Maximizing Performance by Retiming and Clock Skew Scheduling

Xun Liu Marios C. Papaefthymiou
Department of Electrical Engineering
and Computer Science
University of Michigan
Ann Arbor, Michigan 48109

Eby G. Friedman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, New York 14627

Abstract

The application of retiming and clock skew scheduling for improving the operating speed of synchronous circuits under setup and hold constraints is investigated in this paper. It is shown that when both long and short paths are considered, circuits optimized by the simultaneous application of retiming and clock scheduling can achieve shorter clock periods than optimized circuits generated by applying either of the two techniques separately. A mixed-integer linear programming formulation and an efficient heuristic are given for the problem of simultaneous retiming and clock skew scheduling under setup and hold constraints. Experiments with benchmark circuits demonstrate the efficiency of this heuristic and the effectiveness of the combined optimization. All of the test circuits show improvement. For more than half of them, the maximum operating speed increases by more than 21% over the optimized circuits obtained by applying retiming or clock skew scheduling separately.

1 Introduction

Retiming improves the speed of a digital circuit by relocating its storage elements while preserving the functionality of the original design. Clock scheduling achieves the same effect as retiming by introducing skew between the clock signals that control the timing of the storage elements within a circuit. This paper investigates the simultaneous application of retiming and clock scheduling for increasing the operating speed of a digital circuit. The combined application of the two optimization techniques adds flexibility during circuit synthesis. Moreover, it results in faster circuits than if either of the two optimizations is applied separately.

Two main contributions are presented in this paper. First, the problem of simultaneous retiming and clock scheduling under setup and hold constraints is formulated as a mixed-integer linear program (MILP) with $O(E^2)$ constraints, where E is the number of wires in the circuit. This program can be solved exactly using general MILP solvers that rely on branch-and-bound. The second contribution of this paper is an efficient and effective heuristic scheme for simultaneous retiming and clock scheduling.

Experiments with benchmark circuits from the LGSynth93 and ISCAS89 suites demonstrate that simultaneous retiming and clock scheduling can yield significantly faster circuits than the independent application of the two optimization techniques. They also indicate the kinds of circuit structures that are more amenable to speed-up by retiming and clock skew scheduling. For more than one half of the test circuits, the maximum operating speed improved by at least 21% over separate retiming or clock skew scheduling. Moreover, our heuristic scheme was orders of magnitude more efficient than the MILP solver with no sacrifice in accuracy.

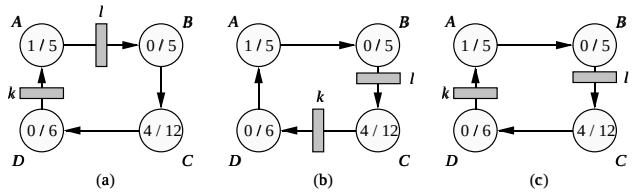

Figure 1: Simultaneous retiming and clock skew scheduling. (a) Original circuit. (b) Fastest retimed circuit with zero skew. (c) Fastest retimed circuit with nonzero skew.

Due to their complementary nature, retiming and clock scheduling have usually been investigated separately. Retiming has been investigated for a variety of clocking disciplines [7, 9, 11, 16], delay models [8, 17], and optimization objectives [1, 4, 13, 15]. A linear programming formulation of the clock scheduling problem was first described in [5]. A graph-theoretical approach to clock scheduling was presented in [3]. The combined application of retiming and clock scheduling was discussed in [12]. Retiming and clock scheduling for maximum tolerance to delay variations under setup and hold constraints was investigated in [10]. A two-step procedure for maximizing the operating frequency of a synchronous circuit by combining retiming and clock scheduling was proposed in [2]. That work considers only setup violations, however, and does not explore the expanded solution space that results when both setup and hold constraints are considered.

The remainder of this paper has seven sections. The superiority of combined retiming and clock scheduling over the separate application of these two optimizations is described in Section 2. Background material is given in Section 3. Section 4 presents necessary and sufficient conditions for correct timing when optimizing a circuit by retiming and clock skew scheduling. These constraints are restated as an equivalent mixed-integer linear program in Section 5. Our retiming and clock scheduling heuristic is presented in Section 6. In Section 7 the results obtained by the separate application of retiming and clock scheduling are compared with those obtained using the proposed integrated retiming and clock scheduling optimization technique. Our contributions are summarized in Section 8.

2 Motivation

An example that illustrates the effectiveness of simultaneous retiming and clock skew scheduling as compared to the separate application of the two optimization techniques is discussed in this section. The example also shows that the optimal circuit resulting from simultaneous retiming and clock skew scheduling cannot be obtained by simply computing an optimal retiming of the original circuit with zero clock skews and then sequentially performing clock skew scheduling.

In the circuit graph shown in Figure 1, each vertex represents a block of combinational logic, and each edge represents a wire. The rectangles denote edge-triggered registers. The pair x/y associated with each vertex denotes the minimum and maximum propagation

delay of the signals through the corresponding logic block.

When the clock skew is zero, the minimum clock period is the longest delay of all the combinational paths in the circuit. In this case, the goal of retiming is to balance the longest delay of all the data paths by relocating the registers. If nonzero clock skew is introduced, however, the circuit can successfully operate at a clock period which equals the largest difference in the delays of the slowest path and the fastest path between any pair of registers.

The original circuit depicted in Figure 1(a) achieves a clock period of $\Phi = 23$ tu (time units) when the clock skew is zero. If register l "sees" the clock edge by 4 tu earlier than register k, the circuit can function correctly with a clock period of $\Phi = 19$ tu. This time is the shortest clock period that can be achieved by scheduling the clock skews without introducing any signal races, since the propagation delay difference along the critical path BCD is 19 tu.

The retimed circuit in Figure 1(b) is obtained from the original circuit by shifting register k backward across block D and register l forward across block B. With zero clock skew, this circuit is optimally retimed and achieves a clock period of $\Phi = 16$ tu. If the clock edge arrives at register k earlier than at register l by 1 tu, this retimed circuit can achieve a shorter clock period of $\Phi = 15$ tu. No further clock period improvements are possible because the propagation delay difference along path DAB is 15 tu.

In Figure 1(c), another retimed version of the original circuit that is obtained by shifting register l across block B is shown. For this circuit, the shortest clock period that will not result in any timing violations when the clock skew is zero is $\Phi = 18$ tu. If the clock edge arrives at register k later than at register l by 4 tu, however, this circuit can function correctly with a clock period of $\Phi = 14$ tu. This time is the shortest clock period that can be achieved by applying both retiming and clock scheduling, since the total propagation delay around the cycle is 28 tu which must be distributed between two combinational paths.

3 Background

In Subsection 3.1, the circuit and delay model is presented. Background material for retiming and clock skew scheduling is given in Subsections 3.2 and 3.3, respectively.

3.1 Circuit and Delay Model

We model an edge-triggered circuit as a directed multigraph $G = \langle V, E, d, w\rangle$. The vertices V correspond to the combinational logic elements in the circuit. For each vertex $v \in V$, the propagation delay of the corresponding combinational logic block is $d(v)$. Our results can be extended to include the case where each logic block has a maximum propagation delay $d_{\max}(v)$ and a minimum propagation delay $d_{\min}(v)$.

The directed edges E in G model the interconnections among the combinational blocks. Each edge $e \in E$ corresponds to a wire connecting a combinational block output 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 an integer, nonnegative edge-weight $w(e)$. For every directed cycle of G, there is an edge with a strictly positive register count.

3.2 Retiming

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

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

Under this definition, the output of v's computation in G_r is generated $r(v)$ clock cycles later than in G. For the retimed circuit G_r to be *well-formed*, all edges $e \in E$ must satisfy the inequality

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

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

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

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

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

The only paths $u \overset{p}{\rightsquigarrow} v$ that can become combinational in G_r, and thus could possibly lead to a timing violation, 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

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

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

where $d(p) = \sum_{x \in p} d(x)$ represent the longest and shortest propagation delays from u to v, respectively, whenever the retimed circuit includes a combinational path between the two vertices. Therefore, the clock period of any circuit that is obtained by retiming a circuit G is always some element in the $O(V^2)$-size set of $D(u, v)$.

When only long paths are considered, a retimed circuit that achieves a given clock period c can be computed in $O(VE)$ steps. A retimed circuit that achieves the minimum possible clock period can be computed in $O(VE + V^2 \lg V)$ steps [9].

3.3 Clock Skew Scheduling

In synchronous digital circuits, a clock signal provides a global time reference that synchronizes the flow of the data between the storage elements. The clock signals are delivered throughout the circuit by a clock distribution network [6]. The difference between the arrival times of a clock signal at two sequentially-adjacent registers in a circuit is known as the clock skew between these two registers. A *clock schedule* of an edge-triggered circuit $G = \langle V, E, d, w\rangle$ is a real-valued edge-labeling $s : E \to \mathbf{R}$. This labeling describes the propagation delay from the global clock source to each wire e in a circuit. By adjusting these local clock skews, timing violations can be fixed (or created). Consider, for example, a combinational path $u \overset{p}{\rightsquigarrow} v$ which is bounded by registers on the edges $? \xrightarrow{e} u$ and $v \xrightarrow{e'} ?$. If $s(e) \geq s(e')$, then the time available for the propagation of signals from e to e' decreases by the difference $s(e) - s(e')$. This decrease may cause p to become a critical path or eliminate a race condition along p. Conversely, if $s(e) \leq s(e')$, then the time available for the signals to propagate from e to e' increases by the difference $s(e') - s(e)$. This increase may remove a long path violation or introduce a short path violation.

In the clock scheduling problem, the clock skews are adjusted so that no setup or hold violations exist in the circuit. A linear programming framework for clock scheduling was first presented

in [5]. A graph-theoretic approach to clock scheduling was subsequently described in [3]. In both papers, the relative placement of the storage elements is assumed to be fixed. Algorithms for scheduling local clocks to improve the tolerance of a circuit to process parameter variations are described in [14].

4 Clock Scheduling and Retiming Constraints

This section describes a set of $O(E^2)$ necessary and sufficient conditions that must be satisfied by a retiming function and a clock skew scheduling function to obtain a circuit that achieves a target clock period with no setup or hold violations. The following theorem presents a shortest-paths formulation of the clock scheduling problem under setup and hold constraints and fixed register locations [5].

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

$$\Delta(u, v) + s(e) - s(e') \geq T_{hold} \ , \tag{7}$$

$$D(u, v) + s(e) - s(e') \leq c - T_{setup} \ . \tag{8}$$

□

The following theorem provides a set of $O(E^2)$ constraints for correct timing when clock skew scheduling and retiming are applied simultaneously. These constraints follow directly from the clock scheduling conditions gicen in Theorem 1. The proof is straightforward and is omitted due to lack of space.

Theorem 2 *Let $G = \langle V, E, d, w\rangle$ be an edge-triggered circuit and c be a constant. Moreover, let $r : V \rightarrow \boldsymbol{Z}$ be a retiming function and let $s : E \rightarrow \boldsymbol{R}$ be a clock scheduling function. The retimed circuit G_r is well formed and achieves a clock period c if and only if for every edge $u \xrightarrow{e} v \in E$, we have*

$$w(e) + r(v) - r(u) \geq 0 \ , \tag{9}$$

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

$$E(e, e') > 0 \ \ \Rightarrow \ \ W_r(u, v) \geq 1 \text{ or } w_r(e) = 0 \text{ or } w_r(e') = 0 \ , \tag{10}$$

where $E(e, e') = D(u, v) + s(e) - s(e') - c + T_{setup}$ for the setup constraints, and $E(e, e') = -(\Delta(u, v) + s(e) - s(e') - T_{hold})$ for the hold constraints. □

5 Mixed-Integer Linear Program

The necessary and sufficient conditions in Theorem 2 do not appear to be amenable to efficient algorithmic solutions. In this section we present an equivalent, mixed-integer linear programming formulation of these constraints that can be solved using general MILP solvers. These $O(E^2)$ constraints are obtained by restricting the solution space of the constraints in Lemma 3 while maintaining their feasibility. The final set of constraints comprises only linear inequalities with integer and real unknowns.

We first define a *companion graph* $G' = \langle V', E', w'\rangle$ that serves as a useful tool for transforming the timing constraints from Theorem 2 into a mixed-integer linear program. The construction of the graph G' from the circuit graph G is identical to the graph presented in [8]. Each edge $u \xrightarrow{e} v \in E$ is segmented into two edges, $u \xrightarrow{e_1} x_{uv}$ and $x_{uv} \xrightarrow{e_2} v$, where x_{uv} is a dummy vertex. The edge e_1 has exactly one register where the corresponding edge $e \in E$ has a positive register count and zero registers otherwise. Thus, the register count of e_1 serves as an *index function* for the register count of the corresponding generating edge $e \in E$. The edge e_2 carries the balance of the registers up to $w(e)$.

In mathematical terms, the companion graph $G' = \langle V', E', w'\rangle$ is defined as

$$\begin{aligned} V' &= V \cup \left\{ x_{uv} : u \xrightarrow{e} v \in E \right\} \ , \\ E' &= \left\{ u \xrightarrow{e_1} x_{uv}, x_{uv} \xrightarrow{e_2} v : u \xrightarrow{e} v \in E \right\} \ , \end{aligned}$$

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

$$\begin{aligned} w'(e_1) &= \min\{1, w(e)\} \ , \text{ and} \\ w'(e_2) &= w(e) - \min\{1, w(e)\} \ . \end{aligned}$$

The following lemma recasts Theorem 2 in terms of the companion graph G' and a corresponding retiming function r'. If r' is known, $r(u)$ can be obtained for every $u \in V$ by setting $r(u) = r'(u)$.

Lemma 3 *Let $G = \langle V, E, d, w\rangle$ be an edge-triggered circuit, let $G' = \langle V', E', w'\rangle$ be its corresponding companion graph, and let c be a constant. Moreover, let $r' : V' \rightarrow \boldsymbol{Z}$ be a retiming function, and let $s_M : E \rightarrow \boldsymbol{R}$ be a clock scheduling function. Then the retimed circuit G_r is well formed and achieves a clock period c if and only if for every edge $u \xrightarrow{e} v \in E'$, we have*

$$w'(e) + r'(v) - r'(u) \geq 0 \ , \tag{11}$$

for every edge $u \xrightarrow{e_1} x_{uv} \in E'$,

$$w'(e_1) + r'(x_{uv}) - r'(u) \leq 1 \ , \tag{12}$$

for every pair of edges $u \xrightarrow{e_1} x_{uv}$, $x_{uv} \xrightarrow{e_2} v \in E$,

$$w'(e_2) + r'(v) - r'(x_{uv}) \leq F \cdot \left(w'(e_1) + r'(x_{uv}) - r'(u) \right) \ , \tag{13}$$

where $F = \max\{W(u, v) + W(v, u) : u, v \in V\}$, and for every pair of edges $? \xrightarrow{e} u$, $v \xrightarrow{e'} ? \in E$,

$$E(e, e') > 0 \ \Rightarrow \ W_{r'}(u, v) \geq 1 \text{ or } w_{r'}(e_1) = 0 \text{ or } w_{r'}(e_1') = 0 \ , \tag{14}$$

where $E(e, e') = D(u, v) + s(e) - s(e') - c + T_{setup}$ for the setup constraints, and $E(e, e') = -(\Delta(u, v) + s(e) - s(e') - T_{hold})$ for the hold constraints. □

The constraints in Lemma 3 can be simplified by reducing the number of disjunctions in Relation (14). In the following lemma, an equivalent set of constraints is given with only one disjunction.

Lemma 4 *For every pair of edges $? \xrightarrow{e} u$, $v \xrightarrow{e'} ? \in E$, Relation (14) is equivalent to the disjunction,*

$$E(e, e') \leq 0 \ \text{ or } \ w'_{r'}(e_1) + w'_{r'}(e_1') - W_{r'}(u, v) \leq 1 \ , \tag{15}$$

where $E(e, e') = D(u, v) + s(e) - s(e') - c + T_{setup}$ and $E(e, e') = -(\Delta(u, v) + s(e) - s(e') - T_{hold})$ for the setup and hold constraints, respectively. □

The following lemma gives two upper bounds on the quantity $w'_{r'}(e_1) + w'_{r'}(e_1') - W_{r'}(u, v)$ based on Relation (15).

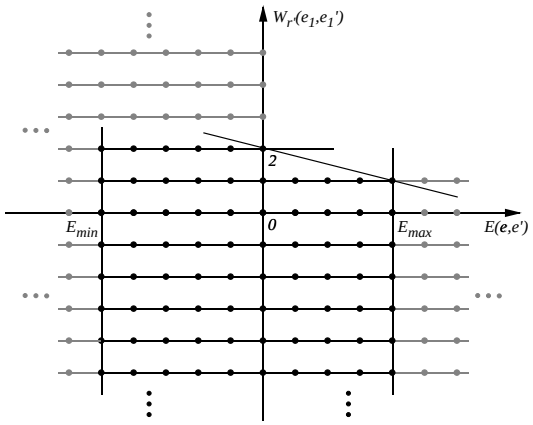

Figure 2: Equivalent convex solution space.

Lemma 5 *Let $r' : V' \rightarrow \boldsymbol{Z}$ be a retiming function that satisfies the conditions in Lemma 3. Then, for every pair of edges $? \xrightarrow{e} u, v \xrightarrow{e'} ? \in E$,*

$$w'_{r'}(e_1) + w'_{r'}(e'_1) - W_{r'}(u,v) \leq 2 \,, \tag{16}$$

$$w'_{r'}(e_1) + w'_{r'}(e'_1) - W_{r'}(u,v) \leq 2 - \frac{E(e,e')}{E_{\max}(e,e')} \,, \tag{17}$$

where $E_{\max}(e,e')$ is an upper bound of $E(e,e')$ that depends on the maximum possible clock skew values, which are determined by the largest realizable chip die size. □

The solution space derived from the bounds in Lemma 5 is illustrated in Figure 2. The bold line segments represent possible solutions. The shaded lines and points denote the points of the original solution space that have been excluded using the upper bounds. The horizontal line in the second quadrant arises from Inequality (16), and the sloped upper bound in the first quadrant arises from Inequality (17). The two vertical lines correspond to the bounds on $E(e,e')$.

Based on Lemmas 4 and 5, the simultaneous retiming and clock scheduling problem can now be recast as a mixed-integer linear program with $O(E^2)$ constraints.

Theorem 6 *Let $G = \langle V, E, d, w \rangle$ be an edge-triggered circuit, and let c be a constant. Moreover, let $r : V \rightarrow \boldsymbol{Z}$ be a retiming function, and let $s : E \rightarrow \boldsymbol{R}$ be a clock scheduling function. Then the retimed circuit G_r is well formed and achieves a clock period $\Phi(G_r) \leq c$ if and only if for every edge $u \xrightarrow{e} v \in E'$, we have*

$$w'(e) + r'(v) - r'(u) \geq 0 \,, \tag{18}$$

for every edge $u \xrightarrow{e_1} x_{uv} \in E'$,

$$w'(e_1) + r'(x_{uv}) - r'(u) \leq 1 \,, \tag{19}$$

for every pair of edges $u \xrightarrow{e_1} x_{uv}, x_{uv} \xrightarrow{e_2} v \in E$,

$$w'(e_2) + r'(v) - r'(x_{uv}) \leq F \cdot \left(w'(e_1) + r'(x_{uv}) - r'(u) \right) \,, \tag{20}$$

where $F = \max\{W(u,v) + W(v,u) : u, v \in V\}$, and for every pair of edges $? \xrightarrow{e} u, v \xrightarrow{e'} ? \in E$,

$$E(e,e') \leq E_{\max}(e,e') \,, \tag{21}$$

$$E(e,e') \geq E_{\min}(e,e') \,, \tag{22}$$

```
RS(G)
 1 set G_r = G
 2 (c, s) = schedule(G_r)
 3 while HR(G_r, s, c) = success
 4       (c', s) = schedule(G_r)
 5       if c' < c
 6         then c = c'
 7         else if no improvement for n consecutive iterations
 8                 then break
 9 return (G_r, s, c)
```

Figure 3: Procedure RS for simultaneous retiming and clock skew scheduling.

```
HR(G_r, s, c)
 1 compute S_tight, H_tight, and G_tight
 2 if S_tight contains a directed cycle
 3   then return fail
 4   else for every vertex u in a directed L ∈ G_tight
 5           if ∃x, v ∈ L such that (u, x) ∈ H_tight, (u, v) ∈ S_tight
 6             then shift register u forward
 7                  return success
 8           if ∃x, v ∈ L such that (x, u) ∈ H_tight, (v, u) ∈ S_tight
 9             then shift register u backward
10                  return success
11 return fail
```

Figure 4: Procedure HR for heuristic retiming.

$$w'_{r'}(e_1) + w'_{r'}(e'_1) - W_{r'}(u,v) \leq 2 \,, \tag{23}$$

$$w'_{r'}(e_1) + w'_{r'}(e'_1) - W_{r'}(u,v) \leq 2 - \frac{E(e,e')}{E_{\max}(e,e')} \,, \tag{24}$$

where $E(e,e') = D(u,v) + s(e) - s(e') - c + T_{setup}$ for the setup constraints, and $E(e,e') = -(\Delta(u,v) + s(e) - s(e') - T_{hold})$ for the hold constraints. □

6 Heuristic Scheme for Retiming and Clock Scheduling

In this section we describe a heuristic procedure for simultaneous retiming and clock skew scheduling. This heuristic executes faster than general MILP solvers by examining a subset of the solution space and creating a set of results that converge towards the optimal solution. Consequently, sometimes only a suboptimal solution is reached.

A pseudocode description of our heuristic procedure RS for simultaneous retiming and clock skew scheduling is shown in Figure 3. Given an edge-triggered circuit G, this heuristic returns a retimed circuit G_r, a minimum clock period c, and a clock schedule s such that G_r achieves c. The main idea in this heuristic is to iteratively perform clock skew scheduling followed by retiming. In line 1, G_r is initialized to G. The function $schedule$ in line 2 computes a clock schedule s that results in the shortest possible clock period c for the original G. Subsequently, the **while** loop in lines 3–8 iterates until a retimed G_r with minimum clock period is computed. Each time line 3 is executed, a heuristic retiming procedure HR is performed with the objective to relax the "tight" setup and hold constraints, that is, the constraints that are satisfied with equality under r, s, and c. In line 4, clock scheduling is performed

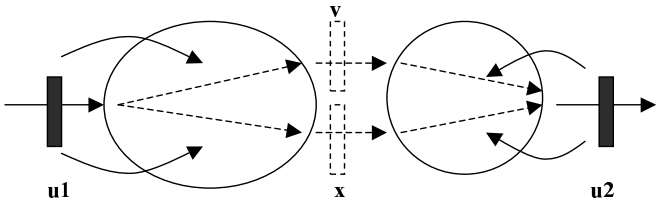

Figure 5: Relocation of registers in procedure HR.

on the retimed circuit G_r generated by procedure HR. The loop exits when no new G_r is found or the clock period cannot be improved any further. A predefined number n is introduced so that the algorithm can escape a local minimal.

The pseudocode of the heuristic retiming procedure HR is shown in Figure 4. Three *slack graphs*, called S_{tight}, H_{tight}, and G_{tight}, are used in this procedure to encode the timing constraints corresponding to each clock skew schedule. For each edge $e \in G_r$ with nonzero register count, a vertex is introduced in all three graphs. For each pair of registers that are connected by a combinational path in G_r, if $D(u,v) + s(u) - s(v) = c - T_{setup}$, an edge (u,v) is added in S_{tight}. Similarly, if $\Delta(u,v) + s(u) - s(v) = T_{hold}$, an edge (u,v) is added in H_{tight}. G_{tight} is the union of S_{tight} and H_{tight} with all the edges in H_{tight} inverted.

Procedure HR operates as follows. In line 1, the three slack graphs are computed. In lines 2–3, the heuristic determines if G_r has a set of setup constraints that precludes further improvement of the clock period by retiming or clock scheduling. The circuit is retimed in lines 4–10 so that the register boundaries of a "critically imbalanced" circuit segment approach each other. Such a segment exists whenever two registers are connected by two possibly sequential paths in which the setup and hold constraints are tight and the difference of maximum and minimum delays is maximum. This operation is illustrated in Figure 5. If no register can be moved, line 11 reports the failure to build a new G_r.

The following lemma shows that lines 2–3 of the heuristic correctly terminate the search procedure. The proof is omitted due to lack of space.

Lemma 7 *If there exists a directed cycle in S_{tight}, the clock period of the circuit cannot be reduced any further by retiming and/or clock scheduling.*

In the following two theorems we demonstrate that procedure HR always makes progress towards a shorter clock period. The first theorem ensures that when the slack graph is acyclic, it is always possible to find a clock schedule for which the circuit G_r can achieve a shorter clock period. The second theorem shows that the register relocation in lines 4–10 breaks any directed cycles within the slack graph. Proofs are omitted due to lack of space.

Theorem 8 *If no directed cycle exists in G_{tight}, the clock period can always be further reduced by retiming and clock skew scheduling.*

Theorem 9 *Each execution of lines 4–10 in procedure* HR *breaks a directed cycle in the slack graph G_{tight}.*

7 Experimental Results

This section presents results from the application of simultaneous retiming and clock scheduling on LGSynth93 and ISCAS89 benchmark circuits. The following experimental methodology was applied. The shortest clock period of each original circuit under zero skew was first computed. Subsequently, each test circuit was optimized by retiming, clock scheduling, and simultaneous retiming and clock scheduling to achieve the minimum possible clock period. Simultaneous retiming and clock scheduling was performed using the heuristic procedure RS. It was also performed using a MILP-based branch-and-bound solver that we implemented independently. Our results show that simultaneous retiming and clock scheduling can result in significant performance improvements over each of the two optimizations applied separately. Moreover, they demonstrate the efficiency and effectiveness of our heuristic.

The results of our experiments are shown in Table 1. The first three columns give the name and size of each circuit in our test suite. These test circuits were slightly modified versions of the LGSynth93 and ISCAS89 benchmark circuits that were obtained by introducing an additional register around each loop. Since most of the original benchmark circuits are finite state machines with single-register loops, they could not be improved by either retiming or clock skew scheduling. The fourth column of the table gives the shortest clock period that is achievable by the unoptimized circuits. The shortest clock period that is achievable by optimal retiming or optimal clock scheduling (whichever is better) is listed in the fifth column.

The results from the application of the heuristic procedure RS are listed in columns six through eight in Table 1. The shortest clock period that was achieved by simultaneous retiming and clock scheduling is given in the sixth column. The relative improvement in the minimum clock period of the circuit, which is derived by dividing the difference of columns six and five by the data in column six, is listed in the seventh column. The eighth column gives the runtime of the heuristic procedure.

The performance of our MILP-based branch-and-bound solver is reported in columns nine through eleven of Table 1. This solver uses a topological sort order to arrange all integer variables. Once a retiming function is computed, the clock delays are computed using a Bellman-Ford single-source shortest-paths algorithm. For all circuits but `s1423`, a timeout equal to 10,000 seconds was specified. For `s1423`, the timeout was set to 20,000 seconds. The shortest clock period that is computed by the MILP-based algorithm is given in the ninth column. The relative performance improvements achieved over the best clock period achievable by either retiming or clock scheduling is given in the tenth column. The runtime of the solver is listed in the last column. The letters t/o indicate that the algorithm reached its timeout limit and was interrupted.

In our experiments, simultaneous retiming and clock scheduling improved the performance of all test circuits and resulted in significant improvements for most of them. For more than one half of the test circuits, relative improvements exceeded 21%. For most circuits, the MILP-based scheme ran out of time. Whenever the MILP solver terminated with the optimum answer, it was two to three orders of magnitude slower than the heuristic approach. The heuristic procedure RS always computed a solution that was at least as good as the solution obtained with the MILP solver. In fact, in several cases it computed a better solution, because the exact MILP solver reached the timeout limit. Our experiments were performed on a Pentium Pro II with 128MB of main memory.

The capacitance load model used in the gate delay calculations was based on the formula $a + b \cdot (fanout + rand)$, which is derived from the widely used linear delay formula $a + b \cdot fanout$. The parameters a and b denote the intrinsic gate delay and the delay increment of a single gate load, respectively. These values were obtained from the library `iwls93.mis2lib` in the LGSynth93 benchmark. The parameter $rand$ was uniformly distributed in the interval [-1,1] to introduce a zero-mean variation to gate delays.

Table 1: Clock period improvements achieved by simultaneous retiming and clock scheduling.

circuit	vertex count	edge count	Φ (tu)	$\min\{\Phi_r, \Phi_s\}$ (tu)	Φ_{RS} (tu)	speedup (%)	CPU time (sec)	Φ_{MILP} (tu)	speedup (%)	CPU time (sec)
daio	24	37	3.24	1.26	0.86	31	0.1	0.86	31	1.0
dk17	31	61	5.21	5.16	4.67	4	0.2	4.67	4	45.8
tav	38	71	3.73	2.22	1.75	21	0.3	1.75	21	1.4
bbtas	41	97	4.10	4.10	3.71	9	0.6	3.71	9	23.4
s208	49	124	5.19	3.77	2.97	21	2.6	3.09	18	t/o
s420	60	191	7.02	5.31	5.06	5	4.6	5.21	2	t/o
bbara	68	195	5.45	4.84	4.50	7	4.2	4.50	7	22.1
dk14	81	250	6.26	5.75	5.58	3	7.0	5.58	3	387.1
ex4	84	221	6.47	6.38	5.80	9	7.1	5.80	9	118.3
s208.1	131	209	8.68	6.00	5.35	11	21.5	6.00	0	t/o
s298	152	283	8.94	7.91	4.41	44	302.4	7.91	0	t/o
s444	245	422	9.92	5.98	4.67	23	2,757.5	5.98	0	t/o
s344	257	377	24.25	19.13	17.89	6	66.7	19.00	0	t/o
s349	263	386	19.03	14.20	13.60	5	68.5	14.20	0	t/o
s382	287	441	15.10	9.26	7.31	21	1,266.7	9.16	0	t/o
s526n	304	561	8.81	6.97	5.23	25	1,426.7	6.97	0	t/o
s400	308	472	16.84	10.59	8.30	23	1,312.0	10.59	0	t/o
s526	342	589	10.36	9.06	5.64	38	2,348.3	9.06	0	t/o
s1423	1034	1515	63.88	57.52	43.08	23	16,541.0	57.52	0	t/o

8 Conclusion

This paper has presented an investigation of optimizing synchronous circuits to maximize their operating speeds by simultaneous retiming and clock scheduling. Our work considers both setup and hold constraints. It is shown that in this context, the combined optimization can result in faster circuits than if either of the two optimizations is applied separately. A precise mathematical characterization of the integrated retiming and clock scheduling problem is presented as a mixed-integer linear program. Moreover, an efficient and effective heuristic is described. Our experimental results on benchmark circuits demonstrate that the combined application of retiming and clock scheduling can significantly increase the performance of the original circuits.

Acknowledgments

This research was supported in part by the National Science Foundation under Grant No. MIP-9610108, a grant from the New York State Science and Technology Foundation, and by grants from the Xerox, IBM, and Intel Corporations.

References

[1] S. Chakradhar and S. Dey. Resynthesis and retiming for optimum partial scan. In *Proceedings of the 31st ACM/IEEE Design Automation Conf.*, pages 87–93, June 1994.

[2] L.-F. Chao and E. H.-M. Sha. Retiming and clock skew for synchronous systems. In *Proc. International Symp. on Circuits and Systems*, pages 283–286, June 1994.

[3] R. B. Deokar and S. S. Sapatnekar. A graph-theoretic approach to clock skew optimization. In *Proc. International Symp. on Circuits and Systems*, pages 407–410, May 1995.

[4] S. Dey and S. Chakradhar. Retiming sequential circuits to enhance testability. In *Proc. 12th IEEE VLSI Test Symp.*, pages 28–33, April 1994.

[5] J. P. Fishburn. Clock skew optimization. *IEEE Trans. on Computers*, 39(7):945–951, July 1990.

[6] E. G. Friedman. *Clock Distribution Networks in VLSI Circuits and Systems*. IEEE Press, 1995.

[7] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. Optimizing two-phase, level-clocked circuitry. *Journal of the ACM*, 41(1):148–199, January 1997.

[8] 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.

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

[10] X. Liu, M. C. Papaefthymiou, and E. G. Friedman. Optimal clock skew scheduling tolerant to process variations. In *Design, Automation, and Test in Europe*, pages 643–649, March 1999.

[11] 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.* MIT Press, March 1992.

[12] H.-G. Martin. Retiming by combination of relocation and clock delay adjustment. In *Proc. European Design Automation Conf.*, pages 384–389, September 1993.

[13] J. Monteiro, S. Devadas, and A. Ghosh. Retiming sequential circuits for low power. In *Digest of Technical Papers of the 1993 IEEE International Conf. on CAD*, pages 398–402, November 1993.

[14] J. L. Neves and E. G. Friedman. Optimal clock skew scheduling tolerant to process variations. In *Proc. 33rd ACM/IEEE Design Automation Conf.*, pages 623–628, June 1996.

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

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

[17] T. Soyata, E. G. Friedman, and J. H. Mulligan, Jr. Incorporating interconnect, register, and clock distribution delays into the retiming process. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16(1):105–120, January 1997.