1] and [15] is not possible because the quality of the results obtained with those methods can vary significantly depending on the *given* conditions, that is, the fixed locations of the latches in [1] and the fixed skew values and clocking scheme in [15]. Depending on the values of these parameters, our approach can outperform [1] and [15]. For example, Fig. 4 shows a two-phase level-clocked implementation of the edge-triggered circuit from Fig. 2(a). For the edge-triggered version of this circuit, combined retiming and clock scheduling can achieve a target clock period 12 tu with a tolerance of 8 tu, as shown in Section I. The clock scheduling algorithm in [1] cannot achieve a tolerance greater than 6 tu, however, since the timing constraints in that paper provide an upper bound of $\pi - D(u,v) + \Delta(u,v)$ for the tolerance. For $\pi = 12$ tu, the upper bound of 6 tu is obtained for $u = B$ and $v = C$ with $D(B,C) = 8$ tu and $\Delta(B,C) = 2$ tu. Furthermore, the retiming algorithm in [15] cannot achieve tolerance greater than 5 tu. Specifically, to achieve a clock period of 12 tu, retiming must place a latch on an edge other than the interface, since $D(A,C) = 15 > 12$ and [15] constrains the two clock phases to be nonoverlapping. Such a latch cannot have tolerance greater than $\pi - \max_{i\in\{A,B,C\}} d(i) = 5$ tu, however. In general, retiming and clock scheduling with level-clocked circuits is at least as effective as with edge-triggered circuits, assuming no skew variations among latches on the same wire. The combined application of the two optimizations on level-clocked circuits has yet to be investigated, however, and constitutes an interesting future research direction.

III. Retiming and Clock Scheduling Constraints

In this section, we give a set of necessary and sufficient conditions for correct timing in a digital circuit that is optimized by retiming and clock scheduling. We first give the setup and hold constraints that must be satisfied by a correctly timed circuit when the placement of its registers is given and all clock skews are zero. We also give the constraints for correct timing with nonzero clock skew and a lower bound on the circuit's tolerance. The conditions for correct timing with retiming and clock scheduling are then derived by adapting these constraints to include the relocation of registers. For simplicity, we assume that register propagation delays are zero. It is straightforward to extend our analysis to encompass registers with nonzero propagation delay t_d.

When all clock skews are zero, the arrival times of the clock signals at the registers are all equal. In this case, the maximum time allowable for the propagation of data along every combinational path between any two registers in the circuit is equal to the clock period π minus the register setup time t_s. The shortest clock period that will not result in setup violations is thus bounded from below by the sum of the register setup time and the delay of the longest combinational path in the circuit. To avoid races, the minimum data propagation time should not be shorter than the register hold time t_h. The following proposition gives a mathematical description of these setup and hold constraints that must hold for correct timing.

Proposition 1: Let $G = \langle V, E, d, \delta, w\rangle$ be an edge-triggered circuit, and let π be a constant. Then, G can be correctly timed by a zero-skew clock with period π if and only if for every edge pair $? \xrightarrow{e} u$, $v \xrightarrow{\epsilon} ?$ in E such that $w(e) \geq 1$, $w(\epsilon) \geq 1$, and $W(u,v) = 0$, we have

$$D(u,v) \leq \pi - t_s \tag{9}$$

$$\Delta(u,v) \geq t_h. \tag{10}$$

□

When clock scheduling is introduced, the arrival times of the clock signals can be adjusted to balance the maximum allowable propagation times between different register pairs. The following theorem from [7] gives $O(E^2)$ setup and hold constraints that must hold for a circuit to be correctly timed with a clock schedule s and a target clock period π.

Theorem 2: Let $G = \langle V, E, d, \delta, w\rangle$ be an edge-triggered circuit and π a constant. Moreover, let $s : E \to \mathbf{R}$ be a clock scheduling function. Then, the circuit is timed correctly if and only if for every edge pair $? \xrightarrow{e} u, v \xrightarrow{\epsilon} ?$ in E such that $w(e) \geq 1$, $w(\epsilon) \geq 1$, and $W(u,v) = 0$, we have

$$D(u,v) + s(e) - s(\epsilon) \leq \pi - t_s \tag{11}$$

$$\Delta(u,v) + s(e) - s(\epsilon) \geq t_h. \tag{12}$$

□

Inequalities (11) and (12) can be rewritten as difference constraints in the form of a shortest paths problem [4]. A clock schedule s that satisfies these constraints can be computed efficiently in $O(VE + V^2 \lg V)$ time by solving a shortest paths problem on an appropriately constructed edge-weighted graph $\mathcal{H} = \langle V_{\mathcal{H}}, E_{\mathcal{H}}, w_{\mathcal{H}}\rangle$. For each edge $e \in E$, the set $V_{\mathcal{H}}$ includes a vertex u_e. For each inequality (11), the set $E_{\mathcal{H}}$ includes an edge $u_\epsilon \xrightarrow{i} u_e$ with weight $w_{\mathcal{H}}(i) = \pi - t_s - D(u,v)$. Moreover, for each inequality (12), $E_{\mathcal{H}}$ includes an edge $u_\epsilon \xrightarrow{i} u_e$ with weight $w_{\mathcal{H}}(i) = \Delta(u,v) - t_h$.

Due to variations in the physical parameters of a circuit, the delay values of its clock signals may change. In deep-submicron VLSI technologies, where interconnect delays dominate,

such variations can be significant compared with gate delays. A correctly designed circuit should be able to tolerate this variation, which means, when the clock scheduling variable $s(e)$ assumes any value in the interval $[s_m(e), s_M(e)]$, the constraints in Theorem 2 should always hold. Therefore, they should also hold under the worst case assumptions, as stated in the following lemma.

Lemma 3: Let $G = \langle V, E, d, \delta, w\rangle$ be an edge-triggered circuit, and let π be a given constant. Moreover, let $s_m : E \to \mathbf{R}$ and $s_M : E \to \mathbf{R}$ be assignments of minimum and maximum clock delays, respectively. Then, G is timed correctly if and only if for every edge pair $? \xrightarrow{e} u$, $v \xrightarrow{\epsilon} ?$ in E such that $w(e) \geq 1$, $w(\epsilon) \geq 1$, and $W(u, v) = 0$, we have

$$D(u,v) + s_M(e) - s_m(\epsilon) \leq \pi - t_s \tag{13}$$
$$\Delta(u,v) + s_m(e) - s_M(\epsilon) \geq t_h \tag{14}$$

where t_h and t_s are hold and setup time, respectively. □

Proof: We will show that (13) holds if and only if all setup constraints in the circuit hold. The proof for the hold constraints is similar.

($\Rightarrow$) Assume (13) holds. For any clock schedule $s(e) \in [s_m(e), s_M(e)]$, we have

$$\begin{aligned} D(u,v) + s(e) - s(\epsilon) &\leq D(u,v) + s(e) - s_m(\epsilon) \\ &\leq D(u,v) + s_M(e) - s_m(\epsilon) \\ &\leq \pi - t_s \end{aligned}$$

since $s_m(\epsilon) \leq s(\epsilon)$, $s_M(e) \geq s(e)$, and (13) holds. From (11) it follows that there are no setup violations.

($\Leftarrow$) if (13) is violated, then the setup constraint between the edges e and ϵ is violated. □

The setup and hold constraints for correct timing under nonzero clock skew and a specified tolerance to the variation of clock delays follow immediately from Lemma 3.

Corollary 4: Let $G = \langle V, E, d, \delta, w\rangle$ be an edge-triggered circuit. Moreover, let π and τ be given real constants. Then, G achieves a clock period π with tolerance τ if and only if there exist functions $s_m : E \to \mathbf{R}$ and $s_M : E \to \mathbf{R}$ such that for each edge $u \xrightarrow{e} v$

$$\tau \leq s_M(e) - s_m(e) \tag{15}$$

and for every edge pair $? \xrightarrow{e} u$, $v \xrightarrow{\epsilon} ?$ in E such that $w(e) \geq 1$, $w(\epsilon) \geq 1$, and $W(u, v) = 0$

$$D(u,v) + s_M(e) - s_m(\epsilon) \leq \pi - t_s \tag{16}$$
$$\Delta(u,v) + s_m(e) - s_M(\epsilon) \geq t_h. \tag{17}$$

□

By relying on Corollary 4 we can give a precise mathematical formulation for the *retiming and clock scheduling problem*. In this problem, we wish to compute a retiming r, a clock schedule s_M, and s_m, so that the optimized circuit achieves a clock period π with no timing violations and with tolerance τ. The following theorem formulates this problem as a set of $O(E^2)$ constraints.

Theorem 5: Let $G = \langle V, E, d, \delta, w\rangle$ be a synchronous circuit, and let π and τ be given constants. Moreover, let $r : V \to \mathbf{Z}$ be a retiming function, let $s_M : E \to \mathbf{R}$ be an assignment of maximum clock delays, and let $s_m : E \to \mathbf{R}$ be an assignment of minimum clock delays. Then the retimed circuit G_r is well formed and achieves a clock period π with tolerance τ if and only if for every edge $u \xrightarrow{e} v \in E$, we have

$$\tau \leq s_M(e) - s_m(e) \tag{18}$$
$$w(e) + r(v) - r(u) \geq 0 \tag{19}$$

and for every pair of edges $? \xrightarrow{e} u, v \xrightarrow{\epsilon} ? \in E$, we have

$$\begin{aligned} A(e,\epsilon) > 0 &\Rightarrow W_r(u,v) \geq 1 \\ \text{or} \quad (w_r(e) = 0 \quad &\text{or} \quad w_r(\epsilon) = 0) \end{aligned} \tag{20}$$

$$\begin{aligned} A^*(e,\epsilon) > 0 &\Rightarrow W_r(u,v) \geq 1 \\ \text{or} \quad (w_r(e) = 0 \quad &\text{or} \quad w_r(\epsilon) = 0) \end{aligned} \tag{21}$$

where $A(e,\epsilon) = D(u,v) + s_M(e) - s_m(\epsilon) - \pi + t_s$ and $A^*(e,\epsilon) = -(\Delta(u,v) + s_m(e) - s_M(\epsilon) - t_h)$ for the setup and hold constraints, respectively. □

Proof: Inequality (18) follows from (7) and ensures that the tolerance of G_r is τ. Inequality (19) follows from (1) and (2) and, therefore, the resulting circuit G_r is well formed.

It remains to show that the timing constraints are satisfied. We will prove that (20) holds if and only there are no setup violations in G_r. The proof for the hold constraints is similar.

($\Rightarrow$) We show that if the setup constraints hold then (20) holds. The proof will be by contradiction. Assume that for every vertex pair u and v in G_r, (13) holds. Moreover, let $? \xrightarrow{e} u, v \xrightarrow{\epsilon} ?$ be two edges in G such that $A(e,\epsilon) = D(u,v) + s_M(e) - s_m(\epsilon) - \pi + t_s > 0$, $W_r(u,v) = 0$, $w_r(e) \geq 1$, and $w_r(\epsilon) \geq 1$. It follows that for e and ϵ, we have $D(u,v) + s_M(e) - s_m(\epsilon) > \pi - t_s$, which contradicts (13).

($\Leftarrow$) We now show that if some setup constraint in G_r is violated then (20) does not hold. The violation of (13) implies that for some edge pair $? \xrightarrow{e} u$, $v \xrightarrow{\epsilon} ?$ in G_r such that $w_r(e) \geq 1$, $w_r(\epsilon) \geq 1$, and $W_r(u,v) = 0$, we have $D(u,v) + s_M(e) - s_m(\epsilon) > \pi - t_s$. By the definition of $A(e,\epsilon)$ and a straightforward algebraic manipulation, it follows that $A(e,\epsilon) = D(u,v) + s_M(e) - s_m(\epsilon) - \pi + t_s > 0$. Therefore, (20) does not hold for the edges $? \xrightarrow{e} u$ and $v \xrightarrow{\epsilon} ?$ in G_r. □

IV. Exact MILP Formulation

The constraints in the statement of Theorem 5 do not appear to be amenable to efficient algorithmic solutions. In this section we recast them in the form of an equivalent exact mixed-integer linear program, thus enabling the application of powerful MILP solvers to the retiming and clock scheduling problem. We express the retiming and clock scheduling problem as a set of $O(E^2)$ linear inequalities with integer and real unknowns. This set is obtained by restricting the solution space of the constraints in Theorem 5 while maintaining their feasibility.

First, for each $e \in E$, we introduce two new integer variables $g(e)$ and $h(e)$ such that

$$w_r(e) = g(e) + h(e) \tag{22}$$
$$g(e) \geq 0 \tag{23}$$
$$h(e) \geq 0 \tag{24}$$
$$g(e) \leq 1 \tag{25}$$
$$h(e) \leq F \cdot g(e). \tag{26}$$

The parameter F in (26) is the maximum number of registers that retiming can place on any edge in G_r and equals

$$F = \max\{F_{loop}, F_{I/O}\} \tag{27}$$

where $F_{loop} = \max\{W(u,v) + W(v,u) : u, v \in V\}$, and $F_{I/O} = \max\{W(u,v) : u \in PI, v \in PO\}$, and PI and PO are the sets of the primary inputs and outputs, respectively, in G. This upper bound follows directly from the fact that retiming does not change the register count of cycles or I/O paths in a circuit graph.

The following lemma proves that $g(e)$ is an indicator variable for $w_r(e)$. Intuitively, $g(e)$ maps each edge $e \in E$ to the set $\{0,1\}$, indicating whether the retimed register count $w_r(e)$ is positive or not. Accordingly, $h(e)$ gives the number of registers on e in excess of 1, after retiming.

Lemma 6: Assuming that the conditions in the statement of Theorem 5 hold, we have

$$w_r(e) = 0 \Leftrightarrow g(e) = 0. \tag{28}$$

□

Proof: ($\Rightarrow$) Since $g(e)$ and $h(e)$ are nonnegative and their sum equals $w_r(e)$, if $w_r(e) = 0$ then $g(e) = 0$.

($\Leftarrow$) If $g(e) = 0$, (26) and (24) yield $h(e) = 0$. Therefore, their sum, $w_r(e)$, is zero. □

Using $g(e)$ and $h(e)$, (20) and (21) can be simplified by eliminating the implication and reducing the number of disjunctions it includes. The following lemma gives equivalent relations with only one disjunction.

Lemma 7: Assuming that the conditions in the statement of Theorem 5 hold, (20) and (21) are equivalent to the disjunction

$$A(e,\epsilon) \leq 0 \quad \text{or} \quad g(e) + g(\epsilon) - W_r(u,v) \leq 1 \tag{29}$$

$$A^*(e,\epsilon) \leq 0 \quad \text{or} \quad g(e) + g(\epsilon) - W_r(u,v) \leq 1 \tag{30}$$

where $A(e,\epsilon) = D(u,v) + s_M(e) - s_m(\epsilon) - \pi + t_s$ and $A^*(e,\epsilon) = -(\Delta(u,v) + s_m(e) - s_M(\epsilon) - t_h)$ for the setup and hold constraints, respectively. □

Proof: Since the predicate $p \Rightarrow q$ is equivalent to $\overline{p} \vee q$, (20) and (21) can be rewritten in the equivalent form

$$A(e,\epsilon) \leq 0 \text{ or } (W_r(u,v) \geq 1 \text{ or } w_r(e) = 0 \text{ or } w_r(\epsilon) = 0) \tag{31}$$

$$A^*(e,\epsilon) \leq 0 \text{ or } (W_r(u,v) \geq 1 \text{ or } w_r(e) = 0 \text{ or } w_r(\epsilon) = 0). \tag{32}$$

Therefore, to prove the equivalence of relations (20) and (21) with relations (29) and (30), it remains to show that

$$W_r(u,v) \geq 1 \text{ or } w_r(e) = 0 \text{ or } w_r(\epsilon) = 0 \tag{33}$$

holds if and only if

$$g(e) + g(\epsilon) - W_r(u,v) \leq 1 \tag{34}$$

holds.

($\Rightarrow$) Assuming that (33) holds, we prove that (34) holds by a straightforward case analysis on the values of $W_r(u,v)$, $w_r(e)$, and $w_r(\epsilon)$.

Let $W_r(u,v) \geq 1$. Since $g(e) \leq 1$ and $g(\epsilon) \leq 1$. Therefore

$$\begin{aligned} g(e) + g(\epsilon) - W_r(u,v) &\leq 1 + 1 - W_r(u,v) \\ &\leq 1 \end{aligned}$$

and (34) holds.

Now, assume that $w_r(e) = 0$. From Lemma 6, we have $g(e) = 0$. Since $g(\epsilon) \leq 1$ and $W_r(u,v) \geq 0$, we have

$$\begin{aligned} g(e) + g(\epsilon) - W_r(u,v) &\leq 0 + 1 - W_r(u,v) \\ &\leq 1 \end{aligned}$$

and therefore (34) holds. The proof for the case $w_r(\epsilon) = 0$ is similar.

($\Leftarrow$) We prove the contrapositive. Suppose that (33) does not hold. Then, from (2), (25), and (28), we have

$$W_r(u,v) = 0 \text{ and } g(e) = 1 \text{ and } g(\epsilon) = 1. \tag{35}$$

It follows that $g(e) + g(\epsilon) - W_r(u,v) = 1 + 1 - 0 > 1$, and therefore (34) does not hold. □

In the following lemma we show how (29) and (30) can be replaced by two linear inequalities, thus paving the way for the formulation of the retiming and clock scheduling problem as a mixed-integer linear program. To obtain these bounds, we assume that the parameter A and A^* are bounded from above by some known constant $A_{\max}$ that depends on the chip die size and physical constraints on the chip realization.

Lemma 8: Assuming that the conditions in the statement of Theorem 5 hold, (29) and (30) are equivalent to

$$g(e) + g(\epsilon) - W_r(u,v) \leq 2 - \frac{A(e,\epsilon)}{A_{\max}} \tag{36}$$

$$g(e) + g(\epsilon) - W_r(u,v) \leq 2 - \frac{A^*(e,\epsilon)}{A_{\max}} \tag{37}$$

where $A(e,\epsilon) = D(u,v) + s_M(e) - s_m(\epsilon) - \pi + t_s$ and $A^*(e,\epsilon) = -(\Delta(u,v) + s_m(e) - s_M(\epsilon) - t_h)$ for the setup and hold constraints, respectively, and $A_{\max}$ is an upper bound on $A(e,\epsilon)$ and $A^*(e,\epsilon)$. □

Proof: ($\Rightarrow$) We first prove that (29) implies (36).

Inequality (36) follows by a straightforward case analysis on the value of the parameter $A(e,\epsilon)$. If $A(e,\epsilon) \leq 0$, since G_r is well formed, $g(e) + g(\epsilon) - W_r(u,r) \leq 1 + 1 - W_r(u,r) \leq 2 \leq 2 - A(e,\epsilon)/A_{\max}$. If $A(e,\epsilon) > 0$, then (29) implies that

$$\begin{aligned} g(e) + g(\epsilon) - W_r(u,v) &\leq 1 \\ &\leq 2 - \frac{A(e,\epsilon)}{A_{\max}} \end{aligned}$$

since the sum $g(e) + g(\epsilon) - W_r(u,v)$ is integer and $A(e,\epsilon)/A_{\max} \leq 1$. Therefore, (36) holds. The proof for (37) is similar by replacing A with A^*.

($\Leftarrow$) We now prove that (36) implies (29) by a case analysis on the value of $A(e,\epsilon)$. If $A(e,\epsilon) \leq 0$, then (29) trivially holds. If $A(e,\epsilon) > 0$, then $2 - A(e,\epsilon)/A_{\max} < 2$. Since $g(e) + g(\epsilon) - W_r(u,v)$ is an integer, (36) yields $g(e) + g(\epsilon) - W_r(u,v) \leq 1$, and therefore Relation (29) holds. Similarly, we can prove (37) implies (30). □

Based on Lemmas 7 and 8, the problem of retiming and clock scheduling problem can now be recast as a mixed-integer linear

program with $O(E^2)$ linear inequalities, $|V| + |E|$ integer unknowns, and $2 \cdot |E|$ real unknowns. The complete statement of this program is given in the following theorem.

Theorem 9: Let $G = \langle V, E, d, \delta, w \rangle$ be an edge-triggered circuit, and let π and τ be given constants. Then there exists a retiming function $r : V \to \mathbf{Z}$ and clock schedules $s_M : E \to \mathbf{R}$ and $s_m : E \to \mathbf{R}$ such that the retimed circuit G_r is well formed and achieves clock period π with tolerance τ if and only if there exists a retiming function $r : V \to \mathbf{Z}$, clock schedules $s_M : E \to \mathbf{R}$ and $s_m : E \to \mathbf{R}$, and functions $g : E \to \mathbf{Z}$ and $h : E \to \mathbf{Z}$ such that for every edge $u \xrightarrow{e} v \in E$, we have

$$w(e) + r(v) - r(u) = g(e) + h(e) \tag{38}$$
$$g(e) \geq 0 \tag{39}$$
$$h(e) \geq 0 \tag{40}$$
$$g(e) \leq 1 \tag{41}$$
$$h(e) \leq F \cdot g(e) \tag{42}$$

for every edge $e \in E$, we have

$$\tau \leq s_M(e) - s_m(e) \tag{43}$$

and for every pair of edges $u \xrightarrow{e} v, \mu \xrightarrow{\epsilon} \nu \in E$, we have

$$A(e, \epsilon) \leq A_{\max} \tag{44}$$
$$A^*(e, \epsilon) \leq A_{\max} \tag{45}$$
$$g(e) + g(\epsilon) - W_r(u, v) \leq 2 - \frac{A(e, \epsilon)}{A_{\max}} \tag{46}$$
$$g(e) + g(\epsilon) - W_r(u, v) \leq 2 - \frac{A^*(e, \epsilon)}{A_{\max}} \tag{47}$$

where $A(e, \epsilon) = D(v, \mu) + s_M(e) - s_m(\epsilon) - \pi + t_s$ and $A^*(e, \epsilon) = -(\Delta(v, \mu) + s_m(e) - s_M(\epsilon) - t_h)$ for the setup and hold constraints, respectively. □

Proof: This theorem follows directly from Theorem 5 and Lemmas 7 and 8. Relation (38) and inequalities (39)–(42) ensure that G_r is well formed and $g(e)$ is an indicator of $w_r(e)$. Inequality (43) ensures that the resulting schedule has tolerance τ. Inequalities (44) and (45) enforce the upper bound of the parameters A and A^*. Inequalities (46) and (47) enforce the setup and hold constraints for the clock period π. The unknowns are the $|V|$ integers $r(v)$, the $|E|$ integers $g(e)$, the $|E|$ integers $h(e)$, the $|E|$ reals $s_M(e)$, and the $|E|$ reals $s_m(e)$. (Note that the variables $h(e)$ are not independent, since (38) can be eliminated by substituting $h(e) = w(e) + r(v) - r(u) - g(e)$ into the left-hand sides of (40) and (42).) All inequalities are linear in their unknowns. □

V. Heuristic Optimization

In this section we describe a heuristic approach for solving the minimum-period and the maximum-tolerance optimization versions of the retiming and clock scheduling problem. Even though the MILP constraints presented in Section IV can be used to compute an exact solution to the problem, the worst case exponential runtime of MILP solvers becomes evident even for circuits of relatively small size. Our heuristics run substantially faster than the exact MILP solvers. They rely on a common procedure that identifies critical paths based on clock scheduling information and generates a retimed circuit with improved clock period or tolerance to delay variations. To obtain an optimal circuit, this scheduling-guided retiming procedure is applied iteratively until a local optimum of the solution space is reached. Section V-A describes our Procedure SGR for scheduling-guided retiming. Section V-B proves that SGR correctly identifies and removes timing bottlenecks. Section V-C gives our heuristics for the problems of minimum-period retiming and clock scheduling and maximum-tolerance retiming and clock scheduling.

A. Scheduling-Guided Retiming

The scheduling-guided retiming procedure we describe in this section relies on three auxiliary *slack graphs* to encode and manipulate *tight* path constraints, that is, timing and tolerance constraints that hold with equality for given π and τ. Specifically, a graph $S(\pi, \tau)$ is used to encode the tight setup constraints, a graph $H(\pi, \tau)$ is introduced to capture the tight hold constraints, and a graph $T(\pi, \tau)$ is constructed to encode both the setup and the hold constraints in a single representation.

The three slack graphs are constructed as follows. For each edge e in G such that $w(e) > 0$, a vertex is introduced in all three graphs. Each vertex is associated with a weight equal to the register count $w(e)$ of its corresponding edge e. For simplicity, we overload e to denote in each slack graph the vertex of a corresponding edge e in G. Similarly, we use $w(e)$ to denote the corresponding vertex weight. All edges connected to a primary input or output of G are assumed to have a register and are represented in each slack graph by a single vertex $e_{I/O}$ with $w(e_{I/O}) = 1$. The register on the edge $e_{I/O}$ serves as the I/O interface of the circuit and is not relocated by retiming.

The directed edges of the slack graphs correspond to combinational register-to-register paths in G for which both the timing and the tolerance constraints are satisfied with equality. Specifically, for each pair of edges $u \xrightarrow{e} v, \mu \xrightarrow{\epsilon} \nu$ in G such that $w(e) \geq 1$, $w(\epsilon) \geq 1$, and $W(v, \mu) = 0$, an edge $e \to \epsilon$ is introduced in $S(\pi, \tau)$ if

$$D(v, \mu) + s_m(e) + \tau - s_m(\epsilon) = \pi - t_s. \tag{48}$$

Equation (48) is derived by combining (15) and (16) and replacing the inequality by equality. Whenever (48) holds, the corresponding setup constraint is said to be *tight* and gives rise to a timing bottleneck. Similarly, an edge $e \to \epsilon$ is added to $H(\pi, \tau)$ if

$$\Delta(v, \mu) + s_m(e) - s_m(\epsilon) - \tau = t_h. \tag{49}$$

This equation is derived by combining (15) and (17) and signifies a timing bottleneck due to a hold constraint between registers on e and ϵ.

The graph $T(\pi, \tau)$ is the union of $S(\pi, \tau)$ and $H(\pi, \tau)$ with all the edges in $H(\pi, \tau)$ inverted. This inversion is a mere convenience that enables the uniform handling of both the setup and the hold constraints. As it can be verified by bringing (17) into the form

$$-\Delta(v, \mu) + s_M(\epsilon) - s_m(e) \leq -t_h \tag{50}$$

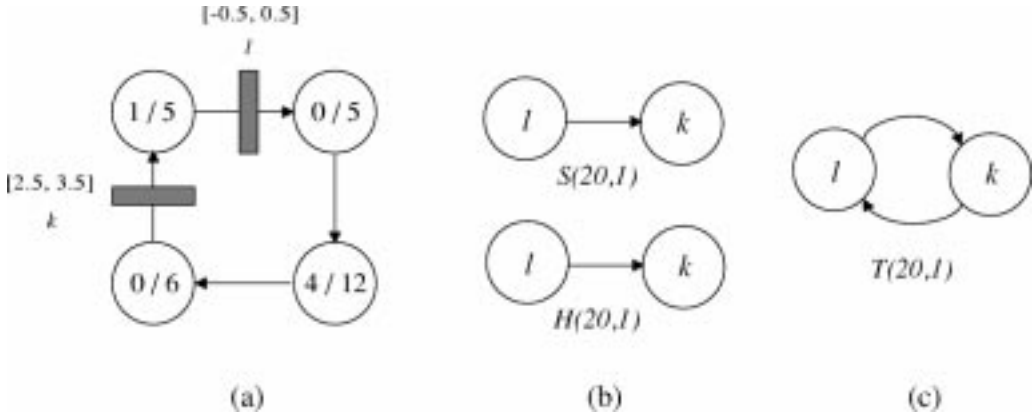

Fig. 5. Circuit graph and associated slack graphs.

```
SGR(G, s_M, s_m, π, τ)
 1  compute S(π,τ), H(π,τ), and T(π,τ)
 2  for each vertex e in a directed cycle c ∈ S(π,τ) such that w(e) > 1
 3      do shift a register from e onto a register-free edge in c
 4          return G_r
 5  for each vertex e in a directed cycle c ∈ T(π,τ)
 6      do if ∃x, v ∈ c such that (e,x) ∈ H(π,τ), (e,v) ∈ S(π,τ)
 7              then shift a register from e forward in G
 8                  return G_r
 9          if ∃x, v ∈ c such that (x,e) ∈ H(π,τ), (v,e) ∈ S(π,τ)
10              then shift a register from e backward in G
11                  return G_r
12  return FALSE
```

Fig. 6. Algorithm SGR for scheduling-guided retiming.

and subsequently comparing it with (16), from a mathematical standpoint the hold constraint from e to ϵ is equivalent to a setup constraint from ϵ to e with path delay $-\Delta(v,\mu)$ and setup time $-t_h$. Thus, when the direction of the hold edges is inverted in $T(\pi,\tau)$, timing bottlenecks due to combinations of tight setup and tight hold constraints appear as directed paths, and timing optimality can be verified by identifying a directed cycle in $T(\pi,\tau)$. A formal proof of this statement is given in Theorem 10.

Fig. 5 gives an example of how slack graphs are constructed. When tolerance is set to 1 tu, the original circuit in Fig. 5(a) can be correctly timed with a minimum clock period of 20 tu. The ranges $[s_m(e), s_M(e)]$ of the optimal clock schedules are indicated next to each register, assuming that both t_s and t_h are zero. Fig. 5(b) shows the corresponding slack graphs $S(20,1)$ and $H(20,1)$. The path from register l to register k in the original circuit satisfies both (48) and (49). An edge is therefore added between the corresponding vertices in $S(20,1)$ and $H(20,1)$. The graph $T(20,1)$ combining $S(20,1)$ and $H(20,1)$ is shown in Fig. 5(c). The directed cycle in this graph indicates that for the specified tolerance and register placement, the given clock schedule is optimum and achieves the minimum possible clock period.

Our algorithm SGR for scheduling-guided retiming is given in Fig. 6. The inputs to this procedure are a circuit graph G, clock schedules s_m and s_M, a clock period π, and a tolerance τ. The input graph G may be an original or a retimed circuit. If the procedure succeeds in identifying and eliminating a timing bottleneck in G by generating a retimed graph G_r, it returns G_r. Otherwise, it returns False. Initially, algorithm SGR constructs the slack graphs $S(\pi,\tau)$, $H(\pi,\tau)$, and $T(\pi,\tau)$. It then proceeds to detect timing bottlenecks by identifying directed cycles in $S(\pi,\tau)$ and $T(\pi,\tau)$. Once discovered, a bottleneck is removed by retiming the original input graph G. The **for** loop in lines 2–4 searches $S(\pi,\tau)$ for the special case of timing bottlenecks comprising only setup constraints. If an edge with register count greater than 1 participates in such a constraint, one of its registers is shifted to a register-free edge along the bottleneck. A tight setup constraint is thus removed, creating the potential for a shorter clock period or greater tolerance. The **for** loop in lines 5–11 searches $T(\pi,\tau)$ for timing bottlenecks comprising both setup and hold constraints. In lines 6–8, the procedure discovers a tight setup and a tight hold constraint that start from the same register. By shifting that register forward, it decreases the maximum delay of the long path and eliminates the corresponding setup constraint. It also broadens the range of clock

schedules that satisfy the hold constraint along the short path, thus creating opportunities for further clock period or tolerance optimization. In lines 9–11, a pair of tight setup and hold constraints leading to the same register is detected, and the register is shifted backward to relax them. If no register can be moved, line 12 reports the failure to build a new G_r.

The runtime of algorithm SGR is dominated by its three loops. Each loop iterates $O(E^2)$ times, since each slack graph has $O(E^2)$ edges. The runtime of each loop body execution depends on the implementations of the shifting transformations in lines 3, 7, and 10. A straightforward implementation of register shifting that propagates the perturbation to adjacent edges until no edge has a register deficit takes $O(E)$ time. In this case, the total runtime of each loop is $O(E^3)$ steps. More elaborate shifting schemes that also consider constraints on the register counts of specific edges require the solution of single-source shortest-paths problems with negative edge-weights. The construction of the three slack graphs in line 1 takes $O(E^2)$ time, since G may have $O(E)$ edges with nonzero register counts. In lines 2 and 5, the detection of all edges that participate in some directed cycle of $S(\pi,\tau)$ or $T(\pi,\tau)$ can be performed in $O(E^2)$ steps by running a breadth-first search that proceeds until it discovers $2 \cdot E$ levels. Algorithm SGR runs very fast in actual circuits because the number of vertices in the slack graphs equals to the number of registers in the circuit at most, which is usually far less than E.

B. Timing Bottleneck Elimination

In this section we prove that algorithm SGR correctly identifies timing bottlenecks in a circuit G, given a specified clock schedule, a clock period π, and a tolerance τ. We then prove that it correctly retimes G to remove a tight timing constraint that contributes to an identified bottleneck.

Algorithm SGR identifies timing bottlenecks by detecting directed cycles in the slack graph $T(\pi,\tau)$. The following theorem proves that each timing bottleneck in G does indeed correspond to a special directed cycle in $T(\pi,\tau)$. In addition to bottlenecks due to cyclic sequential paths in G, the theorem encompasses bottlenecks due to acyclic sequential paths from a primary input to a primary output, since each such path corresponds to a closed path in $T(\pi,\tau)$ that starts and ends with the vertex $e_{I/O}$. Furthermore, it also considers possible bottlenecks due to acyclic sequential paths between any two registers since both setup and hold constraints are considered.

Theorem 10: Let $G = \langle V, E, d, \delta, w\rangle$ be an edge-triggered circuit, and let π and τ be given reals. Moreover, let $s_m : E \to \mathbf{R}$ and $s_M : E \to \mathbf{R}$ be minimum and maximum clock schedules, respectively, such that G achieves a clock period π with tolerance τ. Then for the given placement of registers in G, π is the shortest clock period that clock scheduling can achieve with tolerance τ if and only if there exists a directed cycle in $T(\pi,\tau)$ which contains a tight setup constraint. □

Proof: ($\Rightarrow$) The proof is by contradiction. Specifically, we show that if the directed graph $T(\pi,\tau)$ has no cycle containing a tight setup constraint, then there exist clock schedules s'_m and s'_M such that G achieves a clock period $\pi' < \pi$ with tolerance τ, thus contradicting the minimality of π. To that end, we first describe a procedure for labeling the vertices in $T(\pi,\tau)$. We then rely on these labels to compute new clock schedules s'_m and s'_M that distribute timing slack among all tight timing constraints and yield a clock period $\pi' < \pi$ with tolerance τ.

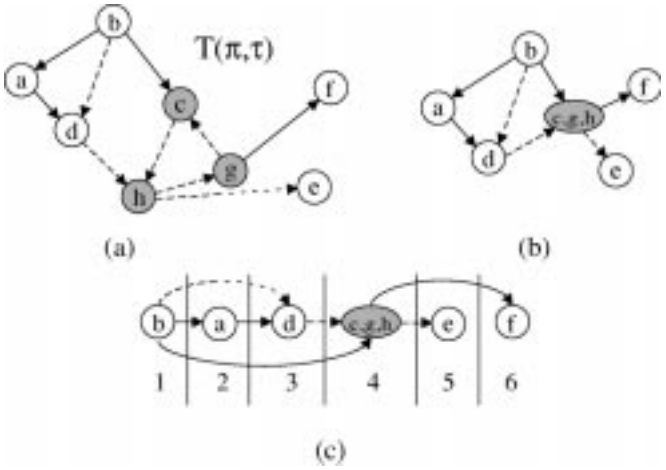

Fig. 7. (a) $T(\pi,\tau)$ with a single strongly connected component c, g, h from edges corresponding to hold constraints. (b) Construction of $T'(\pi,\tau)$. (c) Labeling of $T'(\pi,\tau)$.

The absence of tight setup constraints from all directed cycles in $T(\pi,\tau)$ allows us to compute a labeling of its vertices such that for each edge $e \to \epsilon$, we have

$$(e \to \epsilon) \in S(\pi,\tau) \Rightarrow l(\epsilon) \geq l(e) + 1 \tag{51}$$

$$(\epsilon \to e) \in H(\pi,\tau) \Rightarrow l(\epsilon) \geq l(e) \tag{52}$$

where $l(\epsilon)$ and $l(e)$ are the labels of ϵ and e, respectively. The labels l can be computed by performing a single-source longest-paths computation on a directed acyclic graph $T'(\pi,\tau)$. This graph is derived from $T(\pi,\tau)$ by grouping together into a single vertex all vertices that participate in the same strongly connected component formed by tight hold constraints. Fig. 7 shows the derivation of $T'(\pi,\tau)$ from a tight graph $T(\pi,\tau)$. The solid lines represent tight setup constraints, and the dashed lines represent tight hold constraints. The shaded nodes c, h, g form a single strongly connected component from edges corresponding to tight hold constraints and are grouped together in $T'(\pi,\tau)$. Since $T(\pi,\tau)$ has no cycle containing a tight setup constraint, the graph $T'(\pi,\tau)$ is acyclic. It is thus possible to assign to each vertex e' in $T'(\pi,\tau)$ the length $l'(e')$ of the longest path from a source vertex v_s to e' [4]. Fig. 7(c) shows the derivation of labels on $T'(\pi,\tau)$. For each vertex $e \in T$ that is also present in T', we have $l(e) = l'(e)$. For all vertices $e \in T$ in the same strongly connected component represented by $e' \in T'$, we have $l(e) = l'(e')$. In other words, all vertices in the same strongly connected component in T have the same label.

We now define the *slack* $\sigma(e)$ of a vertex e in $T(\pi,\tau)$ as the minimum timing margin over all hold constraints between e and all upstream registers in G but not in the same strongly connected component and over all setup constraints between e and downstream registers in G. Intuitively, this slack is the largest value by which the skew between e and any ϵ can increase without violating any timing constraint except for hold violations within the same strongly connected component. If some of these timing constraints is tight, then $\sigma(e) = 0$. In mathematical

terms, for an edge $u \xrightarrow{e} v$ in E with $w(e) \geq 1$, the margin $\sigma(e)$ is defined as follows:

$$\begin{aligned}\sigma(e) = \min_{\epsilon \in E} \{ \{ & \Delta(\nu, u) + s_m(\epsilon) - s_m(e) - \tau - t_h : \\ & \mu \xrightarrow{\epsilon} \nu \rightsquigarrow u \xrightarrow{e} v \in G, \quad w(\epsilon) \geq 1, W(\nu, u) = 0, \\ & l(e) \neq l(\epsilon) \} \\ \cup \{ & \pi - t_s - (D(v, \mu) \\ & + s_m(e) + \tau - s_m(\epsilon)) : \\ & u \xrightarrow{e} v \rightsquigarrow \mu \xrightarrow{\epsilon} \nu \in G, w(\epsilon) \geq 1, \\ & W(v, \mu) = 0 \} \}. \end{aligned} \tag{53}$$

The slack σ of the tight graph $T(\pi, \tau)$ is defined as $\sigma = \min \{\sigma(e) : e \in E, \sigma(e) > 0\}$. Based on the acyclicity of $T'(\pi, \tau)$, we can prove that $\sigma > 0$. Indeed, since $T'(\pi, \tau)$ is acyclic, there exists a vertex $e' \in T'$, and a corresponding vertex $e \in T$, with no outgoing edge. Therefore, e has no tight setup constraint with any downstream register in G. Moreover, e has no tight hold constraint with any upstream register in G except for those within a strongly connected component. From the definition of $\sigma(e)$ it follows that $\sigma(e) > 0$, and therefore $\sigma > 0$.

We now rely on the labeling l and the slack σ to compute new clock schedules s'_m and s'_M. For each vertex $e \in T$, the new clock schedule $s'_m(e)$ is defined as

$$s'_m(e) = s_m(e) + \frac{\sigma \cdot l(e)}{l}$$

where $l = 1 + \max \{l(e) : e \in T(\pi, \tau)\}$. Furthermore, the clock schedule $s'_M(e)$ is defined as

$$s'_M(e) = s'_m(e) + \tau.$$

The following straightforward case analysis shows that the new clock schedules s'_m and s'_M satisfy the timing constraints in G for clock period $\pi - \sigma/l$ and tolerance τ. Let $u \xrightarrow{e} v \rightsquigarrow \mu \xrightarrow{\epsilon} \nu$ be a path in G such that $w(e) \geq 1$, $w(\epsilon) \geq 1$, and $W(v, \mu) = 0$. By relying on the definitions of s'_m and s'_M, the setup constraint along this path can be written as follows:

$$\begin{aligned} D(v, \mu) + s'_M(e) - s'_m(\epsilon) &= D(v, \mu) + (s'_m(e) + \tau) - s'_m(\epsilon) \\ &= D(v, \mu) + \left(s_m(e) + \frac{\sigma \cdot l(e)}{l} \right) \\ &\quad + \tau - \left(s_m(\epsilon) + \frac{\sigma \cdot l(\epsilon)}{l} \right) \\ &= D(v, \mu) + (s_m(e) + \tau) - s_m(\epsilon) \\ &\quad + \left(\frac{\sigma \cdot l(e)}{l} - \frac{\sigma \cdot l(\epsilon)}{l} \right) \\ &= D(v, \mu) + (s_m(e) + \tau) - s_m(\epsilon) \\ &\quad + \frac{\sigma \cdot (l(e) - l(\epsilon))}{l}. \end{aligned} \tag{54}$$

The schedules s_m and s_M satisfy the setup constraint between e and ϵ with π and τ. If this constraint is not tight, then $e \rightarrow \epsilon \notin T'(\pi, \tau)$. In this case, a straightforward algebraic manipulation yields

$$\begin{aligned} D(v, \mu) + (s_m(e) + \tau) - s_m(\epsilon) + \frac{\sigma \cdot (l(e) - l(\epsilon))}{l} & \\ \leq (\pi - t_s - \sigma) + \frac{\sigma \cdot (l(e) - l(\epsilon))}{l} & \\ = \pi - t_s + \frac{\sigma \cdot (l(e) - l(\epsilon) - l)}{l} & \\ \leq \pi - t_s - \frac{\sigma}{l} & \\ = \left(\pi - \frac{\sigma}{l} \right) - t_s. & \end{aligned} \tag{55}$$

If the setup constraint between e and ϵ is tight, it follows that $e \rightarrow \epsilon \in T'(\pi, \tau)$ and $l(e) \geq l(\epsilon) + 1$. Therefore

$$\begin{aligned} D(v, \mu) + (s_m(e) + \tau) - s_m(\epsilon) + \frac{\sigma \cdot (l(e) - l(\epsilon))}{l} & \\ \leq D(v, \mu) + (s_m(e) + \tau) - s_m(\epsilon) - \frac{\sigma}{l} & \\ \leq (\pi - t_s) - \frac{\sigma}{l} & \\ \leq \left(\pi - \frac{\sigma}{l} \right) - t_s. & \end{aligned} \tag{56}$$

From (54)—(56), and the definitions of s'_m and s'_M, it follows that s'_m and s'_M satisfy the setup constraint from e to ϵ with clock period $\pi - \sigma/l$ and tolerance τ. A similar case analysis shows that s'_m and s'_M satisfy the hold constraint between e and ϵ. Moreover, in the case that e and ϵ belong to the same connected component in $T(\pi, \tau)$, their clock schedules change by the same amount $\sigma \cdot l(e)/l$, and the hold constraints among them still hold tight. Therefore, there exists a clock schedule that achieves a clock period $\pi' < \pi$ with tolerance τ.

($\Leftarrow$) We now prove that if there exists a directed cycle c in $T(\pi, \tau)$ which contains at least one tight setup constraint, then π is the shortest clock period that clock scheduling can achieve with tolerance τ.

Consider an edge $e \rightarrow \epsilon$ in c that encodes a setup constraint on a combinational path from a register on edge $u \xrightarrow{e} v$ to a register on edge $\mu \xrightarrow{\epsilon} \nu$ in G. From (15) and (16), this constraint yields

$$D(v, \mu) + s_m(e) + \tau - s_m(\epsilon) \leq \pi - t_s. \tag{57}$$

Alternatively, if $e \rightarrow \epsilon$ encodes a hold constraint from $\mu \xrightarrow{\epsilon} \nu$ to $u \xrightarrow{e} v$, then (15) and (17) yield

$$\Delta(\nu, u) + s_m(\epsilon) - s_m(e) - \tau \geq t_h. \tag{58}$$

By rearranging the terms in (57) and (58), we obtain the equivalent inequalities

$$s_m(e) - s_m(\epsilon) + \tau \leq \pi - t_s - D(v, \mu) \tag{59}$$

$$s_m(e) - s_m(\epsilon) + \tau \leq \Delta(\nu, u) - t_h. \tag{60}$$

Adding up (59) and (60) along c, we obtain

$$\begin{aligned} \sum_{e \rightarrow \epsilon \in c} (s_m(e) - s_m(\epsilon) + \tau) \leq & \sum_{e \rightarrow \epsilon \in c_s} (\pi - t_s - D(v, \mu)) \\ & + \sum_{e \rightarrow \epsilon \in c_h} (\Delta(\nu, u) - t_h) \end{aligned}$$

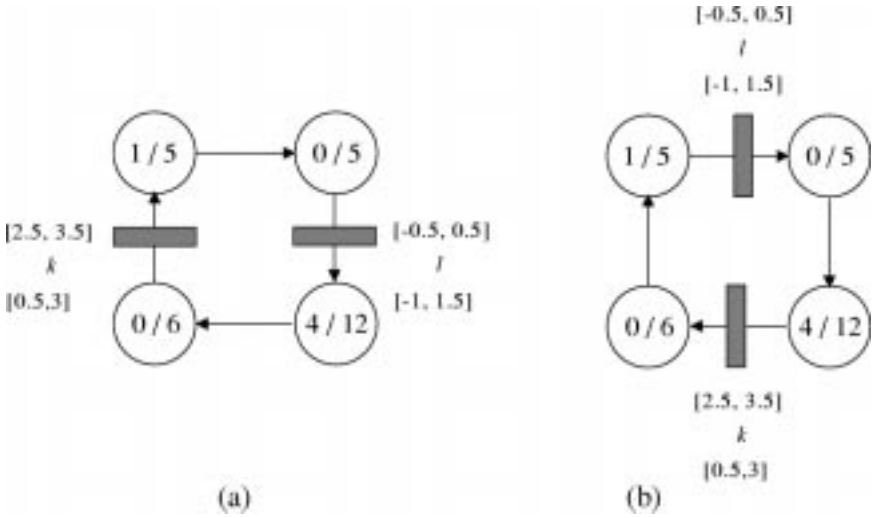

Fig. 8. Result of register relocation for the original circuit in Fig. 5. (a) Register l is shifted forward. (b) Register k is shifted backward.

where c_s and c_h denote the edges in c that correspond to setup and hold constraints, respectively. The summation on the left-hand side of this inequality telescopes, yielding

$$|c| \cdot \tau \leq \sum_{e \rightarrow \epsilon \in c_s} (\pi - t_s - D(v,\mu)) + \sum_{e \rightarrow \epsilon \in c_h} (\Delta(\nu,u) - t_h) \tag{61}$$

where $|c|$ denotes the edge count of the set. Since there exists at least one tight setup constraint, we can now obtain the following lower bound on the clock period π by rearranging (61) as follows:

$$\pi \geq \frac{|c|}{|c_s|} \cdot \tau + \frac{1}{|c_s|} \cdot \sum_{e \rightarrow \epsilon \in c_s} (t_s + D(v,\mu)) + \frac{1}{|c_s|} \cdot \sum_{e \rightarrow \epsilon \in c_h} (t_h - \Delta(\nu,u)) \tag{62}$$

where $|c_s|$ denotes the edge count of the sets. The right-hand side of (62) gives a lower bound for π that is independent of the clock schedules s_m and s_M. If the choice of s_m and s_M does induce the directed cycle c in $T(\pi,\tau)$, (57) and (58) are satisfied with equality along c. Propagating this equality all the way to (62), we infer that the lower bound is attained. Therefore, the associated clock period π is the shortest one achievable with tolerance τ. □

Theorem 10 implies that whenever $T(\pi,\tau)$ has a directed cycle which contains at least a tight setup constraint, the clock period of a circuit cannot be improved by clock scheduling without reducing the tolerance to clock delay variations. Retiming can be used to "break" such cycles by relocating the registers along the critical paths of the circuit. The following theorem shows that the register relocations in lines 3, 7, and 10 of algorithm SGR break a directed cycle in $T(\pi,\tau)$.

Theorem 11: Each execution of lines 3, 7, and 10 in algorithm SGR eliminates a tight timing constraint associated with the timing bottleneck detected in lines 2, 6, and 9, respectively.□

Proof: When line 3 executes, a register is placed on a register-free edge x, between the edges e and ϵ that give rise to a tight setup constraint. Since $e \rightsquigarrow x$ is a subpath of $e \rightsquigarrow \epsilon$, it follows that $d(e \rightsquigarrow x) < d(e \rightsquigarrow \epsilon)$ and the tight setup constraint between e and ϵ has thus been removed.

When line 7 executes, the delay along the setup path that ends at v is reduced. The corresponding tight setup constraint is thus removed. Similarly, when line 10 executes, the setup path starting at v becomes shorter and the corresponding tight constraint is eliminated. □

The retiming transformations in lines 7 and 10 create the potential for further clock period and tolerance optimization. For example, consider the forward relocation of a register k across a vertex u under the conditions of the predicate in the **if** statement of line 6. In the transformed circuit, the maximum delays of the paths that end at v and x decrease by $d(u)$, whereas the minimum delays decrease by $\delta(u)$. Thus, clock scheduling can redistribute a slack of $d(u)$ among other tight setup constraints downstream from v. Moreover, it can redistribute a slack of $d(u) - \delta(u)$ among other tight hold constraints downstream from x. It thus allows for the timing constraints to be met with a shorter clock period or greater tolerance.

The reader should note that even though they promote optimization, the retiming transformations of algorithm SGR do not guarantee the improvement of the clock period or the tolerance in the resulting circuit. When a register is shifted forward to remove a tight setup constraint with a downstream register, for example, a setup violation with an upstream register may result. A careful implementation of the relocation steps in lines 7 and 10 of algorithm SGR should consider the violation of constraints that currently hold before deciding to move a register.

Fig. 8 shows how register relocation can result in a faster or more delay-tolerant circuit. By examining the slack graphs in Fig. 5 for $\pi = 20$ tu and $\tau = 1$ tu, we infer that vertex l satisfies the condition in line 6 of algorithm SGR and can thus be shifted forward. Furthermore, vertex k satisfies the condition of line 9 and can be shifted backward. The circuits resulting from each of these possible transformations are shown in Fig. 8. With tolerance remaining at $\tau = 1$ tu, clock scheduling results in a shorter clock period for both circuits. The retimed circuit in Fig. 8(a) achieves $\pi = 16$ tu, whereas the circuit in Fig. 8(b)

```
RSMINP(G, τ)
 1 G_r = RET(G)    ▷ retime for minimum clock period
 2 (π, s_M, s_m) = SCHEDMINP(G_r, τ)    ▷ schedule for minimum clock period with tolerance τ
 3 while SGR(G_r, s_M, s_m, π, τ) = NOT FALSE
 4       (π', s_M, s_m) = SCHEDMINP(G_r, τ)
 5       if π' < π
 6         then π = π'
 7         else if no improvement for n consecutive iterations
 8                 then break
 9 return (G_r, π, s_M, s_m)
```

Fig. 9. Procedure RSMINP for simultaneous retiming and clock scheduling to minimize clock period.

```
RSMAXT(G, π)
 1 set G_r = RET(G)    ▷ retime for minimum clock period
 2 (τ, s_M, s_m) = SCHEDMAXT(G_r, π)    ▷ schedule for maximum tolerance with clock period π
 3 while SGR(G_r, s_M, s_m, π, τ) = success
 4       (τ', s_M, s_m) = SCHEDMAXT(G_r, π)
 5       if τ' > τ
 6         then τ = τ'
 7         else if no improvement for n consecutive iterations
 8                 then break
 9 return (G_r, τ, s_M, s_m)
```

Fig. 10. Procedure RSMAXT for simultaneous retiming and clock scheduling to maximize clock delay tolerance.

achieves $\pi = 15$ tu. The intervals above the register names specify the ranges of the skewed clock arrival times. Alternatively, with the clock period fixed at $\pi = 20$ tu, clock scheduling can increase the delay tolerance of both retimed circuits to $\tau = 2.5$ tu. The ranges of the clock arrival times are given below the corresponding register names.

C. Period and Tolerance Optimization

This section describes two heuristic procedures for the optimization versions of the retiming and clock scheduling problem. These procedures are not guaranteed to find the optimum solution to their respective problems. As shown in Section VI, however, they outperform the MILP-based binary search algorithms by achieving a good tradeoff between program runtime and circuit performance improvement.

Fig. 9 gives pseudocode for algorithm RSMINP that performs simultaneous retiming and clock scheduling to minimize clock period. Given an edge-triggered circuit G and a target delay tolerance τ, algorithm RSMINP returns a retimed circuit G_r, an optimal clock period π, and clock schedules s_M and s_m such that G_r achieves π with the given tolerance τ. The main idea in this heuristic is to iteratively perform clock scheduling followed by scheduling guided retiming. In line 1, G_r is initialized to a retimed version of the input circuit G that operates with minimum clock period under zero clock skew. The registers in this circuit have been relocated so that maximum delays are distributed evenly along its paths. In line 2, algorithm SCHEDMINP solves the clock scheduling optimization problem for G_r with tolerance τ. Based on Section III, this computation can be performed by binary searching the tolerance domain with a single-source shortest-paths algorithm. The **while** loop in lines 3–8 iterates until the clock period cannot be reduced any further for a predefined number of iterations. Each execution of line 3 invokes algorithm SGR with the objective to relax the tight setup and hold constraints for the clock schedules s_m, s_M and the corresponding optimal clock period π returned by algorithm SCHEDMINP. In line 4, clock scheduling is performed on the retimed circuit G_r returned by algorithm SGR. Lines 5–6 update the best solution found so far. The loop exits when no new G_r is found or the clock period has not improved for a given number of times.

Fig. 10 gives a pseudocode description of algorithm RSMAXT for maximum-tolerance retiming and clock scheduling. Similar to algorithm RSMINP, this procedure is based on scheduling-guided retiming to create a sequence of retimed circuits and converge toward an optimal tolerance. Given an edge-triggered circuit G and a target clock period π, this procedure returns a retimed circuit G_r, an optimal clock tolerance τ, and corresponding clock schedules s_M and s_m such that G_r achieves the given clock period π with maximum tolerance τ. Procedure

SCHEDMAXT in line 2 performs a binary search to compute the maximal delay tolerance that G_r can achieve with clock period π. Next, the **while** loop in lines 3–8 iteratively retimes G_r until tolerance cannot be improved any further or no new retiming can be found.

VI. EXPERIMENTAL RESULTS

In this section, we discuss extensive experimental results from the application of retiming and clock scheduling to the optimization of the LGSynth93 and ISCAS89 benchmark circuits. The statistics of the circuits in our test suite are given in Table I. Section VI-A presents the results we obtained for the clock period minimization problem and Section IV-B gives our results for the tolerance maximization problem. In both cases, the combined application of the two optimizations resulted in significant improvements over the separate application of either of the two. We also compared the results of our heuristic with those of sequential retiming and clock scheduling. For the circuits in our test suite, our heuristic achieved marginally better speedups than retiming for maximum speed followed by scheduling for maximum speed. With respect to tolerance maximization, however, our heuristic achieved 12% higher tolerance, on the average, than sequential retiming and clock scheduling. Our proposed heuristic approach was orders of magnitude faster than the exact MILP-based optimizer with little sacrifice in accuracy. Our software was developed in C and ran on a Pentium Pro II with 128MB of main memory.

A. Clock Period Minimization

Simultaneous retiming and clock scheduling was applied to improve the clock period of LGSynth93 and ISCAS89 benchmark circuits. We first evaluated the extent to which the combination of retiming and clock scheduling is better than either optimization. We subsequently compared our heuristic with the sequential application of the two optimizations.

The following experimental methodology was used to compare the joint retiming and clock scheduling with either of the two approaches. First, for each circuit, the shortest clock period under zero skew π° was computed. Subsequently, each benchmark circuit was optimized by retiming, clock scheduling, and simultaneous retiming and clock scheduling to achieve the minimum possible clock periods π_r°, π_s°, and π_{rs}°, respectively. All clock periods were computed with zero tolerance. Simultaneous retiming and clock scheduling was performed using the heuristic procedure RSMINP.

Fig. 11 gives the relative improvement $\min\{\pi_r^\circ, \pi_s^\circ\}/\pi_{rs}^\circ - 1$ achieved by heuristic retiming and clock scheduling over the best result obtained by either retiming or clock scheduling. Our results show that combined retiming and clock scheduling does improve over the separate application of the two optimizations. Improvements greater than 10% are achieved in only five cases, however, and most of the circuits show no improvement. This relatively unimpressive performance can be attributed to the structure of our original test circuits. Most of these circuits are finite state machines with single-register loops or with input/output combinational paths. Thus, they are not amenable to performance optimization by retiming or clock scheduling.

TABLE I
STATISTICS OF TEST CIRCUITS

number	circuit	nodes	edges	π^o
1	bbara	72	199	5.30
2	bbtas	44	100	4.08
3	beecount	56	132	6.80
4	daio	26	39	3.23
5	dk14	84	253	6.30
6	dk15	61	166	5.87
7	dk16	139	586	9.88
8	dk17	53	127	5.28
9	dk27	34	64	4.11
10	dk512	53	121	4.55
11	ex1	222	916	16.11
12	ex4	88	225	6.43
13	ex6	121	398	8.22
14	lion	24	42	3.49
15	opus	93	264	9.37
16	s1196	863	1357	21.21
17	s1238	854	1401	19.96
18	s1423	1105	1586	48.28
19	s208.1	139	217	5.68
20	s208	54	129	5.13
21	s298	166	297	6.18
22	s344	272	392	15.34
23	s349	278	401	15.51
24	s382	302	456	12.29
25	s386	190	385	10.12
26	s400	323	487	12.04
27	s420.1	362	528	10.57
28	s420	65	196	6.24
29	s444	260	437	8.06
30	s526	363	610	10.05
31	s526n	325	582	8.57
32	s641	412	587	38.67
33	s713	478	691	46.26
34	s838.1	734	1076	13.48
35	s838	722	1004	46.99
36	s953	784	1155	15.43
37	sbc	1249	1963	14.67
38	sse	110	355	8.15
39	tav	40	73	3.65

To explore the potential of simultaneous retiming and clock scheduling on a test suite that is more representative of typical circuits, we slightly modified the original benchmark circuits by inserting an additional register into each single-register loop. Specifically, each register whose output was connected to its own input via a purely combinational path was replaced by a pair of back-to-back registers. This modification does not change the clock period of the unoptimized circuits. It nevertheless creates more potential for retiming and clock scheduling to improve circuit performance.

The optimal clock periods π_r^m, π_s^m, and π_{rs}^m were computed on the modified test circuits. Relative improvements $\min\{\pi_r^m, \pi_s^m\}/\pi_{rs}^m - 1$ are given in Fig. 12. Simultaneous retiming and clock scheduling improved the clock period for more than half of the modified test suite. For one-third of

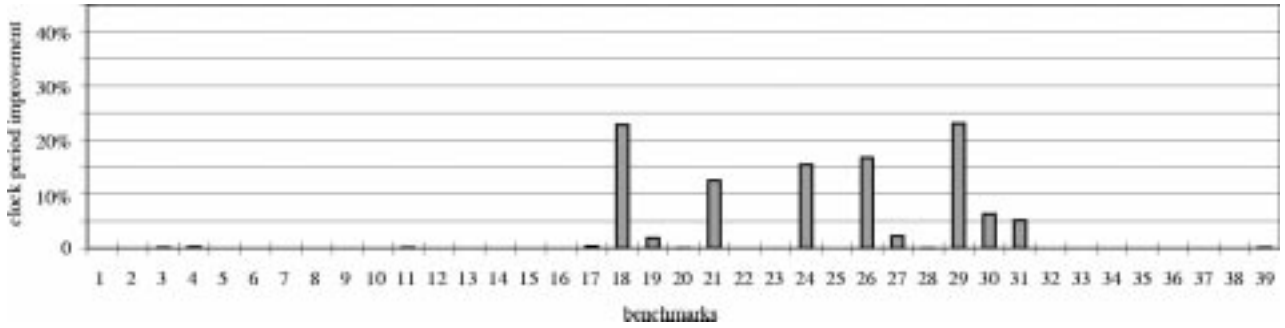

Fig. 11. Relative clock period improvements with original circuits over retiming or clock scheduling.

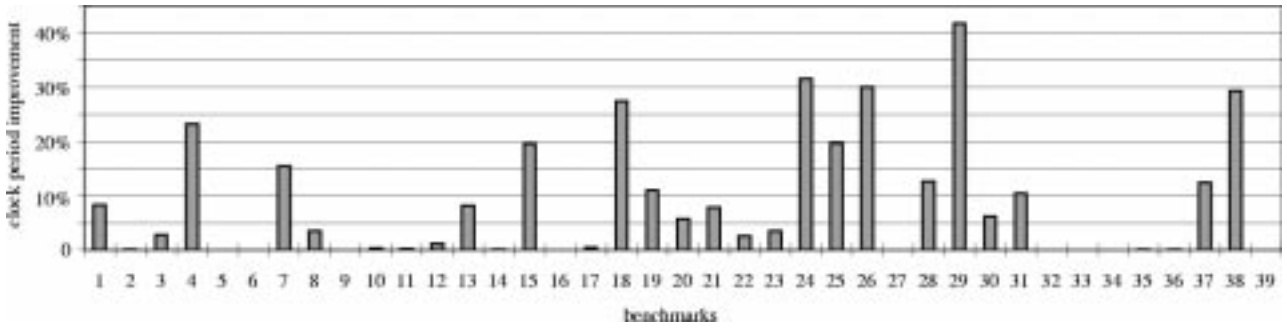

Fig. 12. Relative clock period improvements with modified circuits over retiming or clock scheduling.

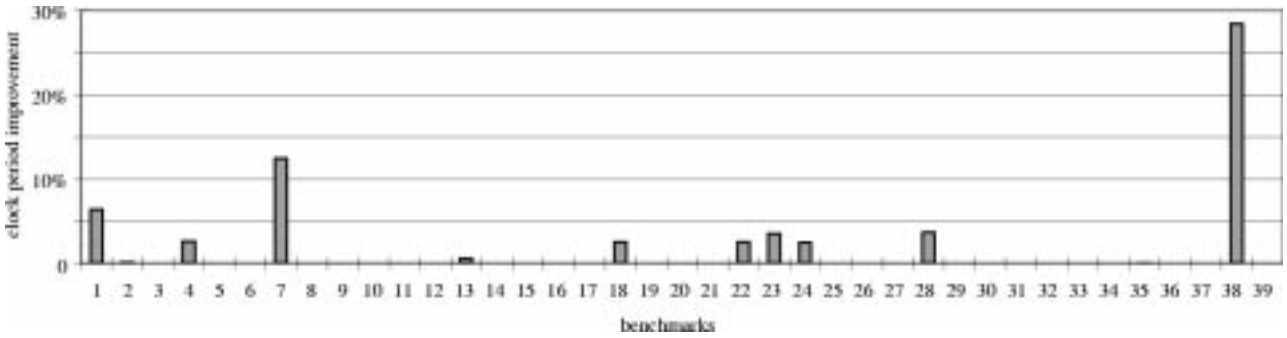

Fig. 13. Relative clock period improvements with modified circuits over the sequential retiming and clock scheduling.

the test circuits, improvements exceeded 10%. In four cases, relative improvements exceeded 30%. The average improvement was 8%. These results show the superiority of combined retiming and clock scheduling to the separate application of the two optimizations.

In addition to the separate application of retiming or clock scheduling, we compared our heuristic algorithm RSMINP with a relatively straightforward sequential heuristic. Specifically, for all modified circuits, we computed the minimum possible clock periods $\pi^m_{r;s}$ that can be obtained by performing minimum-period retiming followed by minimum-period scheduling. As with our heuristic, the tolerance was set to zero. Although this heuristic may fail to obtain the shortest possible clock period, as we demonstrated in Section I, it proved to be quite effective on our modified circuits with respect to clock period minimization. Fig. 13 shows the relative improvements obtained by our heuristic over the sequential heuristic on the modified circuits. Noticeable improvements were achieved for only a quarter of the circuits in the test suite. Average and maximum improvements were 1.6% and 28%, respectively.

TABLE II
CLOCK PERIOD MINIMIZATION WITH ALGORITHM RSMINP AND AN EXACT MILP-BASED BRANCH-AND-BOUND ALGORITHM

Circuit	RSMINP		MILP		$\Delta\pi^m_{rs}$ (%)
	π^m_{rs}	CPU (sec)	π^m_{rs}	CPU (sec)	
daio	0.63	0.2	0.63	1902.2	1
dk27	2.01	0.2	2.01	63306.1	0
beecount	3.07	1.9	3.07	t/o	0
dk15	3.35	1.7	3.35	2271.5	0
dk17	2.63	1.8	2.62	t/o	0
lion	1.37	0.1	1.37	141.8	0
s208	1.92	2.6	1.92	t/o	0
tav	1.72	0.3	1.72	3.8	0
s838.1	7.13	1001.1	7.13	t/o	0
s1423	28.65	35125.0	39.20	t/o	-37

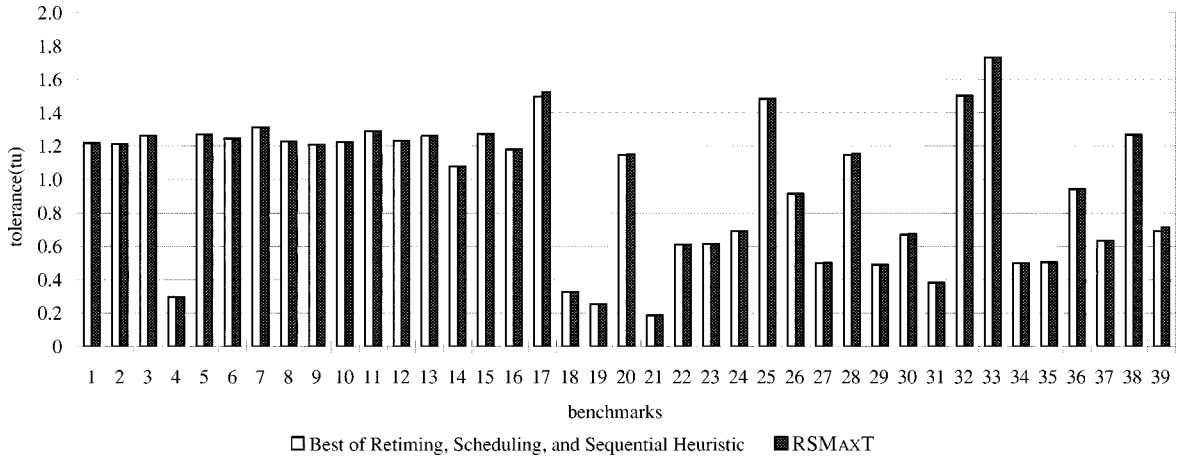

Fig. 14. Optimal tolerance with $\pi = \pi^{\circ}$.

The runtime of Procedure RSMINP was always longer than that of the sequential heuristic, since RSMINP uses the sequential heuristic as a preprocessing step. If RSMINP could not find a better solution than the sequential heuristic, it terminated quickly and achieved comparable runtimes. Whenever Procedure RSMINP computed better solutions, however, its runtime could be several times longer then the sequential heuristic.

To evaluate the relative speed and efficiency of our retiming and clock scheduling heuristic, we independently developed a MILP-based branch-and-bound solver. Table II compares the runtimes and output clock periods of the two programs for a subset of our test suite. In general, the CPU requirements of the MILP-based optimizer grow very fast, due to the high computational complexity of mixed-integer linear programming. With a 48-hour timeout, the MILP-solver runs out of time on most circuits, without having discovered a better solution than the heuristic scheme. The last column of Table II shows the relative clock period improvement achieved by the MILP-based solver over our heuristic. Except for daio, the fastest circuit computed by the heuristic is as good as that of the MILP solver. For daio, the heuristic solution comes within 1% of the MILP solver. In the case of s1423, the heuristic computes a better solution, because the exact solver reaches its timeout limit before discovering a better solution. Whenever the exact solver terminates with the optimal answer, it is one to five orders of magnitude slower than the heuristic.

In all the experiments, gate delays were derived from the widely used linear delay formula $a + b \cdot fanout$. For each gate u, we set $d(u) = a + b \cdot (fanout + rand)$ and $\delta(u) = a + b \cdot (fanout - rand)$, where a and b were parameters obtained from the library iwls93.mis2lib in the LGSynth93 benchmark. The parameter $rand$ was a random number uniformly distributed in the interval [0,1). The relatively small range of this parameter resulted in small variations between the maximum and minimum gate delays, thus limiting the effectiveness of combined retiming and clock scheduling over the sequential application of the two heuristics. The delays d and δ were computed once and then were kept constant throughout the retiming process. Although fanout can change in zero-skew retiming, when clock skew is not zero, registers can only be merged when they have the same skew, an event that occurs very rarely. In this paper, we did not consider register sharing, and therefore the fanout did not change during retiming.

B. Delay Tolerance Maximization

In addition to clock period optimization, we applied retiming and clock scheduling to our modified benchmark circuits to maximize their tolerance to clock delay variations. In our experiments, target clock periods ranged from $0.65 \times \pi^{\circ}$ to π°. For each clock period, the test circuits were optimized by retiming, clock scheduling, sequential retiming and clock scheduling, and Procedure RSMAXT to achieve the maximum tolerance τ_r^m, τ_s^m, $\tau_{r;s}^m$, and τ_{rs}^m, respectively. The sequential heuristic performed minimal-period retiming followed by maximal tolerance scheduling with the given target clock period.

Generally speaking, retiming had little effect on tolerance improvement in our experiments and was outperformed by clock scheduling for most circuits. The sequential heuristic achieved the highest tolerance among τ_r^m, τ_s^m, and $\tau_{r;s}^m$. As shown in Figs. 14–18, for most target clock periods and most test circuits, our heuristic RSMAXT achieved higher tolerance than the best achievable by the other three optimization methods. For $\pi = \pi^{\circ}$, clock scheduling performs as well as the combined application of retiming and clock scheduling in almost all cases. When the target clock period is relatively long, each circuit has only a few critical paths and abundant timing slack that can be distributed with no need for register relocation. It is thus possible for clock scheduling to set the clock arrival times near the middle of their respective permissible ranges, thus achieving the same tolerance as the combination of retiming and clock scheduling. As the target clock period decreases, our retiming and clock scheduling heuristic outperforms the other three alternatives by an increasingly higher margin. Moreover, it is capable of achieving shorter clock periods than the alternatives. Fig. 19 gives the fraction of test circuits for which our heuristic and its three alternatives can achieve correct timing. For $\pi = 0.65 \times \pi^0$, Procedure RSMAXT can meet the required timing for about 20% more circuits than the combination of the other three methods.

For each circuit and each target clock period π, we calculated the normalized tolerance τ_{rs}^m/π achieved by our heuristic RSMAXT and the best of the other optimizations

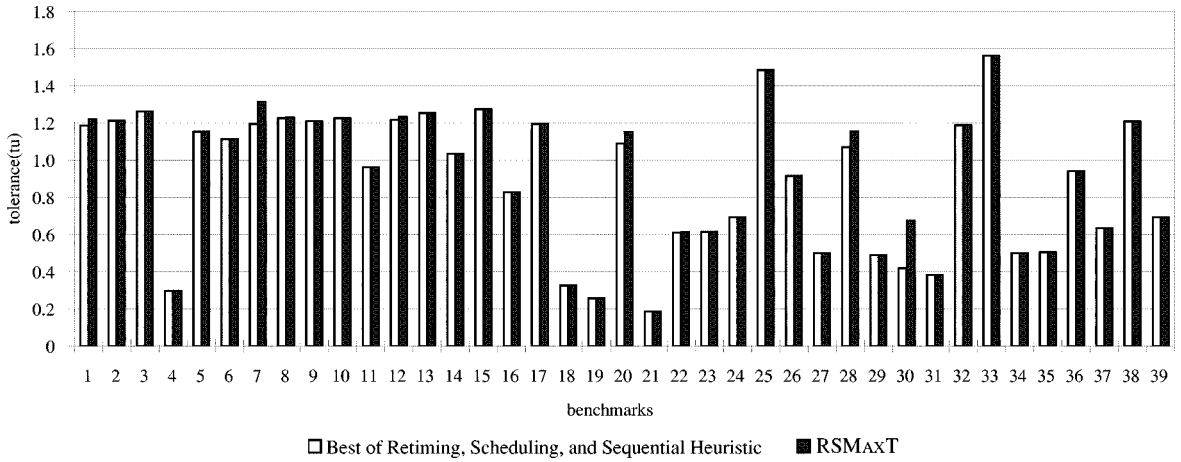

Fig. 15. Optimal tolerance with $\pi = 0.95 \times \pi^{\circ}$.

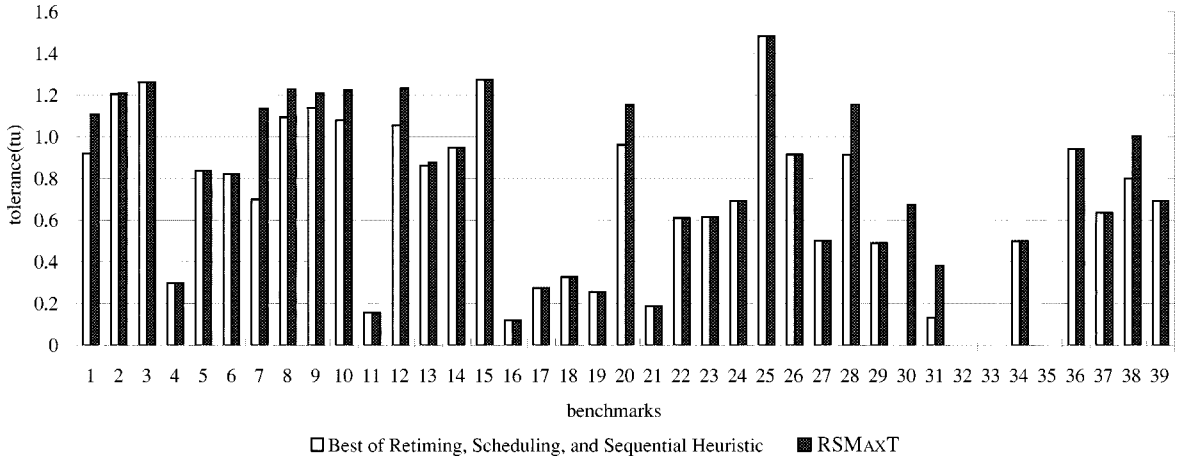

Fig. 16. Optimal tolerance with $\pi = 0.85 \times \pi^{\circ}$.

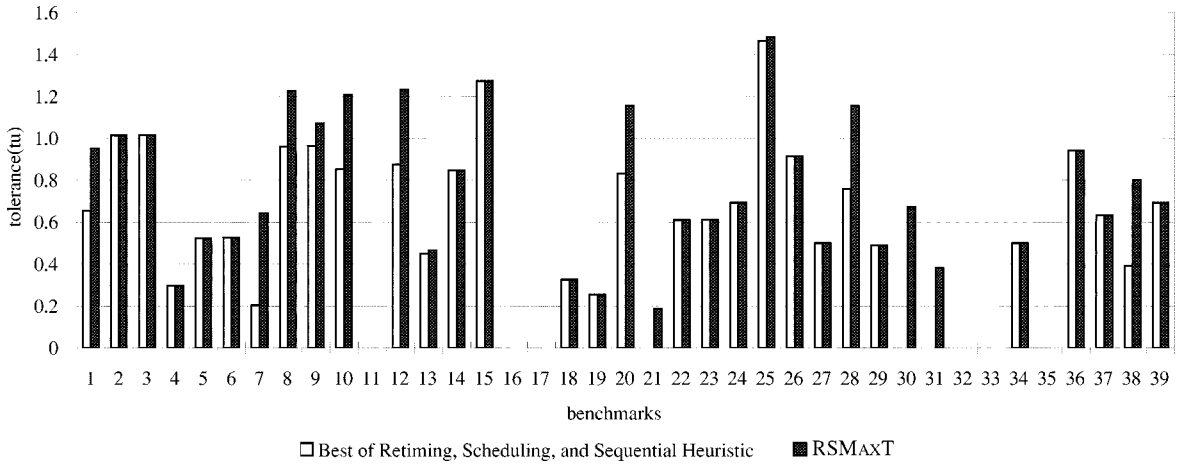

Fig. 17. Optimal tolerance with $\pi = 0.75 \times \pi^{\circ}$.

$\max\{\tau_r^m, \tau_s^m, \tau_{r;s}^m\}/\pi$. Whenever the target clock period could not be achieved, we set tolerance to zero. Fig. 20 shows the average tolerance improvements with respect to the target clock period. As the target π decreases, average relative improvements increase monotonically, exceeding 25% at $\pi = 0.65 \times \pi^0$ and averaging 12% across the entire range. Our combined retiming and clock scheduling methodology is thus most promising for high-speed circuit design with a tight timing budget.

Table III compares the efficiency and effectiveness of the heuristic solver and the MILP-based solver for a target clock of $0.75 \times \pi^{\circ}$. With a timeout limit set to 48 hours, the MILP solver improves the tolerance of only a small number of circuits from our test suite. For every circuit omitted from the table, the MILP

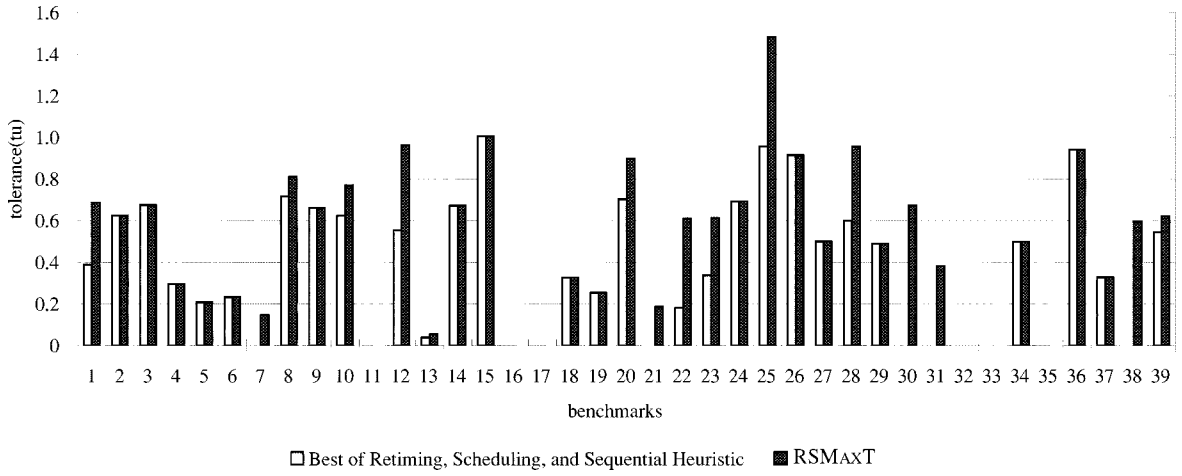

Fig. 18. Optimal tolerance with $\pi = 0.65 \times \pi^{\circ}$.

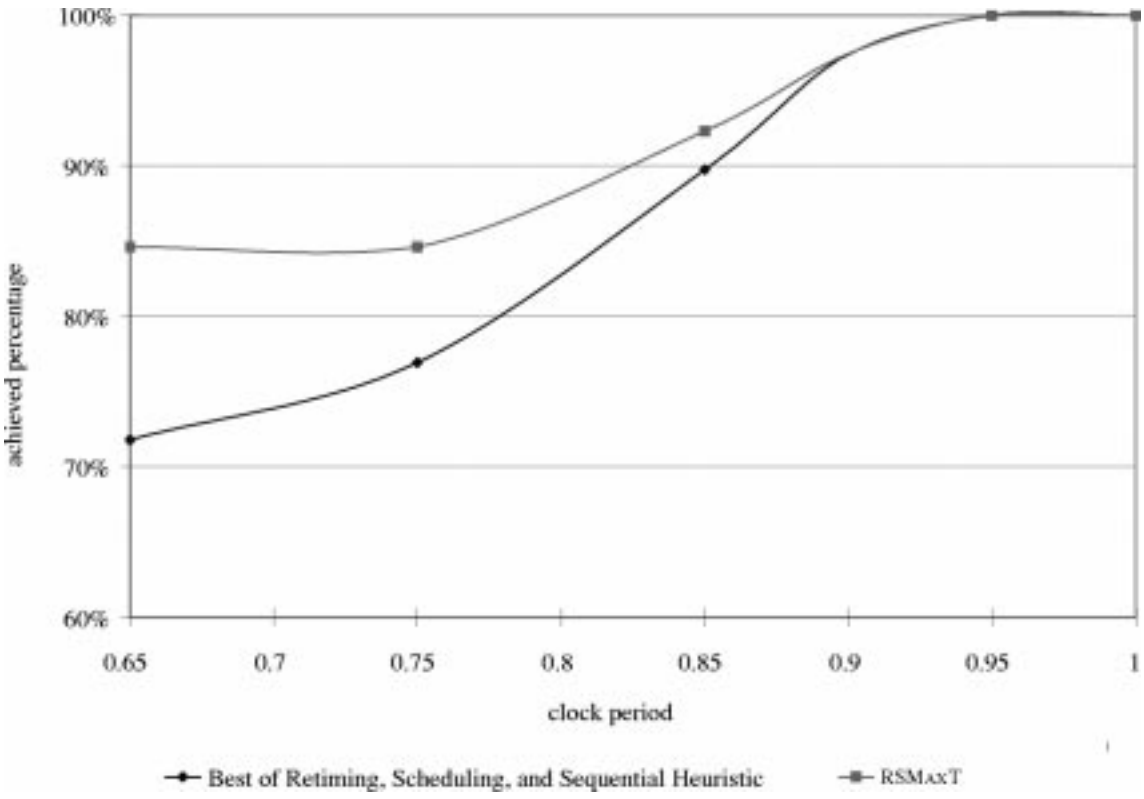

Fig. 19. Percentages of circuits that achieve correct timing under different target periods.

solver reaches its timeout without any improvement in circuit tolerance. Whenever both algorithms yield improvements, the heuristic is one to five orders of magnitude faster than the MILP solver. The relative effectiveness of the MILP-based algorithm over the heuristic approach is shown in the sixth column of the table. For all circuits but lion, the heuristic outperforms the MILP-based algorithm.

VII. Conclusion

This paper explores the application of simultaneous retiming and clock scheduling for maximizing the operating speed or the delay-variation tolerance of synchronous circuits with edge-triggered registers. Our work encompasses delay variations caused by process parameter variations, temperature fluctuations, and power supply variations. It is also applicable to the case of variations in clock signal delays due to clock gating or other data-dependent hardware mechanisms.

In the context of setup and hold constraints, we show that the combined optimization can result in faster or more tolerant circuits than when either of the two optimizations is applied separately or sequentially. We give a precise mixed-integer linear programming formulation for the basic problem of retiming and clock scheduling with a target clock period and a specified tolerance to variations in the clock signal delays. Moreover, we

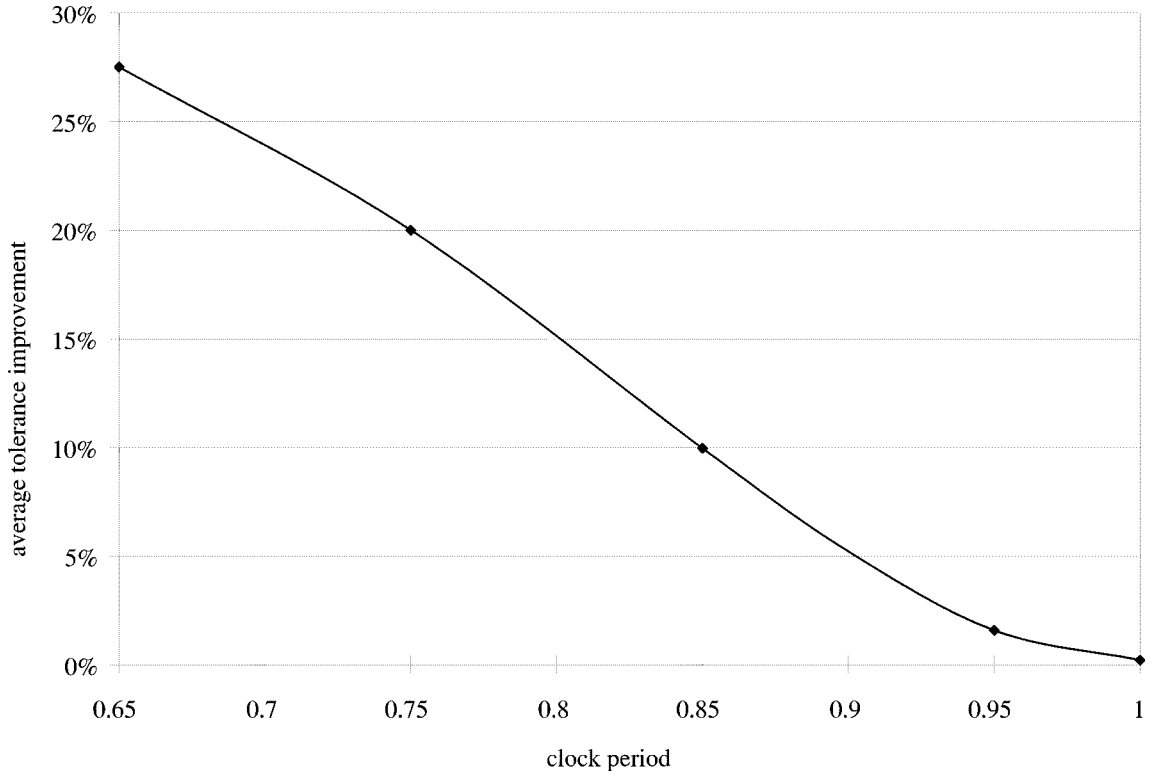

Fig. 20. Relative tolerance improvement by RSMAXT over the best achieved by retiming, clock scheduling, or sequential retiming and clock scheduling.

TABLE III
COMPARISON OF ALGORITHM RSMAXT AND AN EXACT MILP-BASED SOLVER FOR DELAY TOLERANCE MAXIMIZATION

Circuit	RSMAXT		MILP		$\Delta\tau_{rs}^m$ (%)
	τ_{rs}^m	CPU (sec)	τ_{rs}^m	CPU (sec)	
daio	0.30	0.1	0.30	7504.6	0
dk27	1.21	1.0	1.07	t/o	-12
dk15	0.53	1.9	0.53	9382.7	0
dk17	1.23	4.8	1.18	t/o	-4
lion	0.83	0.1	0.86	374.3	4
s208	1.15	36.8	1.11	t/o	-3
s420.1	1.15	119.2	1.04	t/o	-10
tav	0.70	0.3	0.70	19.1	0

present efficient and effective heuristics for the two optimization variants of the basic feasibility problem.

Experiments with a test suite obtained by modifying LGSynth93 and ISCAS89 benchmark circuits demonstrate the significant performance improvements achievable by simultaneous retiming and clock scheduling over the separate application of the two optimizations. For one-third of the circuits in our test suite, clock period improved by at least 10%. The sequential application of retiming and clock scheduling was found to be effective in improving clock frequency. Yet our heuristic can still improve the clock period by a significant 28% in best case. Tolerance to delay variations improved by 12% on the average over the best of retiming, clock scheduling, and sequential retiming and clock scheduling. In general, relative tolerance improvements were higher with more aggressive clock periods, thus suggesting the particular effectiveness of the simultaneous optimization for high-speed circuit design.

Our work can be generalized to encompass several variations of the retiming and clock scheduling problem. The basic MILP formulation can be extended in a straightforward manner to include a register minimization objective so that the target clock period and delay tolerance are achieved with the minimum number of registers. It can also be extended to handle more general delay models that account for changes in the register locations or impose restrictions on the mobility of certain registers.

An interesting problem in practice would be to investigate the effectiveness of combined retiming and clock scheduling when clock delay values are discrete or multiples of a basic delay. Another challenging and interesting research topic is the investigation of retiming and clock scheduling in the context of level-clocked circuitry with level-sensitive latches and multiphase clock schemes.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their helpful and insightful comments.

REFERENCES

[1] C. Albrecht, B. Korte, J. Schietke, and J. Vygen, "Cycle time and slack optimization for vlsi-chips," in *Dig. Tech. Papers 1999 IEEE Int. Conf. Computer-Aided Design*, Nov. 1999, pp. 232–237.

[2] S. Chakradhar and S. Dey, "Resynthesis and retiming for optimum partial scan," in *Proc. 31st ACM/IEEE Design Automation Conf.*, June 1994, pp. 87–93.

[3] L. -F. Chao and E. H. -M. Sha, "Retiming and clock skew for synchronous systems," in *Proc. Int. Symp. Circuits and Systems*, June 1994, pp. 283–286.
[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990.
[5] R. B. Deokar and S. S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," in *Proc. Int. Symp. Circuits and Systems*, May 1995, pp. 407–410.
[6] S. Dey and S. Chakradhar, "Retiming sequential circuits to enhance testability," in *Proc. 12th IEEE VLSI Test Symp.*, Apr. 1994, pp. 28–33.
[7] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Computers*, vol. 39, pp. 945–951, July 1990.
[8] E. G. Friedman, *Clock Distribution Networks in VLSI Circuits and Systems*. New York: IEEE Press, 1995.
[9] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou, "Optimizing two-phase, level-clocked circuitry," *J. ACM*, vol. 41, no. 1, pp. 148–199, January 1997.
[10] K. N. Lalgudi and M. C. Papaefthymiou, "DELAY: An efficient tool for retiming with realistic delay modeling," in *Proc. 32nd ACM/IEEE Design Automation Conf.*, June 1995.
[11] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1, 1991.
[12] X. Liu, M. C. Papaefthymiou, and E. G. Friedman, "Maximizing performance by retiming and clock skew scheduling," in *36th Design Automation Conf.*, June 1999, pp. 231–236.
[13] ——, "Optimal clock skew scheduling tolerant to process variations," in *Design, Automation, and Test in Europe*, Mar. 1999, pp. 643–649.
[14] B. Lockyear and C. Ebeling, "Optimal retiming of multi-phase, level-clocked circuits," in *Advanced Research in VLSI and Parallel Systems: Proc. 1992 Brown/MIT Conf.*, Mar. 1992.
[15] ——, "The practical application of retiming to the design of high-performance system," in *Dig. Tech. Papers 1993 IEEE Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 288–295.
[16] H.-G. Martin, "Retiming by combination of relocation and clock delay adjustment," in *Proc. Eur. Design Automation Conf.*, Sept. 1993, pp. 384–389.
[17] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," in *Dig.Tech. Papers 1993 IEEE Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 398–402.
[18] J. L. Neves and E. G. Friedman, "Optimal clock skew scheduling tolerant to process variations," in *Proc. 33rd ACM/IEEE Design Automation Conf.*, June 1996, pp. 623–628.
[19] M. C. Papaefthymiou and K. H. Randall, "TIM: A timing package for two-phase, level-clocked circuitry," *Proc. 30th ACM/IEEE Design Automation Conf*, June 1993.
[20] N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Retiming of circuits with single phase level-sensitive latches," in *Int. Conf. Computer Design*, Oct. 1991.
[21] T. Soyata, E. G. Friedman, and J. H. Mulligan Jr, "Incorporating interconnect, register, and clock distribution delays into the retiming process," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 105–212, Jan. 1997.
[22] M. C. Papaefthymiou, "Understanding retiming through maximum average-delay cycles," *Math. Syst. Theory*, vol. 27, pp. 65–84, 1994.
[23] T. Soyata and E. Friedman, "Retiming with non-zero clock skew, variable register and interconnect delay," in *Dig. Technical Papers 1994 IEEE Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 234–241.
[24] R. B. Deokar and S. S. Sapatnekar, "A fresh look at retiming via clock skew optimization," in *Proc. 32nd Design Automation Conf.*, June 1995, pp. 310–315.

Xun Liu (S'99) received the B.S. degree from the Department of electronic engineering, Tsinghua University, China, in 1996, and the M.S. degree in electrical engineering from University of Rochester, Rochester, NY, in 1998. Since 1998, he has been working toward the Ph.D. degree at the University of Michigan, Ann Arbor, MI.

His research interests include timing optimization, power estimation, and optimization for VLSI circuit and system designs.

Marios C. Papaefthymiou (M'93) received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, in 1988, and the S.M. and Ph.D. degrees in computer science from the Massachusetts Institute of Technology, Cambridge, in 1990 and 1993, respectively.

After a three-year term as Assistant Professor at Yale University, New Haven, CT, he joined the University of Michigan, Ann Arbor, where he currently is an Associate Professor of Electrical Engineering and Computer Science and Director of the Advanced Computer Architecture Laboratory. His research interests include several aspects of computer system design with an emphasis on efficient algorithms and novel architectures for high-performance, low-energy computing.

Dr. Papaefthymiou has received a Best Paper Award at the 1995 ACM/IEEE Design Automation Conference, a First Prize in the VLSI Design Contest of the 2001 ACM/IEEE Design Automation Conference, an ARO Young Investigator Award, CAREER and ITR Awards from NSF, and a number of IBM Partnership Awards. He is currently Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON VLSI SYSTEMS, and IEEE TRANSACTIONS ON THE COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He has served as the General Chair and Technical Program Chair for the 1997 and 1995 ACM/IEEE International Workshops on Timing Issues in the Specification and Synthesis of Digital Systems. He has also served on the Program Committees of numerous other international conferences and workshops, including the IEEE/ACM International Conference on Computer-Aided Design.

Eby G. Friedman (S'78–M'79–SM'92–F'00) received the B.S. degree from Lafayette College, Easton, PA, in 1979, and the M.S. and Ph.D. degrees from the University of California, Irvine, in 1981 and 1989, respectively, all in electrical engineering.

From 1979 to 1991, he was with Hughes Aircraft Company, rising to the position of Manager of the Signal Processing Design and Test Department, responsible for the design and test of high performance digital and analog ICs. He has been with the Department of Electrical and Computer Engineering at the University of Rochester, Rochester, NY, since 1991, where he is a Professor, the Director of the High Performance VLSI/1C Design and Analysis Laboratory, and the Director of the Center for Electronic Imaging Systems. His current research and teaching interests include high-performance synchronous digital and mixed-signal microelectronic design and analysis with application to high speed por processors and low-power wireless communications. He is the author of more than 150 papers and book chapters and the author or editor of six books in the fields of high-speed and low-power CMOS design techniques, high-speed interconnect, and the theory and application of synchronous clock distribution networks.

Dr. Friedman is the Editor-in-Chief of the IEEE TRANSACTIONS ON VLSI SYSTEMS, Regional Editor of the *Journal of Circuits, Systems, and Computers*, a Member of the editorial boards of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: ANALOG AND DIGITAL SIGNAL PROCESSING, *Analog Integrated Circuits and Signal Processing*, and *Journal of VLSI Signal Processing*. He is a Member of the Circuits and Systems (CAS) Society Board of Governors, and a Member of the technical program committee of a number of conferences. He previously was a Chair of the IEEE TRANSACTIONS ON VLSI SYSTEMS steering committee, CAS liaison to the Solid-State Circuits Society, Chair of the VLSI Systems and Applications CAS Technical Committee, Chair of the Electron Devices Chapter of the IEEE Rochester Section, Program or Technical Chair of several IEEE conferences, Editor of several special issues in a variety of journals, and a recipient of the Howard Hughes Masters and Doctoral Fellowships, an IBM University Research Award, an Outstanding IEEE Chapter Chairman Award, and a University of Rochester College of Engineering Teaching Excellence Award. He is also a Senior Fulbright Fellow.