An Efficient Retiming Algorithm Under Setup and Hold Constraints*

Chuan Lin and Hai Zhou
Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208
{clin, haizhou}@ece.northwestern.edu

ABSTRACT

In this paper we present a new efficient algorithm for retiming sequential circuits with edge-triggered registers under both setup and hold constraints. Compared with the previous work [17], which computed the minimum clock period in $O(|V|^3|E|\lg|V|)$ time, our algorithm solves the same problem in $O(|V|^2|E|)$ time. Experimental results validate the efficiency of our algorithm.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*;
J.6 [**Computer-Aided Engineering**]: Computer-Aided Design

General Terms

Algorithms, Performance, Design

Keywords

Retiming

1. INTRODUCTION

Retiming is a traditional sequential optimization technique that moves registers within a circuit without destroying its functionality. It was originally proposed by Leiserson and Saxe in [10] to reduce the clock period and area of edge-triggered circuits. Since then, it has firmly established its reputation as one of the most effective techniques for sequential circuit optimization.

Besides its steady improvements on performance [6, 9, 21, 24, 27], retiming has been extended to timing and area optimization of level-clocked circuits [8, 13, 14], skew scheduling [18], floorplan and placement [4, 25], circuit partitioning and clustering [3, 16], logic synthesis [19], power optimization [20], and testability [5,15]. Recent progresses on semiconductor technology saw an increase in the number of global wires whose delays are longer than one clock period [2,7], and retiming is again a promising technique that could be leveraged [1, 11, 12, 26, 28].

*This work is supported by the NSF under CCR-0238484.

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

Among these researches, many were focus on minimizing clock period [1,8,11–13,24,27,28]. However, the polynomial-time retiming algorithms proposed in these papers can only be used to satisfy setup constraint.

An algorithm was presented in [22] for retiming single-phase level-clocked circuits under both setup and hold constraints, but its worst-case running time was exponential. On the other hand, [23] proposed to handle hold constraint by inserting extra delays on short paths, but it may result in significant area penalty and may fail when the delay insertion is required to be discrete. The existence of a polynomial-time algorithm for minimum period retiming of edge-triggered circuits under both setup and hold constraints has remained as an open problem until [17].

Given an edge-triggered sequential circuit $G = (V,E)$, a target clock period, a setup time S, and a hold time H, the algorithm in [17] computed a valid retiming satisfying both setup and hold constraints, or reported that there is no such a retiming. However, its worst-case running time was high. In conjunction with binary search, it took $O(|V|^3|E|\lg|V|)$ time to determine the minimum clock period.

In this paper we present an algorithm that finds a minimum period retiming satisfying both setup and hold constraints in $O(|V|^2|E|)$ time. Interestingly, it was conjectured in [17] that an $O(|V|^2|E|)$-time algorithm may be designed for finding a valid retiming under a target clock period. Our algorithm not only confirms the conjecture, but also shows that finding a minimum period retiming can also be solved in $O(|V|^2|E|)$ time—a much better result.

The rest of this paper is organized as follows. Section 2 presents the problem formulation. Our algorithm is presented in Section 3, followed by an illustration example in Section 4. Experimental results are given in Section 5. Conclusions are drawn at the end.

2. PROBLEM FORMULATION

We model an edge-triggered circuit as a directed graph $G = (V,E,d,D,w)$. Each vertex $v \in V$ represents a gate and each edge $e \in E$ represents a signal passing from one gate to another–with gate minimum delays $d : V \to \mathcal{R}^+$, gate maximum delays $D : V \to \mathcal{R}^+$ and register numbers $w : E \to \mathcal{N}$. Except for the minimum delays d, this model is identical to the one used by Leiserson and Saxe [10]. Without loss of generality, we assume that G is strongly connected, which is consistent with [17]. All registers in the circuit are assumed to have equal setup times S and equal hold times H. To ease the presentation, register delays and clock skews are assumed to be zero. Non-zero values can be easily incorporated in the same way as in [17].

The problem we want to solve can be formally described as follows.

PROBLEM 1. *Given a circuit $G = (V,E,d,D,w)$, find a relocation of registers such that both setup and hold constraints are satisfied under the minimum clock period.*

To guarantee that the new registers are actually a relocation of the old ones, a label $r : V \to \mathcal{Z}$ is used to represent how many registers are moved from the outgoing edges to the incoming edges of each vertex. Based on this, the number of registers on an edge (u,v) after retiming can be computed as

$$w_r(u,v) \stackrel{\triangle}{=} w(u,v) + r(v) - r(u).$$

We use $w(p)$ to denote the number of registers on a simple path $p = u \rightsquigarrow v$ before retiming. Then the number of registers on p after retiming can be represented as

$$w_r(p) \stackrel{\triangle}{=} w(p) + r(v) - r(u).$$

A partial order $(\leq)$ can be defined between two labels r and r' as follows.

$$r \leq r' \stackrel{\triangle}{=} \big(\forall v \in V : r(v) \leq r'(v)\big).$$

Furthermore, to avoid explicitly enumerating paths, we introduce two labels $t : V \to \mathcal{R}^+$ and $T : V \to \mathcal{R}^+$ to represent the earliest and the latest output arrival times of a gate respectively. In other words, $t(v)$ and $T(v)$ are the minimum and the maximum delays to the output of gate v from the nearest preceding registers. Using these notations, the validity of a retiming (r,t,T) is defined by the following conditions.

$$P0(r) \stackrel{\triangle}{=} \big(\forall (u,v) \in E : w_r(u,v) \geq 0\big)$$

$$P1(r) \stackrel{\triangle}{=} \big(\forall (u,v) \in E : w_r(u,v) \leq 1\big)$$

$$P2(r,t,T) \stackrel{\triangle}{=} \big(\forall (u,v) \in E : w_r(u,v) = 0 \;\Rightarrow\; T(v) - T(u) \geq D(v) \;\wedge\; t(v) - t(u) \leq d(v)\big)$$

$$P3(r,t,T) \stackrel{\triangle}{=} \big(\forall (u,v) \in E : w_r(u,v) = 1 \Rightarrow\; T(v) \geq D(v) \;\wedge\; t(v) \leq d(v)\big)$$

$$P4(r,t) \stackrel{\triangle}{=} \big(\forall (u,v) \in E : w_r(u,v) = 1 \;\Rightarrow\; t(u) \geq H\big)$$

The condition $P0$ states that a valid retiming should have non-negative number of registers on any edge. The condition $P1$ comes from the fact that an edge cannot accommodate more than one register without causing hold violations among the registers. The conditions $P2$ and $P3$ define T and t as the latest and the earliest arrival times. More specifically, the latest (earliest) arrival time of a gate is at least (at most) the summation of the gate delay and the latest (earliest) arrival time of its fanins. The condition $P4$ is the hold constraint. These conditions altogether characterize a valid retiming that permits a feasible clock period of $S + T_{max}$, where S is the setup time and

$$T_{max} \stackrel{\triangle}{=} \max_{v \in V} T(v).$$

We use P to denote the conjunction of $P0$-$P4$, i.e.,

$$P(r,t,T) \stackrel{\triangle}{=} P0(r) \wedge P1(r) \wedge P2(r,t,T) \wedge P3(r,t,T) \wedge P4(r,t)$$

The optimality of a min-period retiming (r,t,T) is given by the following condition.

$$P5(r,t,T) \stackrel{\triangle}{=} \big(\forall r',t',T' : P(r',t',T') \;\Rightarrow\; T_{max} \leq T'_{max}\big)$$

Therefore, (r,t,T) is a min-period retiming if and only if $P(r,t,T) \wedge P5(r,t,T)$, i.e., among all valid retimings–those satisfying P–, the current (r,t,T) has the minimum T_{max}.

3. ALGORITHMS

We first present an algorithm in Section 3.1 for finding a retiming that satisfies the hold constraint. We start with the original circuit, i.e., $r(v) = 0$, $\forall v \in V$. While some of $P0$-$P4$ is violated, we will compare the current r with an imagined valid retiming and increase r to approach it.

Section 3.2 explains how T_{max} can be reduced by increasing r to approach an imagined optimal retiming. Since it only increases r, which is consistent with the r adjustments in the algorithm for finding a valid retiming, we combine them to yield a min-period retiming algorithm.

3.1 Retiming for hold constraint

Let $(\bar{r},\bar{t},\bar{T})$ be a valid retiming, i.e., $P(\bar{r},\bar{t},\bar{T})$. It should be noted that $(\bar{r}+c,\bar{t},\bar{T})$, where $c \in \mathcal{Z}$ is an arbitrary constant, is also a valid retiming. Therefore in the remainder of this paper, we assume that a valid retiming should satisfy

$$\big(\forall v \in V : \bar{r}(v) \geq 0\big) \wedge \big(\exists v \in V : \bar{r}(v) = 0\big).$$

In addition, since $(\bar{r},\bar{t},\bar{T})$ satisfies $P1$, there is an upper bound for $w_r(p)$ on any simple path p. This is stated in the following lemma.

LEMMA 1. *For any valid retiming $(\bar{r},\bar{t},\bar{T})$, we have $w_{\bar{r}}(p) \leq |V|$ for every simple path or simple cycle p.*

The next result is a corollary of the above lemma.

COROLLARY 1.1. *For any valid retiming $(\bar{r},\bar{t},\bar{T})$, we have $\big(\forall v \in V : \bar{r}(v) \leq |V|\big)$.*

PROOF. For the sake of contradiction, we assume that $\bar{r}(v) > |V|$ for some $v \in V$. Let u be the vertex such that $\bar{r}(u) = 0$. Since G is strongly connected, consider a simple path p from u to v, we have $w_{\bar{r}}(p) = w(p) + \bar{r}(v) - \bar{r}(u) = w(p) + \bar{r}(v)$. By Lemma 1, $w_{\bar{r}}(p) \leq |V|$, thus $\bar{r}(v) \leq |V| - w(p) \leq |V|$. $\square$

Based on the above corollary, we can define a necessary condition for a valid retiming as

$$B(r) \stackrel{\triangle}{=} \big(\forall v \in V : 0 \leq r(v) \leq |V|\big) \wedge \big(\exists v \in V : r(v) = 0\big)$$

Thus, $B(\bar{r})$ is true for any valid retiming $\bar{r}$.

To reach $(\bar{r},\bar{t},\bar{T})$, we start with the original circuit, i.e., $r(v) = 0$, $\forall v \in V$, which trivially satisfies $P0$. If $P1$ is not satisfied under the current r, we have $w_r(u,v) > 1 \geq w_{\bar{r}}(u,v)$ for some $(u,v) \in E$, i.e.,

$$r(v) - r(u) > \bar{r}(v) - \bar{r}(u) \tag{1}$$

We need to remove at least $w_r(u,v) - 1$ registers out of (u,v) by increasing $r(u)$. However, the increase of $r(u)$ may violate $P0$ on other edges (u,x) going out of u if $w_r(u,x) = 0$ before the increase. In other words, we have $w_r(u,x) < 0 \leq w_{\bar{r}}(u,x)$ after the increase of $r(u)$, i.e.,

$$\bar{r}(x) - \bar{r}(u) > r(x) - r(u) \tag{2}$$

We then increase $r(x)$ by the same amount to restore $P0$ on (u,x). The above process is iterated until both $P0$ and $P1$ are satisfied on all edges.

After that, we find t to satisfy $P2$ and $P3$ by shortest path computation. In addition, we keep a label $f : V \to V$ such that a shortest combinational path to v starts from $f(v)$. We do not consider T since we only focus on the hold constraint in this section.

What remains is to check t against $P4$. If it is violated, it means that $t(u) < H$ for some $(u,v) \in E$ with $w_r(u,v) = 1$. Let $z = f(u)$. By the definition of f, a shortest combinational path p to u starts from z. In addition, there is at least one edge $(y,z) \in E$ such that $w_r(y,z) = 1$, otherwise p cannot be the shortest.

To fix the hold violation at u, we can either remove the register on (u,v), or increase $t(u)$ by moving the register on (y,z) further ahead. In general, the number of registers on the path $q = \{(y,z)\} \cup p \cup \{(u,v)\}$ should be less than 2, i.e., $w(q) + r(v) - r(y) = 2 > w(q) + \bar{r}(v) - \bar{r}(y)$, or

$$r(v) - r(y) > \bar{r}(v) - \bar{r}(y) \tag{3}$$

We will move r closer to $\bar{r}$ by increasing $r(y)$ by 1.

The pseudocode for finding a valid retiming is given in Figure 1.

```
ValidHold(G,r,t)

While (B(r) ∧ ¬P(r,t,∞)) do
  ▷Satisfy P0 and P1
  While (¬P0(r) ∨ ¬P1(r)) do
    r(v) ← r(v) − w_r(u,v) if w_r(u,v) < 0;
    r(u) ← r(u) + w_r(u,v) − 1 if w_r(u,v) > 1;
  ▷Computer t to satisfy P2 and P3
  t ← ∞;
  While (¬P2(r,t,∞) ∨ ¬P3(r,t,∞)) do
    If (w_r(u,v) = 1 ∧ t(v) > d(v)) then
      t(v), f(v) ← d(v), v;
    If (w_r(u,v) = 0 ∧ t(v) > t(u) + d(v)) then
      t(v), f(v) ← t(u) + d(v), f(u);
  ▷Fix P4
  If P4(r,t) is violated on (u,v) then
    r(y) ← r(y) + 1 if w_r(y, f(u)) = 1;
```

Figure 1: Pseudocode of finding a valid retiming.

The next lemma establishes an invariant during the execution of "ValidHold".

LEMMA 2. *If $r \le \bar{r}$, where $\bar{r}$ is a valid retiming, then $r \le \bar{r}$ after the execution of "ValidHold" in Figure 1.*

PROOF. During "ValidHold", r is increased only when we attempt to satisfy $P0$ or $P1$, or to fix $P4$.

The first one happens when $(\exists (u,v) \in E : w_r(u,v) < 0)$. Let $r'(v) = r(v) - w_r(u,v) = r(v) - (w(u,v) + r(v) - r(u)) = r(u) - w(u,v)$. It follows that $r'(v) \le \bar{r}(u) - w(u,v)$ since $r(u) \le \bar{r}(u)$ due to $r \le \bar{r}$. Given that $w_{\bar{r}}(u,v) = w(u,v) + \bar{r}(v) - \bar{r}(u) \ge 0$, we have $\bar{r}(u) - w(u,v) \le \bar{r}(v)$. Therefore, $r'(v) \le \bar{r}(v)$, i.e., $r \le \bar{r}$ is kept after the increase of $r(v)$.

The second one happens when $(\exists (u,v) \in E : w_r(u,v) > 1)$. Let $r'(u) = r(u) + w_r(u,v) - 1 = r(u) + w(u,v) + r(v) - r(u) - 1 = w(u,v) + r(v) - 1$. It follows that $r'(u) \le w(u,v) + \bar{r}(v) - 1$ since $r(v) \le \bar{r}(v)$ due to $r \le \bar{r}$. Given that $w_{\bar{r}}(u,v) = w(u,v) + \bar{r}(v) - \bar{r}(u) \le 1$, we have $w(u,v) + \bar{r}(v) - 1 \le \bar{r}(u)$. Therefore, $r'(u) \le \bar{r}(u)$, i.e., $r \le \bar{r}$ is kept after the increase of $r(u)$.

When the last one happens, (3) is true. Given that $r \le \bar{r}$, we have $r(y) < \bar{r}(y)$, otherwise $r(y) = \bar{r}(y)$, thus $r(v) > \bar{r}(v)$, which contradicts $r \le \bar{r}$. Therefore, $r \le \bar{r}$ holds after $r(y)$ is increased by 1.

Since none of the above three violates $r \le \bar{r}$, it is kept after the execution of "ValidHold". □

Based on the above lemma, we can establishes the correctness of "ValidHold".

THEOREM 1. *Given that $r(v) \ge 0, \forall v \in V$, the procedure "ValidHold" terminates in $O(|V|^2|E|)$ time. It finds a valid retiming $\bar{r}$ with $r \le \bar{r}$, if such an $\bar{r}$ exists.*

PROOF. The procedure terminates only if $\neg B(r)$ or $P(r,t,\infty)$. For either case, the total number of r increases will not be larger than $|V|^2$ since we start with $r(v) \ge 0$, $\forall v \in V$. Given that satisfying $P2$ and $P3$ can be done in $O(|E|)$ time [10], the amount of time between two consecutive r increases is $O(|E|)$. Therefore, "ValidHold" will terminate in $O(|V|^2|E|)$ time.

If there exists a valid retiming $\bar{r}$ with $r \le \bar{r}$, then Lemma 2 guarantees that $r \le \bar{r}$ upon termination, which implies $B(r)$. Therefore, the procedure is terminated due to $P(r,t,\infty)$, i.e., it finds a valid retiming. □

As a result, we can apply "ValidHold" to the original circuit to obtain a valid retiming, if it exists.

3.2 Min-period retiming (for setup constraint)

Once we obtain a valid retiming r, we can find T to satisfy $P2$ and $P3$ by longest path computation. Similar to f, we introduce a label $F : V \to V$ such that a longest combinational path to v starts from $F(v)$.

Suppose that the valid retiming (r,t,T) we obtained via "ValidHold" from the original circuit is not optimal, i.e., $\neg P5(r,t,T)$. Let (r^*,t^*,T^*) be an optimal retiming. Lemma 2 guarantees that $r \le r^*$. In this section we extend the idea in [27] to realize r^* by adjusting r.

Since $T^*_{max} < T_{max}$, there exists a vertex $v \in V$ such that $T(v) = T_{max} > T^*_{max} \ge T^*(v)$. Let $p = F(v) \rightsquigarrow v$ be a longest combinational path to v. The fact that $T^*(v) < T(v)$ implies that there must be at least one register on p in (r^*,t^*,T^*), i.e., $w_{r^*}(p) > 0 = w_r(p)$, or equivalently,

$$r^*(v) - r^*(F(v)) > r(v) - r(F(v)) \tag{4}$$

We can add more registers on p by increasing $r(v)$. The amount of increase should only be 1 since we do not want to over-adjust r. After that, we apply "ValidHold" to restore $P0$-$P4$. It is interesting to notice that increasing r to approach r^* is consistent with the r adjustments in "ValidHold", thus they can be combined smoothly.

The pseudocode for finding an optimal retiming is presented in Figure 2.

The following theorem establishes the correctness of our algorithm.

THEOREM 2. *The algorithm in Figure 2 terminates in $O(|V|^2|E|)$ time with an optimal retiming satisfying setup and hold constraints under the minimum clock period, or with a report that there is no valid retiming at all.*

PROOF. The algorithm terminates only if $\neg B(r)$. Since we start with $r(v) = 0$, $\forall v \in V$, the total number of r increases is no larger than $|V|^2$. Given that satisfying $P2$ and $P3$ can be done in $O(|E|)$ time, which is also the time between two consecutive r increases, the algorithm terminates in $O(|V|^2|E|)$ time.

The algorithm first attempts to find a valid retiming from the original circuit. If it fails, then, by Theorem 1, we know that there is no valid retiming at all.

Otherwise, we have $r \le r^*$ before we enter the outer while loop. We use r^{opt} to record the valid retiming with the smallest clock period ϕ^{opt} we found so far. If ϕ^{opt} is not the minimum, then (4) is true. It implies that $r(v) < r^*(v)$, otherwise $r(v) = r^*(v)$, thus $r^*(u) < r(u)$, which contradicts $r \le r^*$. Therefore, $r \le r^*$ is kept after $r(v)$ is increased by 1. Together with Lemma 2, $r \le r^*$ is kept before we reach r^*. The fact that $\neg B(r)$ upon termination implies

```
Algorithm  Min-Period Retiming
Input:  A circuit G = (V,E,d,D,w),
        setup time S, hold time H.
Output:  An optimal retiming r^opt satisfying
        setup and hold constraints under
        minimum clock period φ^opt.

  r ← 0;
  ValidHold(G,r,t);
  Return ''No valid retiming'' if ¬B(r);
  φ^opt ← ∞;
  While (B(r)) do
    ▷Compute T to satisfy P2 and P3
    T ← 0;
    While (¬P2(r,t,T) ∨ ¬P3(r,t,T)) do
      If (w_r(u,v) = 1 ∧ T(v) < D(v)) then
        T(v),F(v) ← D(v),v;
      If (w_r(u,v) = 0 ∧ T(v) < T(u)+D(v)) then
        T(v),F(v) ← T(u)+D(v),F(u);
    ▷Update φ^opt and r^opt
    φ^opt, r^opt ← T_max, r if T_max < φ^opt;
    ▷Adjust r to reduce φ^opt
    r(v) ← r(v)+1 for some v with T(v) ≥ φ^opt;
    ValidHold(G,r,t);
  ▷Take into account setup time
  φ^opt ← φ^opt + S;
  Return r^opt and φ^opt;
```

Figure 2: Pseudocode of retiming algorithm.

that we have already reached r^* and went beyond it. Consequently, the value recorded in ϕ^{opt} is the minimum clock period. □

3.3 Additional termination criterion

From Section 3.2, we know that if a valid retiming is not optimal, then (4) is true and we will increase $r(v)$ by 1. A key observation is that if $r\big(F(v)\big)$ is increased before the next increase of $r(v)$, then another increase of $r(v)$ is necessitated because the increase of $r\big(F(v)\big)$ cancels the previous increase of $r(v)$ and makes (4) true again. The relation between $r\big(F(v)\big)$ and $r(v)$ is similar to that between $T\big(F(v)\big)$ and $T(v)$ where any increase of $T\big(F(v)\big)$ will be propagated to $T(v)$ unless the path between them ceases to be the longest.

In fact, the same relation exists between $r(v)$ and $r(u)$ when (1) is true, and between $r(u)$ and $r(x)$ when (2) is true, and between $r(v)$ and $r(y)$ when (3) is true.

Therefore, we introduce another label $m : V \to V \cup \{\emptyset\}$, where $\emptyset$ is the default assignment, and define it as follows. If $r(u)$ is increased due to (1), then $m(u)$ is set to v; if $r(x)$ is increased due to (2), then $m(x)$ is set to u; if $r(y)$ is increased due to (3), then $m(y)$ is set to v; if $r(v)$ is increased due to (4), then $m(v)$ is set to $F(v)$.

Based on the definition of m-labeling, we can formulate the relation between $r\big(m(i)\big)$ and $r(i)$ for all $i \in V$ in the following lemma.

LEMMA 3. *It is true before we reach an optimal retiming r^* that*

$$\Big(\forall i \in V, m(i) \in V : r^*\big(m(i)\big) - r\big(m(i)\big) \le r^*(i) - r(i)\Big).$$

PROOF. By the definition of the m-labeling, $m(i) \in V$ only if $r(i)$ has ever been increased from 0, $\forall i \in V$. For a particular vertex i, we will show that the inequality is kept after the first increase of $r(i)$ and continues to hold before an optimal r^* is reached.

Consider the first time that $r(i)$ is increased. Suppose it is due to (1) on edge (i, j). Then $m(i)$ will be set to j, and (1) becomes $r\big(m(i)\big) - r(i) > r^*\big(m(i)\big) - r^*(i)$, or $r^*\big(m(i)\big) - r\big(m(i)\big) < r^*(i) - r(i)$. After $r(i)$ is increased, we have $r^*\big(m(i)\big) - r\big(m(i)\big) \le r^*(i) - r(i)$. Similar arguments apply to other cases due to the occurrences of (2)-(4).

Even if $r\big(m(i)\big)$ may be increased thereafter, the inequality will remain true until the next increase of $r(i)$, when $m(i)$ will be assigned to a new vertex which may or may not differ from the previous assignment. By the same case study as above, we can show that the inequality will continue to hold. By induction, the lemma is true. □

In addition, if there exists a sequence of vertices x_i, $i = 0, 1, \ldots, k-1$, such that $x_i = m(x_{i+1})$ and $x_k = x_0$, we refer to it as an m-cycle. The next theorem shows that the appearance of an m-cycle is an evidence that we have reached the optimal clock period.

THEOREM 3. *If an m-cycle appears, then we have reached the optimal clock period.*

PROOF. Suppose the last m assignment is $m(x_0) = x_{k-1}$, by setting which the m-labeling forms a cycle, i.e., a sequence of vertices x_i, $i = 0, 1, \ldots, k-1$, such that $x_i = m(x_{i+1})$ and $x_k = x_0$.

For the sake of contradiction, we assume that an optimal r^* is not reached yet. Therefore, by Lemma 3 and $m(x_0) = x_{k-1}$, we have $r^*(x_{k-1}) - r(x_{k-1}) \le r^*(x_0) - r'(x_0)$ after $r(x_0)$ is increased to $r'(x_0)$ due to any of (1)-(4). It implies that $r^*(x_{k-1}) - r(x_{k-1}) < r^*(x_0) - r(x_0)$ before the increase. On the other hand, Lemma 3 guarantees that $r^*\big(m(x_i)\big) - r\big(m(x_i)\big) \le r^*(x_i) - r(x_i)$, $1 \le i \le k-1$. It follows that $r^*(x_0) - r(x_0) \le r^*(x_{k-1}) - r(x_{k-1})$, which is a contradiction. Therefore, an optimal retiming is already reached. □

Based on Theorem 3, the algorithm will terminate if an m-cycle is found. Since checking m-cycle is carried out after every r increase and the total number of r increases is no larger than $|V|^2$, the complexity of checking m-cycle is at most $O(|V|^3)$. Therefore, adding m-cycle checking in the algorithm will not affect the worst case complexity $O(|V|^2|E|)$ in Theorem 2.

4. EXAMPLE

In this section we use an example in Figure 3 to show how the proposed algorithm finds the minimum period and builds m-labeling to certify the optimality.

Figure 3(a) shows the original sequential circuit (taken from [17]). It has five combinational logic blocks connected in a ring and two edge-triggered registers. The pair of integers in each block gives the maximum and the minimum propagation delays of the data through that block. For example, whenever data propagate through block A, they always require at least 1 time unit and never more than 10 time units. For simplicity, each register is assumed to have zero setup time and a hold time H of 4. The steps of the algorithm are shown by the sequence of figures (b)-(f) in Figure 3.

Starting with the original circuit, i.e., $r(A) = r(B) = r(C) = r(D) = r(E) = 0$, "ValidHold" is called. Since no edge has more than one register, $P0$ and $P1$ are satisfied. Then it calculates the earliest output arrival time of each block, which is shown beside each block. Since none of the earliest arrival times is smaller than H, the current r is a valid retiming. The algorithm proceeds to compute the latest output arrival time of each block and assign the maximum of them, 60, to ϕ^{opt}.

In order to reduce ϕ^{opt}, any block whose output arrival time is at least ϕ^{opt} will have its r value increased. In our case we have

$T(C) = \phi^{opt}$, thus $r(C)$ is increased by 1, which means that a register is moved from the output of C to its input. This increment is accompanied by setting $m(C) = F(C) = A$ (because a longest combinational path to C starts from A), which is represented by a dotted edge in Figure 3(b). After that, "ValidHold" is called again. In this case, a hold violation at B is identified since $t(B) = 3 < H = 4$.

To fix the violation, the algorithm increases $r(E)$ by 1. In other words, the register on edge (E,A) is moved ahead to increase $t(B)$. As an accompaniment, $m(E)$ is set to C to indicate that future increases in $r(C)$ will be propagated to $r(E)$ to avoid hold violation on path from E to C. It turns out that the hold violation is fixed and we obtain another valid retiming whose $T_{max} = 50$, as shown in Figure 3(c). Therefore, ϕ^{opt} is updated to 50.

Next, the algorithm attempts to reduce ϕ^{opt} by successively increasing $r(B)$, $r(D)$ and $r(C)$ in Figure 3(d), 3(e), and 3(f) respectively until the appearance of an m-cycle. It is an evidence that there is no valid retiming with a period less than 50, hence 50 is the optimal period and it is realized in Figure 3(c).

It can be seen from the example that m-labeling can only form a forest before the optimal period is reached. In addition, the process of adjusting r to improve ϕ^{opt} and the process of building m to certify optimality are carried out simultaneously. As a result, the algorithm can keep reducing ϕ^{opt} until the optimality is certified in the last iteration.

5. EXPERIMENTAL RESULTS

We implemented the algorithm on a Sun Ultra 10 machine. Our test files were generated from ISCAS-89 benchmark suite using ASTRA [18]. Each gate was assigned a maximum delay equal to the number of fanouts or an upper bound 100, whichever is smaller. The minimum delay was equal to the maximum delay. For simplicity, we assumed zero setup time. Hold time H was set to 2.

Reported in Table 1 are results for large circuits. Column "T_{max}" lists the maximum combinational delays of the original circuits. Note that T_{max} may not be a feasible clock period since the original circuit may have more than one register on an edge. The minimum periods by retiming when only setup constraint is considered are shown in column "ϕ_S". Taking both setup and hold constraints into account, our proposed algorithm computes the minimum periods in column "ϕ_{SH}", where we use "NO" to indicate that there is no valid retiming. Cases with $\phi_S \neq \phi_{SH}$ are highlighted. For runtime comparison, we obtained the source code of the algorithm in [27], which considers only setup constraint for minimum period retiming, and reported the runtime in column " [27]". The runtime of our algorithm is listed in column "ours". We then compute the ratio between them for each case[1] and obtain the arithmetic (geometric) mean of all the ratios in row "arith" ("geo").

Table 1 reveals two things. Firstly, more than half of the circuits have $\phi_{SH} \neq \phi_S$. In other words, the minimum period retimings given by the algorithm in [27] for these circuits have hold violations. The difference between ϕ_{SH} and ϕ_S could be significant even for small $H = 2$, e.g., $(\phi_{SH} - \phi_S)/\phi_S = 53.8\%$ for "s838.1". Two circuits "s13207.1" and "s38584.1" do not even have a valid retiming because of reconvergent paths. It happens when one path from u to v has extra registers and thus requires $r(u) > r(v)$ while another path from u to v has no register and requires $r(u) \leq r(v)$, which contradict each other. Secondly, the proposed algorithm is efficient. Although it checks both setup and hold constraints, it is as efficient as the algorithm in [27] on average.

6. CONCLUSION

[1]If the runtime of one algorithm is "0.00", the case is ignored.

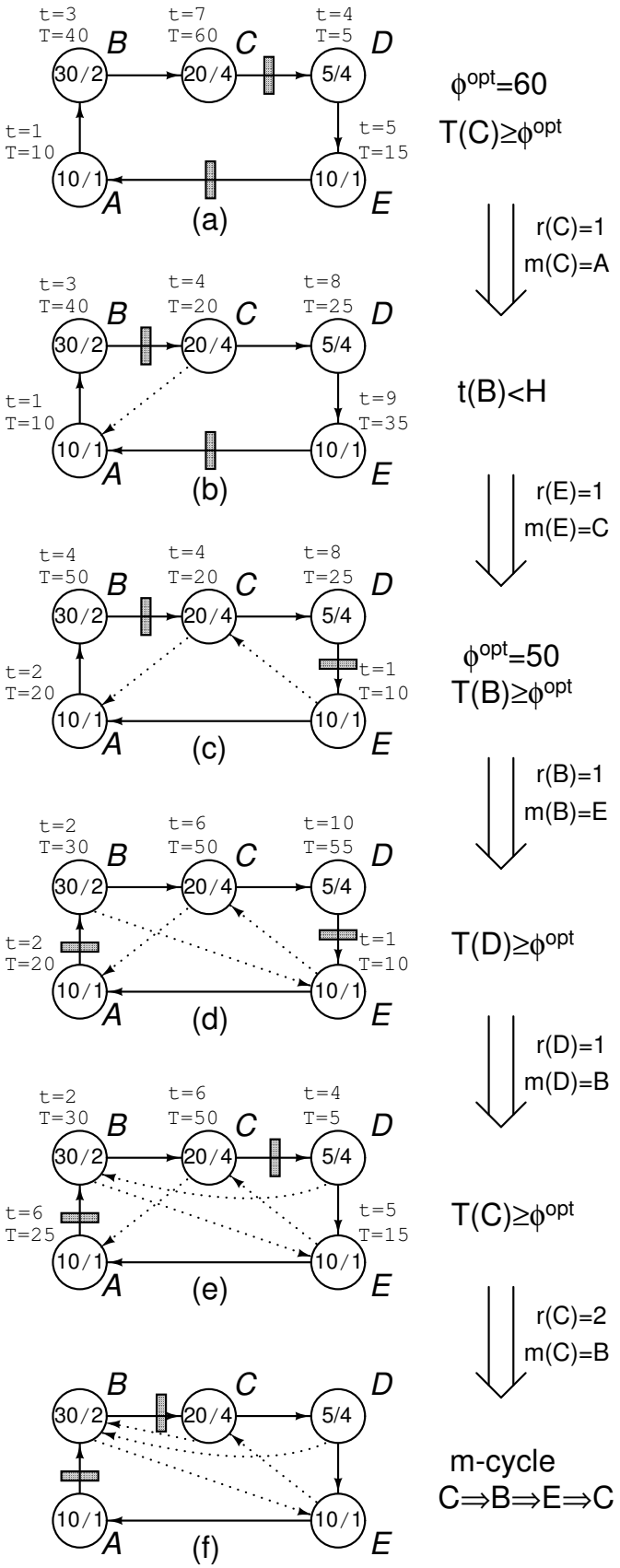

Figure 3: Applying the proposed algorithm on an example.

Table 1: Experimental Results

name	#gates	T_{max}	period		time(s)	
			ϕ_S	ϕ_{SH}	[27]	ours
s838.1	288	94	52	**80**	0.00	0.00
s1238	428	110	110	110	0.01	0.00
s1423	490	332	254	**280**	0.01	0.00
s1494	558	166	164	**166**	0.01	0.01
s5378	1004	92	92	92	0.01	0.01
s9234	2027	178	162	162	0.08	0.10
s9234.1	2027	178	162	162	0.09	0.11
s13207.1	2573	286	270	**NO**	0.08	0.01
s15850	3448	372	154	**210**	0.17	0.21
s15850.1	3448	372	290	290	0.17	0.20
s35932	12204	138	124	**138**	1.23	0.84
s38417	8709	220	112	**120**	0.32	0.75
s38584.1	11448	306	290	**NO**	0.37	0.03
arith					1	1.01X
geo					1	0.72X

A new algorithm is presented for retiming edge-triggered circuits to achieve the minimum clock period under both setup and hold constraints. The worst-case running time of the algorithm is $O(|V|^2|E|)$, which is asymptotically more efficient than the best known result $O(|V|^3|E|\lg|V|)$ in [17]. Experimental results confirm the efficiency of the algorithm.

7. REFERENCES

[1] C. Chu, E. F. Y. Young, D. K. Y. Tong, and S. Dechu. Retiming with interconnect and gate delay. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 221–226, 2003.

[2] P. Cocchini. Concurrent flip-flop and repeater insertion for high performance integrated circuits. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 268–273, 2002.

[3] J. Cong, H. Li, and C. Wu. Simultaneous circuit partitioning/clustering with retiming for performance optimization. In *Proc. of the Design Automation Conf.*, pages 460 – 465, 1999.

[4] J. Cong and X. Yuan. Multilevel global placement with retiming. In *Proc. of the Design Automation Conf.*, pages 208–213, Anaheim, CA, 2003.

[5] S. Dey and S. Chakradhar. Retiming sequential circuits to enhance testability. In *Proceedings of the 12th IEEE VLSI Test Symposium*, pages 28–33, April 1994.

[6] G. Even, I. Y. Spillinger, and L. Stok. Retiming revisited and reversed. *IEEE Transactions on Computer Aided Design*, 15(3):348–357, March 1996.

[7] S. Hassoun and C. J. Alpert. Optimal path routing in single- and multiple-clock domain systems. *IEEE Transactions on Computer Aided Design*, 22(11):1580– 1588, November 2003.

[8] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. Optimizing two-phase, level-clocked circuitry. *Journal of the ACM*, 44(1):148–199, 1997.

[9] K. N. Lalgudi and M. C. Papaefthymiou. An efficient tool for retiming with realistic delay modeling. In *Proc. of the Design Automation Conf.*, San Francisco, CA, June 1995.

[10] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing Synchronous Circuitry by Retiming. In *Advanced Research in VLSI: Proc. of the Third Caltech Conf.*, pages 86–116, Rockville, MD, 1983. Computer Science Press.

[11] C. Lin and H. Zhou. Optimal wire retiming without binary search. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 452–458, 2004.

[12] C. Lin and H. Zhou. Wire retiming as fixpoint computation. *IEEE Transactions on Very Large Scale Integration Systems*, 13(12):1340–1348, December 2005.

[13] B. Lockyear and C. Ebeling. The practical application of retiming to the design of high performance systems. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 288–295, 1993.

[14] N. Maheshwari and S. S. Sapatnekar. Optimizing large multi-phase level-clocked circuits. *IEEE Transactions on Computer Aided Design*, 18(9):1249–1264, September 1999.

[15] S. Malik, E. M. Sentovich, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Retiming and Resynthesis: Optimization of Sequential Networks with Combinational Techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 10(1):74–84, January 1991.

[16] P. Pan, A. K. Karandikar, and C. L. Liu. Optimal clock period clustering for sequential circuits with retiming. *IEEE Transactions on Computer Aided Design*, 17(6):489–498, June 1998.

[17] M. C. Papaefthymiou. Asymptotically efficient retiming under setup and hold constraints. In *Proc. Intl. Conf. on Computer-Aided Design*, 1998.

[18] S. S. Sapatnekar and R. B. Deokar. Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits. *IEEE Transactions on Computer Aided Design*, 15(10):1237–1248, October 1996.

[19] E. M. Sentovich and R. K. Brayton. Preserving Don't Care Conditions During Retiming. In *Proc. Intl. Conf. on VLSI*, pages 461–470, August 1991.

[20] F. Sheikh, A. Kuehlmann, and K. Keutzer. Minimum-power retiming for dual-supply cmos circuits. In *ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2002.

[21] N. Shenoy and R. Rudell. Efficient implementation of retiming. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 226–233, 1994.

[22] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Retiming of circuits with single phase level-sensitive latches. In *Proc. Intl. Conf. on Computer Design*, 1991.

[23] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Minimum padding to satisfy short path constraints. In *Proc. Intl. Conf. on Computer-Aided Design*, 1993.

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

[25] A. Tabbara, B. Tabbara, R. K. Brayton, and A. R. Newton. Integration of retiming with architectural floorplanning. *INTEGRATION, the VLSI journal*, 29:25–43, 2000.

[26] D. K. Y. Tong and E. F. Y. Young. Performance-driven register insertion in placement. In *International Symposium on Physical Design*, pages 53–60, 2004.

[27] H. Zhou. Deriving a new efficient algorithm for min-period retiming. In *Proc. Asian and South Pacific Design Automation Conference*, 2005.

[28] H. Zhou and C. Lin. Retiming for wire pipelining in system-on-chip. *IEEE Transactions on Computer Aided Design*, 23(9):1338–1345, September 2004.