A New Approach for Integration of Min-Area Retiming and Min-Delay Padding for Simultaneously Addressing Short-path and Long-path Constraints [*]

Vijay Sundararajan, Sachin S. Sapatnekar, Keshab K. Parhi
Dept. of ECE University of Minnesota
Minneapolis, MN 55455
E-mail: {vijay, sachin, parhi}@ece.umn.edu

April 1, 2000

Abstract

This paper describes a polynomial time algorithm for min-area retiming for edge-triggered circuits to handle *both* setup and hold constraints. Given a circuit G and a target clock period c, our algorithm either outputs a retimed version of G satisfying setup and hold constraints or reports that such a solution is not possible, in $O(|V|^3 log|V| log(|V|C))$ steps, where $|V|$ corresponds to number of gates in the circuit and C is equal to the number of registers in the circuit. This is the first polynomial time algorithm ever reported for min-area retiming with constraints on both long and short-paths. An alternative problem formulation that takes practical issues in to consideration and lowers the problem complexity is also developed. Both the problem formulations have many parallels with the original formulation of *long path only* retiming by Leiserson and Saxe and all the speed improvements that have been obtained on that problem statement are also demonstrated in simulation for the approach presented here. Finally, a basis is provided for deriving efficient heuristics for addressing both long path and short-path requirements by combining the techniques of retiming and min-delay padding.

1 Introduction

The procedure of moving flip-flops around a VLSI circuit, while maintaining its functionality, to optimize a performance objective for the circuit is known as retiming [1]. Retiming was intro-

[*]This research was supported in parts by the Army Research Office under grant number DA/DAAG55-98-1-0315 and a grant from Intel Corporation.

duced as an optimization technique for edge-triggered circuits [1] but has since been extended to handle level-clocked circuits, [2] (see [3] for detailed list of references). Further extensions of retiming now include logic synthesis, power optimization and testability [4, 5]. The execution time required to perform retiming has been drastically reduced in [6, 7].

The thrust of early research firmly established the far-reaching effectiveness of retiming as an optimization strategy for sequential VLSI circuits [1, 4, 5]. These techniques, however, included only the setup time constraints for flip-flops in the circuit and ignored the hold time constraints. Hold-time violations were traditionally corrected by padding delay buffers in violating short-paths using techniques such as [8]. However, with short-paths becoming increasingly prominent in deep submicron circuits, the number of such buffers may become inordinately large and there is a need to incorporate hold-time constraints directly into a retiming formulation. The ideal situation is to be able to combine the techniques of padding delay-buffers and retiming in to a unified framework for addressing both long paths and short-paths simultaneously. Recently, a new retiming strategy was presented [3] which could solve min-period retiming for VLSI circuits with both setup and hold constraints in polynomial time using a strategy known as integer monotonic programming. However, there is no straightforward extension of this technique to incorporate min-area retiming, and the nature of the constraints in [3] do not lend themselves to a fast network flow formulation for a linear objective function.

In this paper we present a novel technique that performs min-period and min-area retiming of edge-triggered circuits with setup and hold constraints. The problem formulation used to perform min-area retiming is similar to the original retiming framework in [1], but with constraints to satisfy both long path and short-path timing requirements. An alternative problem formulation is then developed that lowers the problem complexity to make it practical for large circuits. As in the result in [3], we assume that all flip-flops in the circuit have identical values for the setup and hold times.

Given any edge-triggered sequential circuit G, a target clock period c, a setup time S, and a hold time H, our algorithm computes a retimed circuit G_r in which all long path and all short-path constraints are satisfied. If the problem is infeasible for the given input, it reports so and terminates. The worst case execution time of our technique is bounded by $O(|V|^3 log|V| log(|V|C))$.

The chief advantages of our technique over the technique in [3] are:

1) An efficient solution to the min-area retiming problem is facilitated.

2) The problem formulation is similar to [1] and hence speedup techniques for min-period constraints in [6, 7, 9, 10] are admissible.

The new technique presented in this paper is then used in conjunction with min-delay padding to obtain a basis for obtaining lower area retiming solutions as opposed to the "traditional" long path only retiming, followed by min-delay padding based approaches [8]. It is demonstrated that by a judicious combination of the new retiming approach presented here and min-delay padding the number of buffers required to address short-path violations can be reduced with a minimal increase in the number of flip-flops. The run-time requirements for the new approach in this paper are comparable to fast implementations of traditional retiming [6, 7].

The rest of the paper is organized as follows. We demonstrate the need to include hold constraints as a part of the retiming problem in section 2. A graph model for the problem is developed in section 3. In section 4 we formulate a new set of constraints to guarantee satisfaction of hold time requirements. Additionally, we describe an alternative practical approach to solve the min-area retiming problem that considerably lowers the problem complexity. An approach for combining min-delay padding and retiming to address the problem of satisfying setup and hold time requirements is presented in section 5. In the same section, a description of a basis is presented for deriving useful heuristics for addressing the problem of satisfying both short-path and long path constraints using the retiming formulation in this paper and min-delay padding, along with a new concept called buffer-relocation. Simulation results comparing traditional approaches to retiming and a heuristic derived using techniques introduced here are presented in section 6. Finally, conclusions and directions for future work are presented in section 7.

2 Motivational Example

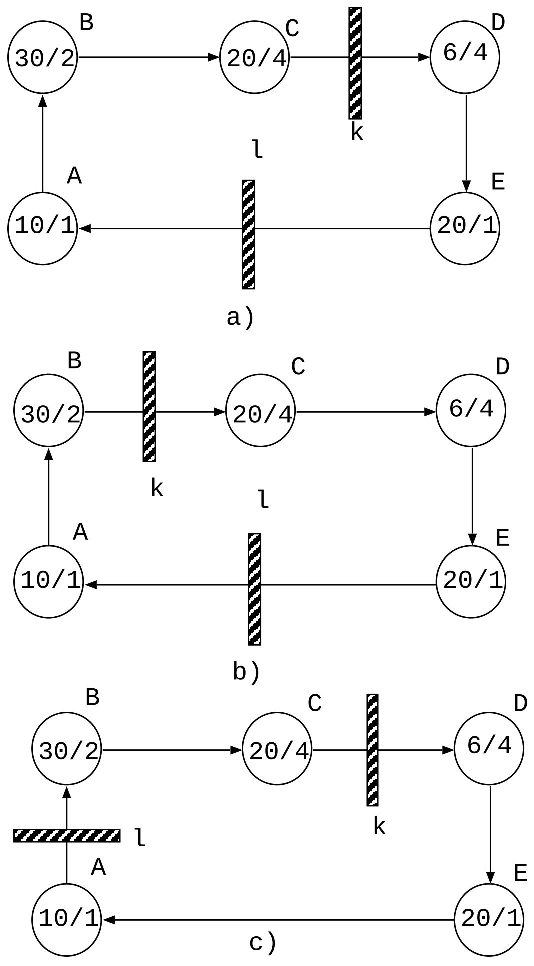

Figure 1: (a) Original circuit. (b) Retimed circuit with hold violation. (c) Retimed circuit with no timing violations.

In this section we reproduce an example from [3] to demonstrate that the solution to the retiming problem under setup and hold time constraints differs from that obtained by conventional retiming algorithms that consider only the setup time constraints.

The sequential circuit shown in Fig. 1(a) 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 minimum propagation delay 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 and wire is assumed to have zero delay, and in addition each register is assumed to have zero setup time, and a hold time of 4. Moreover, clock skew is assumed to be zero. Thus, the shortest clock period achievable by this circuit is $10+30+20 = 60$, corresponding to the longest combinational path, ABC. There are no hold violations in this circuit since the minimum propagation delays along ABC and DE are 7 and 5, respectively, both exceeding the register hold time.

The retimed circuit in Fig. 1(b) is obtained by shifting the register k across block C and is functionally equivalent to the one in Fig. 1(a). This circuit can be computed by applying the retiming algorithm in [1] with a target clock period of 46. Since the longest combinational path in this circuit is CDE with a delay of $20 + 6 + 20 = 46$, there are no setup violations in the retimed circuit. The minimum delay along the path AB is $1+2 = 3 < 4$; therefore, a fast signal propagating along AB can contaminate the inputs of register k and result in erroneous circuit operation.

The retimed circuit in Fig. 1(c) satisfies all setup and hold constraints and achieves a clock period of 50. This is the shortest clock period that can be achieved by retiming the original circuit. Our algorithm will find such an optimal solution in polynomial time.

3 Preliminaries

In this section we will present a graph model to illustrate some background information on retiming. We model an edge-triggered circuit as a directed multi-graph $G = (V, E, d, \delta, w, \sigma)$. For the purpose of compatibility, this model is identical to the one in [3]. Each vertex u in the vertex-set V corresponds to a combinational logic block. The nonnegative weights $d(u)$ and $\delta(u)$ associated with each vertex u denote the maximum and minimum data propagation delay through u, respectively. Each edge $u \xrightarrow{e} v$ in the edge-set E corresponds to a wire from u to v in the circuit. All wires are assumed to have zero delay. For each edge $u \xrightarrow{e} v$, the nonnegative integer edge-weight $w(e)$ denotes the register count of the corresponding wire from $u \xrightarrow{e} v$. All registers in the circuit are assumed to have equal positive setup times S and equal positive hold times H. Without loss of generality, register delays can be assumed to be zero.

In addition to the register count $w(e)$, each edge $u \xrightarrow{e} v$ is associated with a weight $\sigma(e)$ that represents the delay in the propagation of the clock signal from the clock source to the wire denoted by e. Clock skew is assumed to be *monotonic*, that is, the "effective delay" of a path increases with the number of combinational gates on it. Given a path $r \xrightarrow{e_i} u \overset{p}{\rightsquigarrow} v \xrightarrow{e_j} y$, where p is combinational, $w(e_i) \geq 1$, and $w(e_j) \geq 1$, the *minimum effective delay* of p is given by the expression $\Delta(p) + \sigma(e_i) - \sigma(e_j)$, where $\Delta(p) = \sum_{x \in p} \delta(x)$. Clock skew monotonicity is ensured if for each edge pair $r \xrightarrow{e_i} u$, $u \xrightarrow{e_j} y$ in E, we have,

$$\delta(u) + \sigma(e_i) - \sigma(e_j) \geq 0. \tag{1}$$

A retiming of an edge-triggered circuit $G = (V, E, d, \delta, w, \sigma)$ is an integer-valued vertex-labeling $r : V \to Z$. This labeling denotes a transformation of the original circuit G into a functionally equivalent circuit $G_r = (V, E, d, \delta, w_r, \sigma)$, where for each edge $u \xrightarrow{e} v$ in G, w_r is defined by the equation,

$$w_r(e) = w(e) + r(v) - r(u). \tag{2}$$

In order for G_r to be *well-formed* for all edges $e \in E$, we must have,

$$w_r(e) \geq 0. \tag{3}$$

A retiming that satisfies (3) is called *legal.* It can be shown that for any path $u \overset{p}{\rightsquigarrow} v$, we have,

$$w_r(p) = w(p) + r(v) - r(u), \tag{4}$$

where $w(p) = \sum_{e \in p} w(e)$. From (4) it follows that when the path p is a cycle, retiming does not change its register count. The relation in (4) can be used to identify paths that can become critical after retiming. In order to preserve legality of retiming for all paths from node u to node v the maximum reduction in the register count of any path $u \overset{p}{\rightsquigarrow} v$ is given by the expression,

$$W(u,v) = \min\{w(p) : u \overset{p}{\rightsquigarrow} v\}. \tag{5}$$

Thus, the only paths $u \overset{p}{\rightsquigarrow} v$ that can become combinational (and possibly critical) 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 quantity,

$$D(u,v) = \max\{d(p) : u \overset{p}{\rightsquigarrow} v, w(p) = W(u,v)\} \tag{6}$$

gives the longest propagation delay from u to v whenever the retimed circuit includes a combinational path between the two vertices. Assuming zero setup/hold times for all flip-flops in the circuit, the clock period of any retimed circuit G_r is always some element in the $O(|V|^2)$ size set of $D(u,v)$'s.

For the retimed circuit G_r to have a critical path under the target clock period, every combinational path in G_r whose propagation delay exceeds the target clock period c must be interrupted by a register. Thus, for each vertex pair u, v in V with $D(u,v) > c$, the retimed register count $W_r(u,v)$ of each path $u \rightsquigarrow v$ that can become combinational must satisfy,

$$W_r(u,v) = W(u,v) + r(v) - r(u) \geq 1. \tag{7}$$

The $O(|V|^2)$ difference constraints specified by (3) and (7) can be computed in $O(|V||E| + |V|^2 lg|V|)$ steps by an all-pairs shortest-paths algorithm and can be solved in $O(|V|^3)$ steps by a Bellman-Ford algorithm for single-source shortest-paths.

4 Timing Constraints

As techniques to handle setup constraints have been discussed elsewhere, (e.g., [1]), we will not discuss them here. Prior to this work, hold constraints in retiming have only been discussed in [3] and we will derive here a new set of requirements for retimed circuits to satisfy hold constraints.

A retimed circuit is free of hold constraints *if and only if* there exists no register-to-register combinational path whose minimum effective propagation delay is shorter than the register hold time H.

We will now set out to develop two mathematical formulations, both of which will model min-area retiming with long as well as short-path constraints. The first technique we develop (formulation I) will provide a theoretically sound basis for solving min-area retiming with setup and hold constraints under *general* skew values. We will then present a practical argument (formulation II) that has a potential to reduce the problem complexity considerably under practical situations as it leads to a significant reduction in the number of constraints.

4.1 Formulation I

For the remainder of the paper we assume, without loss of generality, that any edge in the given circuit-graph G_r has at most one register on it. Any edge with more than one register on it can be decomposed in to multiple edges, with identical clock-skew, that connect dummy nodes with zero delay.

We begin by defining a new attribute, *critical short-path*, for every pair of edges e_i, e_j.

Definition 4.1 *For a pair of edges e_i, e_j and a path $u \xrightarrow{e_i} x \xrightarrow{p} y \xrightarrow{e_j} v$ the term $\Delta(p) + \sigma(e_i) - \sigma(e_j) - H \times (w(p) + w(e_i) + w(e_j) - 1)$ is the short-path metric of path $e_i \rightarrow p \rightarrow e_j$. Note that nodes x, y are included in the path p. And the path $u \xrightarrow{e_i} x \xrightarrow{p_{min}} y \xrightarrow{e_j} v$ with the smallest short-path metric $\Delta(p_{min}) + \sigma(e_i) - \sigma(e_j) - H \times (w(p_{min}) + w(e_i) + w(e_j) - 1)$ is the critical short-path containing the edges e_i and e_j.*

Theorem 4.1 *A circuit is free of hold time violations if and only if for every pair of edges e_i, e_j the critical short-path p containing these edges has a non-negative short-path metric, i.e., satisfies $\Delta(p) + \sigma(e_i) - \sigma(e_j) - H \times (w(p) + w(e_i) + w(e_j) - 1) \geq 0$.*

Proof: ($\Rightarrow$) Consider a path terminated at both ends by edge triggered registers. Let one register be on edge e_i and the other on e_j such that the path is directed from $e_i \xrightarrow{p} e_j$. Furthermore, let there be no other registers on the path from $e_i \xrightarrow{p} e_j$. Clearly by the assumption of ($\Rightarrow$), $\Delta(p) + \sigma(e_i) - \sigma(e_j) - H \times (2 - 1) \geq 0$. Hence, the path p satisfies hold time constraints. Due to the generality of the path p, all register to register combinational paths satisfy hold constraints.

($\Leftarrow$)We will prove this by contradiction. If possible, assume that no hold time constraints are violated in the circuit and yet for $e_i \xrightarrow{p} e_j$, $\Delta(p) + \sigma(e_i) - \sigma(e_j) - H \times (w(p) + w(e_i) + w(e_j) - 1) < 0$. Assume that the path under consideration has the following form $u_1 \xrightarrow{e_i} v_1 \overset{p_1}{\rightsquigarrow} u_2 \xrightarrow{e_{i_1}} v_2 \overset{p_2}{\rightsquigarrow} u_3 \xrightarrow{e_{i_2}} v_3 ... u_{n-1} \xrightarrow{e_{i_{n-2}}} v_{n-1} \overset{p_{n-1}}{\rightsquigarrow} u_n \xrightarrow{e_{i_{n-1}}} v_n \overset{p_n}{\rightsquigarrow} u_{n+1} \xrightarrow{e_j} v_{n+1}$.

There can be four different scenarios that we have to consider.

Case I: Edges $e_i, e_{i_1}, ..., e_{i_{n-1}}, e_j$ have a single register on them and $p_1, p_2, ..., p_n$ are purely combinational paths. Note that the case where edges have multiple registers are guaranteed to violate hold times [3], and hence violate the assumption in ($\Leftarrow$).

Case II: Edge e_i has no register on it, and the rest of the conditions are identical to Case I.

Case III: Edge e_j has no register on it, and the rest of the conditions are identical to Case I.

Case IV: Edges e_i, e_j have no registers on them, and the rest of the conditions are identical to Case I.

We will prove Case I, and the remaining cases can be proved similarly. In Case I, note that there are $n+1$ registers in the path under consideration (including the registers on edges e_i and e_j). Due to the assumption that all hold time constraints are satisfied we have,

$$\begin{aligned} \Delta(p_1) + \sigma(e_i) - \sigma(e_{i_1}) \geq H, (Path\ e_i \rightarrow e_{i_1}) \\ \Delta(p_k) + \sigma(e_{i_k}) - \sigma(e_{i_{k+1}}) \geq H, (Path\ e_{i_k} \rightarrow e_{i_{k+1}}) \\ 1 \leq k \leq n-2, \\ \Delta(p_n) + \sigma(e_{i_{n-1}}) - \sigma(e_j) \geq H, (Path\ e_{i_{n-1}} \rightarrow e_j). \end{aligned} \tag{8}$$

By adding all the above inequalities we obtain,

$$\begin{aligned} \sum_{k=1}^{n} \Delta(p_k) + \sigma(e_i) - \sigma(e_j) \geq (n)H, \\ \Rightarrow \Delta(p) + \sigma(e_i) - \sigma(e_j) \geq \\ (w(p) + w(e_i) + w(e_j) - 1) \times H, \end{aligned} \tag{9}$$

where we use the fact that $w(p) + w(e_i) + w(e_j) = n + 1$. The last inequality contradicts our initial assumption, and hence we arrive at the required proof.

Cases II-IV can be proved similarly by observing the fact that leading and trailing combinational paths will contribute a nonnegative amount to the expression $\Delta(p) + \sigma(e_i) - \sigma(e_j)$ due to clock skew monotonicity. Stripping away these leading and trailing combinational paths leaves us with Case I.

It follows that after retiming, a circuit will be free of hold time violations if and only if we can guarantee that for all edge pairs e_i and e_j the critical short-path $u \xrightarrow{e_i} x \xrightarrow{p} y \xrightarrow{e_j} v$ satisfies,

$$r(v) - r(u) \leq \frac{\Delta(p) + \sigma(e_i) - \sigma(e_j)}{H} - \frac{H \times (w(p) + w(e_i) + w(e_j) - 1)}{H}. \tag{10}$$

Due to the fact that $r(v)$, $r(u)$ are integral, the above constraint is equivalent to,

$$r(v) - r(u) \leq \left\lfloor \frac{\Delta(p) + \sigma(e_i) - \sigma(e_j)}{H} \right\rfloor - w(p) - w(e_i) - w(e_j) + 1. \tag{11}$$

Note that the above constraints are simple difference constraints very much in the manner of the retiming constraints in [1]. This means that the fast minimum cost network flow algorithms utilized for solving traditional min-area *long path only* retiming can still be used when hold constraints are included. So as long as we can identify a critical short-path for every pair of edges e_i and e_j we can perform min-area and min-period retiming in a manner akin to that in [1]. Thus the number of constraints we add to the original retiming framework is $O(|E|^2)$.

Next we will establish a *retiming invariance* property for the short-path metric for edge-to-edge cycles for edges with a register on them.

Property 4.1 *Let e be any edge with a register on it. Any cycle consisting of edge e has a retiming invariant short-path metric.*

Proof: Consider an edge e with a single register on it. Let e be part of some cycle $C = u \xrightarrow{e} x \overset{p}{\rightsquigarrow} u \xrightarrow{e} x$. The short-path metric from e-to-e along C is $\Delta(p)+\sigma(e)-\sigma(e)-H\times(W(p)+2w(e)-1)$. Now since the path p includes x and u therefore $\Delta(p) = \Delta(C)$. Also, since $w(e) = 1$ by assumption, $W(p) + 2w(e) - 1 = W(p) + w(e) = W(C)$. Hence, the short-path metric from e-to-e along cycle C is $\Delta(C) - H \times W(C)$. Since both $\Delta(C)$ and $W(C)$ are retiming invariant, the present short-path metric is also retiming invariant. Clearly for the hold time constraints to be satisfiable all edge-to-edge cycles should have a non-negative short-path metric and hence $\Delta(C) - H \times W(C) \geq 0$ for all cycles C in G_r.

Next we will show that the problem of finding a critical short-path for any pair of edges can be modeled as a shortest path problem in a graph but with the possibility of negative weight edges in the graph.

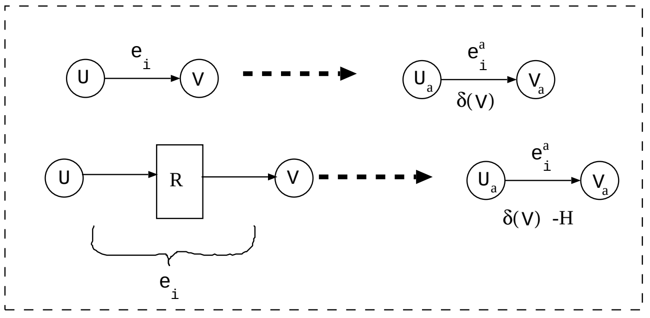

Figure 2: Illustration showing how to obtain the auxiliary graph G_a from the circuit-graph G_r.

From the original circuit graph G_r, we generate an auxiliary graph $G_a = (V_a, E_a, W'(E_a))$ such that every node $v \in V(G_r)$ has a corresponding node in $v_a \in V_a(G_a)$. Also every edge $u \overset{e_i}{\rightarrow} v \in E(G_r)$ has a corresponding edge $u_a \overset{e_i^a}{\rightarrow} v_a \in E_a(G_a)$. In addition this edge e_i^a has a weight $w(e_i^a) \in W'(E_a)$ which has a value $\delta(v)$ if e_i is register-free and a value $\delta(v) - H$ if e_i has a register on it. We illustrate the construction of G_a in Fig. 2. Every acyclic, u to v, path $u \overset{e_i}{\rightarrow} x \overset{p}{\rightarrow} y \overset{e_j}{\rightarrow} v$ therefore has a corresponding path in G_a with total weight $\Delta(p) + \delta(v) - (w(p) + w(e_i) + w(e_j)) \times H$. Note that the *short-path metric* corresponding to the acyclic e_i to e_j path p differs from the previously computed path-weight for the corresponding u_a to v_a auxiliary path in G_a by an additive factor $\sigma(e_i) - \sigma(e_j) - \delta(v) + H$. Note also that this additive factor is the same for all e_i to e_j paths. Therefore an e_i^a to e_j^a path with minimum weight in G_a identifies an e_i to e_j path with minimum *short-path metric* and hence identifies the *critical short-path* from e_i to e_j in G.

Property 4.2 *If the hold time constraints are satisfiable then the auxiliary graph G_a has no negative weight cycles.*

Proof: Now consider a cycle C_a in G_a; clearly there must be a corresponding cycle C in G_r. Also, this cycle C in G_r must contain at least one register as combinational cycles are illegal in G_r. Let e_i be an edge in this cycle C containing a register. The cycle C is therefore of the form $u \overset{e_i}{\rightarrow} x \overset{p}{\rightsquigarrow} u$. Now the short-path metric for this cycle computed from e_i to e_i is $= \Delta(C) - W(C) \times H$ as $w(e_i) = 1$, from Property 4.1. On the other hand the path-weight for C_a in G_a is also given by $\Delta(C) - W(C) \times H$. Hence, a negative weight cycle in G_a will imply a register-to-register cycle with a negative *short-path metric* in G_r. Since a register-to-register cycle with negative short-path metric implies unsatisfiability of the hold time constraints from Property 4.1, so does a negative weight cycle in G_a. Any such cycle can be easily detected by a Bellman-Ford algorithm with $O(|V||E|)$ complexity.

Therefore we can use an all pair shortest path algorithm on the auxiliary graph to compute short-path metrics for all pairs of edges in the circuit graph.

Next, we will setup the min-area retiming problem with setup and hold constraints. Fanout nodes are handled by using mirror vertices as detailed in [1]. We call the new algorithm MARSH ($\underline{\text{M}}$in-$\underline{\text{A}}$rea $\underline{\text{R}}$etiming with $\underline{\text{S}}$etup and $\underline{\text{H}}$old constraints).

Algorithm MARSH

0)Objective:

Minimize $\sum_{u \in V}$ *(Indegree(u) - Outdegree(u))r(u)*

1)Legality constraints: For every edge $u \overset{e_i}{\rightarrow} v$,

$$w_r(e_i) = r(v) - r(u) + w(e_i) \geq 0. \tag{12}$$

2)Long-path Constraints: For every edge pair $x \overset{e_i}{\rightarrow} u \overset{p}{\rightarrow} v \overset{e_j}{\rightarrow} y$ *such that* $D(u,v) + \sigma(e_i) - \sigma(e_j) \geq c - S$ *we have,*

$$W(u,v) + r(v) - r(u) \geq 1. \tag{13}$$

3)Short-path Constraints: For the critical short-path p *of every edge pair* $u \overset{e_i}{\rightarrow} x \overset{p}{\rightarrow} y \overset{e_j}{\rightarrow} v$ *we have,*

$$r(v) - r(u) \leq \left\lfloor \frac{\Delta(p) + \sigma(e_i) - \sigma(e_j)}{H} \right\rfloor - w(p) - w(e_i) - w(e_j) + 1, \tag{14}$$

$$r(u) \in \mathbf{Z}, \ u, v \in V,$$

due to the fact that all the above constraints are difference constraints with at most two retiming variables corresponding to two vertices, many of the constraints will be redundant and by standard elimination techniques the number of constraints can be brought down to $O(|V|^2)$ from $O(|E|^2)$.

We can use the minimum cost circulation techniques presented in [11] and MARSH can therefore be executed in $O(|V||V|^2 log|V| log(|V|C))$ where $C \approx$ Number of registers in the graph. Additionally, we can drastically reduce the number of min-period constraints by using the speedup methods in [6, 7, 9, 10].

4.2 Formulation II

Recall that the basic requirement to ensure satisfaction of short-path constraints is that all register-to-register combinational paths should have a minimum effective propagation delay that exceeds H. Also, clock skew monotonicity ensures that if P_s is a sub-path of a path P then the effective minimum delay of P_s is no larger than the effective minimum delay of P. The last observation leads to the following:

Theorem 4.2 *A retimed circuit is free of hold time violations if and only if any path,* $p = u \overset{e_i}{\rightarrow} x \overset{q}{\rightsquigarrow} y \overset{e_j}{\rightarrow} v$ *satisfies: if* $\Delta(p) + \sigma(e_i) - \sigma(e_j) \leq H$ *then* $w_r(p) \leq 1$.

Proof: ($\Rightarrow$) Assume, if possible that hold time requirements are satisfied and yet a path $p = u \overset{e_i}{\rightarrow} x \overset{q}{\rightsquigarrow} y \overset{e_j}{\rightarrow} v$ has $\Delta(p) + \sigma(e_i) - \sigma(e_j) \leq H$ and $w_r(p) \geq 2$. Clearly, due to clock skew monotonicity any register-to-register sub-path of p will have a hold time violation.
($\Leftarrow$) Clearly if any register-to-register path $p = u \overset{e_i}{\rightarrow} x \overset{q}{\rightsquigarrow} y \overset{e_j}{\rightarrow} v$ has a hold time violation then $\Delta(p) + \sigma(e_i) - \sigma(e_j) \leq H$ while $w_r(p) \geq 2$ and hence $\Leftarrow$ is proved.

The potential problem with the above formulation is that for general values of hold time H and clock skew $\sigma(e_i)$ the number of constraints may grow exponentially. However, as it turns out, for the present retiming model to be applicable, the clock skew value for any edge e must satisfy $\sigma(e) \leq \epsilon$, i.e., we have a near zero-skew clock network. Therefore ϵ can be safely assumed to be restricted to a small number of (say 2-3) gate delays; in addition the value of H is also limited to about 2 gate delays. This implies that the hold time constraints only need to be enumerated for all paths over a small number of logic levels. The formulation of our work and [3] are invalid for larger *deliberate* values of clock skew [12] since additional constraints ensuring logic wave separation [13] need to be included and are beyond the scope of this paper.

The above observations guarantee that the number of constraints added for ensuring satisfaction of hold time requirements in practical circuits is significantly less than $O(|V|^2)$. This complexity reduction can probably be combined with the complexity reduction in [6,7] to possibly develop a faster technique for retiming with both short-path and long path constraints. We will now summarize the new formulation that we call MARSH-II as follows:

Algorithm MARSH-II

0)Objective:

Minimize $\sum_{u \in V}$ *(Indegree(u) - Outdegree(u))r(u)*

1)Legality constraints: For every edge $u \overset{e_i}{\rightarrow} v$,

$$w_r(e_i) = r(v) - r(u) + w(e_i) \geq 0. \tag{15}$$

2)Long-path Constraints: For every edge pair $x \overset{e_i}{\rightarrow} u \overset{p}{\rightarrow} v \overset{e_j}{\rightarrow} y$ *such that* $D(u,v) + \sigma(e_i) - \sigma(e_j) \geq c - S$ *we have,*

$$W(u,v) + r(v) - r(u) \geq 1. \tag{16}$$

3)Short-path Constraints: For all paths $p = u \overset{e_i}{\rightarrow} x \overset{q}{\rightarrow} y \overset{e_j}{\rightarrow} v$ *satisfying* $\Delta(p) + \sigma(e_i) - \sigma(e_j) \leq H$ *we have,*

$$r(v) - r(u) + w(p) \leq 1. \tag{17}$$

$$r(u) \in \mathbf{Z},\ u, v \in V$$

5 The Unification of Retiming and Minimum Delay-Padding

The work in [8] provide a technique that may be used to address short-path constraints as a post-processing step after retiming has been performed for addressing long path violations. This technique will always guarantee the fastest possible circuit implementation that can be achieved

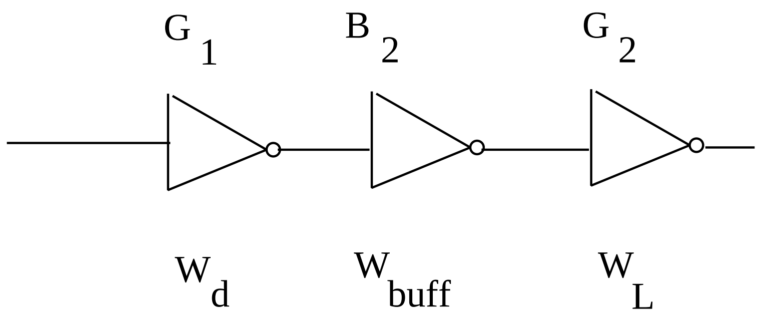

Figure 3: A gate of width W_d driving a buffer with width W_{buff} that drives a load of width W_L.

with retiming. What cannot be guaranteed, however, is that the resulting circuit will have a minimum area. Using retiming alone to solve the problem of simultaneously addressing short-path and long path constraints, on the other hand, can guarantee neither the fastest circuit implementation nor can it guarantee the minimum area solution for a given circuit speed. The trick lies in being able to combine the two approaches in to a single unified framework. Before proceeding further we need to examine the problem of delay padding in detail. Specifically, we will examine the mechanism of trying to delay a signal at the output of a gate driving a fanout load.

5.1 Delay-Padding for a Gate with Fanout

In Fig. 3 a gate G_1 with width W_d is shown to drive a delay-buffer B_2 with width W_{buf}, that in turn drives a fanout load gate G_3 with width W_L. Using the commonly used Elmore delay model [14], we will consider two cases, if the buffer B_2 is replaced by a short circuit then the delay of the signal of the output net of the gate G_1 is given by

$$delay(gate) = a \times \frac{W_L}{W_d}, \tag{18}$$

where a is a constant. In the presence of the delay-buffer, the delay can be computed as

$$delay(gate) = a \times (\frac{W_{buf}}{W_d} + \frac{W_L}{W_{buf}}). \tag{19}$$

Therefore, in order for the buffer to delay the signal on the net we require that,

$$\frac{W_{buf}}{W_d} + \frac{W_L}{W_{buf}} \geq \frac{W_L}{W_d}, \tag{20}$$

which implies that either $W_L \leq 4W_d$ or W_{buf} must satisfy the following,

$$\begin{aligned} W_{buf} &\leq \frac{W_L}{2} - \frac{\sqrt{W_L(W_L - 4W_d)}}{2}, or, \\ W_{buf} &\geq \frac{W_L}{2} + \frac{\sqrt{W_L(W_L - 4W_d)}}{2}. \end{aligned} \tag{21}$$

In case, $W_L \geq 4W_d$, since we are looking for a minimum area solution we would be more interested in $W_{buf} \leq \frac{W_L}{2} - \frac{\sqrt{W_L(W_L-4W_d)}}{2}$. We restrict our discussion to the delay-padding problem where any given wire uses delay buffers of unit delay. If a delay of multiple units is required on the wire, then a chain of buffers is used to achieve this objective. We ignore fanout loading effects for the moment and assume that irrespective of load a buffer of unit delay has unit area. This assumption is made because the vast body of literature on retiming, including our present paper, ignores variable loads on nets due to the presence/absence of flip-flops. So keeping in tune with these assumptions it is reasonable to assume that unit-delay buffers have identical area irrespective of the load they drive. However, we have a solution that can take care of fanout loading effects also, in such cases the previous approximation is enhanced in the following manner. Due to the different loading effects at the output of various wires in the circuit, the area of a unit delay buffer at the output of each wire could be different. The area of the buffer required to produce unit delay on the wire from gate, u to gate v can be denoted by $A_b(u \to v)$. For any fixed configuration of flip-flops in the circuit $A_b(u \to v)$ can be determined for various wires in the circuit and the min-delay padding problem as formulated in [8] can be solved. Once padding has been accomplished to satisfy all the short-path requirements we can further formulate a buffer relocation strategy to obtain a minimum area padding solution, by noting the following,

1. Delaying different wires by the same amount will require buffer chains of different areas

that can be quantified by the cost parameters $A_b(u \to v)$.

2. Relocating the buffers can be accomplished using a technique very similar to [15].

5.2 Buffer-Relocation Constraints for Minimum Delay Padding

Assume that the number of unit-delay buffers on the wire e_i is given by $w_b(e_i)$. The *buffer-relocation* transformation associates an integer valued label $b(u)$ with every gate/flip-flop in the circuit graph and performs the following optimization problem, where FF denotes the set of flip-flops.

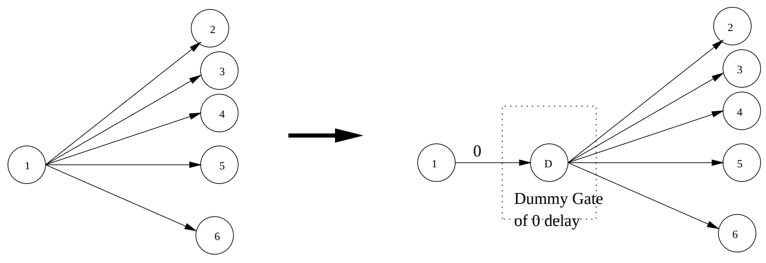

Figure 4: A circuit-graph transformation to handle fanout nodes.

Algorithm Buffer-Relocation

0)Transform the circuit as shown in Fig. 4.

1)Objective:

Minimize $\sum_{u \in V, v \in Fanout(u)} w_b(u \to v)(b(v) - b(u))$

2)Legality constraints: For every edge $u \overset{e_i}{\to} v$,

$$b(v) - b(u) + w_b(e_i) \geq 0. \tag{22}$$

3)Short-path, Long-path Constraints: For all all primary inputs, PI, all primary outputs PO and all flip-flops FF, we must have,

$$b(PI) = b(PO) = b(FF) = 0. \tag{23}$$

The requirement that $b(FF) = 0$ is added to make sure that buffers are not relocated across flip-flops as this can affect the long path/short-path constraints of the circuit. All the properties of retiming [1] also hold for the buffer-relocation problem, [15].

5.3 Putting It All Together

As pointed out earlier, using using MARSHI or MARSHII alone cannot always guarantee the fastest possible circuit implementation. This is due to the fact that relocating flip-flops in such a manner that hold time requirements are also satisfied may make it difficult to satisfy long path constraints for the fastest possible clock period. To guarantee fastest possible circuit implementations, better heuristics must be devised. One strategy tries to selectively introduce short-path constraints in the input circuit with a formulation like MARSHII. The key idea used here is to use flip-flop relocation for fixing short-path violations in those paths that allow relatively higher freedom of flip-flop movement and use min-delay padding elsewhere. The notion of how easy it is to relocate a flip-flop about a region can be obtained by examining the bounds on the retiming variable, [6, 7], associated with circuit-graph nodes in that region. A search technique can be employed by iteratively running a MARSHII like formulation, while retaining the short-path constraints only on paths originating/terminating on nodes with varying "slack" (upper bound of retiming variable - lower bound of retiming variable). In this way a near-optimal search based technique can be formulated.

We now describe the approach in detail. The approach we describe here uses a "modified" branch and bound strategy by using a binary search, [16], based approach. This approach guarantees at least as good a solution as the traditional approach because the traditional approach is used as a starting point for the branching strategy. The advantage of these approaches is that they can reduce the number of buffers inserted for correcting short-path violations while marginally increasing the number of flip-flops.

One such approach will be to first identify nodes in the circuit-graph that are hard to retime. This can be done in a manner similar to [6, 7] where mobility of flip-flops is used to find bounds on the retiming variables of various nodes of the circuit-graph. It is therefore possible to identify certain nodes that are hard to retime, i.e., the retiming variable associated with them can not change by a large amount. Typically in [6, 7] a class of such nodes is identified and is defined as "fixed" nodes. These nodes will have no flip-flops moving either way across them. Intuitively, the neighborhood of any such node is likely to remain relatively unchanged due to retiming. Therefore, if we include short-path constraints on such a node then the resulting problem formulation might turn out to be infeasible. It is possible to identify nodes in a decreasing order of "hardness" to retime by sorting them with respect to their slack (upper bound of retiming variable - lower bound of retiming variable). An iterative approach can then be devised which identifies hard-to-retime nodes by classifying them according their slack. Once such a class of hard-to-retime nodes is identified, we can formulate a relaxed version of MARSHII which relaxes the short-path constraints imposed on edges emanating/terminating from/on such hard-to-retime nodes. We now describe MARSHIII a heuristic in this category that first explores the nodes in the circuit to be retimed and identifies nodes that are hard-to-retime. As mentioned before, this is accomplished by calculating bounds on the retiming variables corresponding to nodes, as in [6, 7]. Any node that has a small difference between the upper and lower bound for the retiming variable associated with it is especially hard to retime and the short-path constraints in MARSHII involving such a node are relaxed. This will allow a greater freedom for the retiming problem than MARSHI or MARSHII alone, so that a possibly faster circuit can be realized.

Now we will describe the steps required to implement MARSHIII.

Algorithm MARSHIII

1)Define FREEDOM of a node = (upper bound of retiming variable - lower bound of retiming variable). Let maximum freedom of a node in the circuit graph be MAX and let the minimum freedom of a node in the circuit-graph be MIN.

2)Perform long-path only retiming followed by min-delay padding, buffer relocation and a MARSHII like approach which introduces short-path constraints only on those paths that either originate or terminate on a node with FREEDOM $\geq$ 1 followed by min-delay padding, buffer-relocation.

Note that the traditional long-path only retiming is mathematically equivalent to introducing short-path constraints for paths either originating or terminating on nodes with FREEDOM $\geq$ MAXIMUM + 1, i.e., there are no such paths. Define ITER(LIM) as the process of performing a MARSHII like approach which introduces short-path constraints only on those paths that either originate or terminate on a node with FREEDOM $\geq$ LIM by min-delay padding, buffer-relocation.

3)Set $MAX_F = FREEDOM + 1$, $MIN_F = 0$. If area after ITER(MAX_F) $>$ area after ITER(MIN_F), then set $MIN_F = \lfloor \frac{MAX_F + MIN_F}{2} \rfloor$. Otherwise, set $MAX_F = \lceil \frac{MAX_F + MIN_F}{2} \rceil$. If either ITER($MAX_F$) or ITER($MIN_F$) is infeasible then return an area of ∞ for that case.

4)Iterate above process till $MIN_F \geq MAX_F$, and put out the circuit which has a lower area among the two.

Note that MARSHIII involves at most $\lceil log_2(FREEDOM + 1) \rceil$ iterations.

6 Simulation Results

Simulation results were obtained for circuits from the ISCAS89 benchmark suite. Additional large circuits obtained by splicing together multiple instances of ISCAS89 benchmark circuits were also simulated. A unit delay model was used for the gates, a setup time of 0 unit and a hold time of 1 unit was assumed for all the flip-flops in the circuit. The fast retiming tool MINARET, [6, 7], was modified to incorporate hold time constraints. Simulation results were obtained for traditional long path only retiming followed by min-delay padding, and for MARSHIII. Table 1 shows the results of the simulation, and displays the number of flip-flops and buffers in each case. The platform used for simulation was an Ultra-Sparc10 Sun workstation. It is observed that a significant reduction in the number of delay buffers is often possible with a consequent increase in the number of flip-flops. As an example, the number of buffers in myex5, in Table. 1, is decreased by 92 at the expense of 10 flip-flops. As these buffers are non-inverting, assuming conservatively that a flip-flop is 3 times as large as a delay buffer, this represents a savings equivalent to 62 delay buffers. We expect to achieve higher gains in real circuits that have a larger number of flip-flops and a larger number of hold constraints to correct. As can be seen from Table. 1, in some cases it is possible to reduce the number of buffers required for min-delay padding with a minimal increase in the number of flip-flops. All circuits have been retimed for the fastest possible clock period possible with retiming. The last column reports the effective savings in buffer area for MARSHIII over MINARET here it is assumed that a flip-flop is 3 times as large as a non-inverting delay buffer.

7 Conclusions

We have presented the first polynomial-time algorithm ever reported for min-area retiming with setup and hold constraints (MARSH). Our algorithm runs in $O(|V|^3 log|V|log(|V|C))$ steps, as

Table 1: Comparison of MINARET (I) and MARSHIII (II) for retiming of benchmark circuits . The last column MAX_{Free} refers to the FREEDOM of the node with the maximum FREEDOM in the circuit.

Circuit	# Hold Violations	#FFs I	#FFs II	#Buffers I	#Buffers II	Eff. Savings I	CPU (TIME) II	CPU (TIME)	MAX_{Free}
s3271	3	168	168	3	3	0	0.19s	0.6s	0
s3384	25	167	167	25	17	6	1.22s	3.9s	8
s38417	146	1395	1397	147	93	9	81.26s	361s	48
s4863	4	138	139	4	0	3	2.76s	5.6s	1
s5378	14	173	173	14	11	8	0.78s	2.34s	3
s13207	47	446	447	47	24	14	5.91s	24s	20
s15850	12	525	525	12	3	6	25.81s	78s	9
s38584	156	1427	1427	156	154	11	39.68s	160s	2
myex2	177	2022	2024	177	155	11	100.24s	401s	16
myex1	146	2293	2296	146	114	7	95.89s	288s	23
myex3	242	3279	3279	242	233	12	278.18s	1210s	9
myex4	289	2803	2816	289	233	13	239.95s	1032s	17
myex5	334	3378	3388	334	242	13	444.35s	1790s	62

described in section 4. We have also provided a practical argument to show that the problem complexity can be be reduced considerably by following an alternative formulation. Also, presented were an extension to the min-delay padding algorithm with a concept called buffer-relocation. Finally, a heuristic was presented which uses iterative applications of relaxed versions of MARSHII to address the retiming problem with setup and hold constraints. The heuristic MARSHIII guarantees at least as good a performance as the traditional approach of performing setup only retiming followed by min-delay padding for addressing hold constraints. The execution time requirements for MARSHIII involve at most $\lceil log_2(FREEDOM + 1)\rceil$ iterations of a process similar to MINARET, which is known to be extremely fast.

References

[1] C. E. Leiserson and J. B. Saxe, "Optimizing Synchronous Systems," *Journal of VLSI and Computer Systems*, vol. 1, no. 1, pp. 11–67, 1983.

[2] C. Ebeling and B. Lockyear, "The Practical Application of Retiming to the Design of High-Performance Systems," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 288–295, November 1993.

[3] M. C. Papaefthymiou, "Asymptotically Efficient Retiming under Setup and Hold Constraints," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 288–295, November 1998.

[4] S. Dey and S. Chakradhar, "Retiming Sequential Circuits to Enchance Testability," *Proceedings of the 12th VLSI Test Symposium*, pp. 28–33, April 1994.

[5] S. Malik, E. Sentovich, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Networks with Combinational Techniques," *Proceedings of the Hawaii International Conference on System Sciences*, June 1990.

[6] N. Maheshwari and S. S. Sapatnekar, "Efficient Retiming of Large Circuits," *IEEE Transactions on VLSI Systems*, pp. 74–83, March 1998.

[7] N. Maheshwari and S. S. Sapatnekar, "Optimizing Large Multiphase Level-Clocked Circuits," *IEEE Transactions on Computer-Aided Design*, pp. 1249–1264, September 1999.

[8] N. Shenoy *et al.*, "Minimum Padding to Satisfy Short Path Constraints," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 156–161, November 1993.

[9] N. Shenoy and R. Rudell, "Efficient implementation of retiming," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 226–233, November 1994.

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

[11] A. V. Goldberg, E. Tardos, and R. E. Tarjan, "Network Flow Algorithms," *Technical Report, Department of Computer Science, Stanford University*, 1989.

[12] J. P. Fishburn, "Clock Skew Optimization," *IEEE Transactions on Computers*, vol. 39, pp. 945–951, July 1990.

[13] D. A. Joy and M. J. Ciesielski, "Clock Period Minimization with Wave Pipelining," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 461–472, April 1993.

[14] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective 2nd Edition.* Addison-Wesley, 1993.

[15] V. Sundararajan and K. K. Parhi, "Low Power Gate Resizing Using Buffer-Redistribution," in *Proceedings of the Twentieth Anniversary Conference on Advanced Research in VLSI*, pp. 170–184, March 1999.

[16] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms.* Cambridge, Mass. : MIT Press; New York : McGraw-Hill, 1990.