Efficient Retiming of Large Circuits

Naresh Maheshwari, *Student Member, IEEE,* and Sachin Sapatnekar, *Member, IEEE*

Abstract—
Retiming, introduced by Leiserson and Saxe, is a powerful transformation of circuits that preserves functionality and improves performance. The ASTRA algorithm proposed an alternative view of retiming using the equivalence between retiming and clock skew optimization, and also presented a fast algorithm for minimum period (minperiod) retiming. Since minperiod retiming may significantly increase the number of flip-flops in the circuit, minimum area (minarea) retiming is an important problem. Minarea retiming is a much harder problem than minperiod retiming, and previous techniques where not capable of handling large circuits in a reasonable time. This work defines the relationship between the Leiserson-Saxe and the ASTRA approaches and utilizes it for efficient minarea retiming of large circuits. The new algorithm, Minaret, uses the same basis as the Leiserson-Saxe approach. The underlying philosophy of the ASTRA approach is incorporated to reduce the number of variables and constraints generated in the problem. This allows minarea retiming of circuits with over 56,000 gates in under fifteen minutes.

I. Introduction

Retiming is a procedure that involves the relocation of flip-flops (FF's) across logic gates to allow the circuit to be operated under a faster clock. The technique was first proposed by Leiserson and Saxe [1], [2], where the algorithmic basis of retiming circuits with edge-triggered FF's was described without specifically focusing on implementational aspects. Retiming to achieve the minimum clock period is termed minperiod retiming, while retiming to minimize the number of memory elements for a given target clock period is called minarea retiming. Several papers have been published since then, primarily dealing with algorithmic issues and extending the Leiserson-Saxe method to handle variations of the original Leiserson-Saxe problem (for example, retiming level-clocked circuits [3], improving the delay model [4], and retiming with equivalent initial states [5] etc.).

It was only recently that practical algorithms for handling large VLSI circuits were introduced [6] [7] [8]. Of these only [6] targets minarea retiming and was able to retime a 7882 gate circuit in 38 hours. In this work, our target is to be able to perform constrained minarea retiming on circuits with tens of thousands of gates in under an hour. The ASTRA algorithm [7] displayed a different view of retiming by using an equivalence between clock skew and retiming. This algorithm was applied to the minperiod retiming problem but not to minarea retiming.

For digital design the only interesting objective function is that of constrained minimum area retiming [6]. However, the high computational expense of this optimization has limited its use. In this work, we approach the problem of constrained minarea retiming for circuits with edge-triggered FF's through an amalgamation of the Leiserson-Saxe approach and the ASTRA approach. By utilizing the merits of both approaches we develop an efficient algorithm for constrained minarea retiming which is also capable of handling very large circuits. The basic idea of the approach is to use the ASTRA approach to find tight bounds on the retiming variables. These bounds help us reduce both the number of variables and the number of constraints in the problem without any loss in accuracy. By spending a small amount of additional CPU time on the ASTRA runs, this method leads to significant reductions in the total execution time of the minarea retiming problem. The reduction in the problem size also reduces the memory requirements, thus enabling retiming of large circuits. Experimental results are shown on large circuits (up to 56,000 gates), and it is seen that they can be solved in very reasonable amount of time (under 15 minutes). As in most of the references on retiming listed above, this paper assumes the circuit to be composed of gates with fixed delays.

The paper is organized as follows. In Section II, the outlines of the ASTRA and the Leiserson-Saxe approaches are presented. Next, in Section III, we show the relationship between these two, and utilize it to efficiently solve the minarea retiming problem. Section IV presents retiming under a more accurate area model. Section V describes our minarea retiming algorithm. Experimental results are presented in Section VI, followed by concluding remarks in Section VII.

II. Background

For completeness, we first describe the ASTRA method and the Leiserson-Saxe approach for minimum area retiming. These approaches will be combined to form "Minaret," a practical minimum area retiming algorithm.

A. *The relationship between clock skew and retiming*

In a sequential VLSI circuit, due to differences in interconnect delays on the clock distribution network, clock signals do not arrive at all of the FF's at the same time. Thus, there is a skew between the clock arrival times at different FF's. In a single-phase clocked circuit, in the case where there is no clock skew, the designer must ensure that each input-output path of a combinational circuit block has a delay that is less than the clock period. In the presence of skew, however, the relation grows more complex as one must compensate for this effect in ensuring that the combinational blocks meet the timing requirements.

The basis of the ASTRA approach is the equivalence

This work was supported in part by National Science Foundation award MIP-9502556, a Lucent Technologies DAC Graduate Scholarship and a Iowa State University Computation Center Grant.

Naresh Maheshwari is with the Department of Electrical & Computer Engineering, Iowa State University. E-mail naresh@iastate.edu
Sachin Sapatnekar is with the Department of Electrical & Computer Engineering, University of Minnesota. E-mail sachin@ee.umn.edu

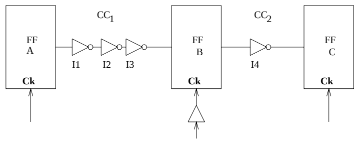


Fig. 1. The advantages of nonzero clock skew.

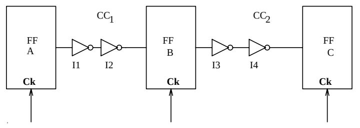


Fig. 2. Retiming for clock period optimization.

between clock skew and retiming, as illustrated by the following example. Let us first consider the use of intentional clock skews for improving the circuit performance. In Figure 1, assume the delays of the inverters to be 1.0 unit each. The combinational blocks CC_1 and CC_2 have delays of 3.0 and 1.0 units, respectively, and therefore, the fastest allowable clock has a period of 3.0 units. However, if a skew of +1.0 unit is applied to the clock line to FF B, the circuit can run with a clock period of 2.0 units. This approach was formalized in the work by Fishburn [9], where the clock skew optimization problem was formulated as a linear program (LP) that may be solved to find the optimal clock period.

However, it is easy to see that for the given circuit, the period can also be minimized to 2.0 units by *retiming*, i.e., by relocating the FF B to the left across the inverter I3. This results in the combinational circuit blocks CC_1 and CC_2 having delays of 2.0 units each as seen in Figure 2.

This leads us to conclude that in each case, one unit of time is borrowed by CC_1 from CC_2; the manner in which cycle-borrowing occurs may either be by the vehicle of clock skew or via retiming.

A.1 Outline of the ASTRA algorithm

The details of the ASTRA algorithm are provided in [7]; a brief description is presented here for completeness. The relationship between skew and retiming motivates the following two-phase solution to the retiming problem:

Phase A : The clock skew optimization problem is solved to find the optimal value of the skew at each FF, with the objective of minimizing the clock period, or to satisfy a given (feasible) clock period. This involves the (possibly repeated) application of the Bellman-Ford algorithm [10] on a constraint graph.

Phase B : The skew solution is translated to retiming and some FF's are relocated across gates in an attempt to set the values of all skews to be as close to zero as possible. We attempt to move each positive skew FF opposite to the direction of signal propagation, and each negative skew FF in the direction of signal propagation to reduce the magnitude of its skew. A formal rationalization is provided in [7], but the example in Figure 1 should suffice to explain the intuition.

After Phase B, any skews that could not be set exactly to zero are forced to zero. This could cause the clock period to increase from Phase A; however, it is shown that this increase will be no greater than the maximum gate delay. Note, however, that this is not necessarily suboptimal since the minimum clock period using skews may not be achievable using retiming, since retiming allows cycle-borrowing only in discrete amounts (corresponding to gate delays), while skew is a continuous cycle-borrowing optimization [9].

B. *The Leiserson-Saxe algorithm for minimum area retiming*

B.1 Notation

A sequential circuit can be represented by a directed graph $G(V, E)$, where each vertex v corresponds to a combinational gate, and a directed edge e_{uv} represents a connection from the output of gate u to the input of gate v, through zero or more registers. Each edge has a weight $w(e_{uv})$, which is the number of registers between the output of gate u and the input of gate v. Each vertex has a constant delay $d(v)$. A special vertex, the host vertex, is introduced in the graph, with edges from the host vertex to all primary inputs of the circuit, and edges from all primary outputs to the host vertex.

A retiming is a labeling of the vertices $r : V \rightarrow Z$, where Z is the set of integers. The weight of an edge e_{uv} after retiming, denoted by $w_r(e_{uv})$ is given by

$$w_r(e_{uv}) = r(v) + w(e_{uv}) - r(u) \quad (1)$$

The retiming label $r(v)$ for a vertex v represents the number of registers moved from its output towards its inputs. One may define the weight of any path p originating at vertex u and terminating at vertex v (represented as $u \leadsto v$), $w(p)$, as the sum of the weights on the edges on p, and its delay $d(p)$ as the sum of the delays of the vertices on p. A path with $w(p) = 0$ corresponds to a purely combinational path with no registers on it; therefore, the clock period can be calculated as $P = \max_{\forall\, p | w(p)=0} \{d(p)\}$.

Another important concept used in the Leiserson-Saxe approach is that of the W and D matrices that are defined as follows:

$$W(u, v) = \min_{\forall\, p: u \leadsto v} \{w(p)\} \quad (2)$$

$$D(u, v) = \max_{\forall\, p: u \leadsto v \text{ and } w(p) = W(u,v)} \{d(p)\} \quad (3)$$

The matrices are defined for all pairs of vertices (u, v) such that there exists a path $p : u \leadsto v$ that does not include the host vertex. $W(u, v)$ denotes the minimum latency, in clock cycles, for the data flowing from u to v and $D(u, v)$ gives the maximum delay from u to v for the minimum latency.

B.2 The minarea retiming algorithm

Consider the circuits in Figure 3(a) and 3(b). Assuming unit gate delays, the minimum achievable value of the clock period is 4.0 units; each of these two circuits achieves this period, but the latter utilizes more FF's. Thus it is possible

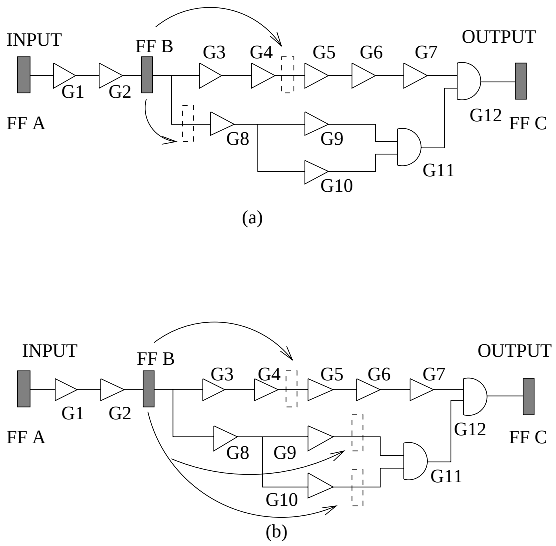


Fig. 3. Possible FF locations after retiming.

to reduce the number of FF's in a circuit by relocating them. The minarea retiming problem for a target period P can be formulated as the following LP:

$$\begin{array}{lll} \min & \sum_{v \in V} \left[(|FI(v)| - |FO(v)|) \cdot r(v) \right] & (4) \\ \text{subject to} & r(u) - r(v) \leq w(e_{uv}) \quad \forall e_{uv} \in E & \\ & r(u) - r(v) \leq W(u,v) - 1 \quad \forall D(u,v) > P & \end{array}$$

where $FI(v)$ and $FO(v)$ represent the fanin and fanout set of the gate v respectively.

The significance of the objective function and the constraints is as follows (the reader is referred to [2] for details).

- The objective function represents the number of registers added to the retimed circuit in relation to the original circuit.
- The first constraint ensures that the weight e_{uv} of each edge (i.e., the number of registers between the output of gate u and the input of gate v) after retiming is non-negative. We will refer to these constraints as *circuit constraints.*
- The second constraint ensures that after retiming, each path whose delay is larger than the clock period has at least one register on it. These constraints, being dependent on the clock period, are often referred to as *period constraints.*

It is pointed out in [2] that the dual of this problem is an instance of a minimum cost network flow problem, and hence can be solved efficiently by solving the dual.

III. Reducing the problem size

A. Background

In practical circuits, it is found that the number of period constraints is phenomenally large. For a circuit with n gates the number of period constraint is $O(n^2)$. However, it is also true that a large fraction of these constraints are redundant as they are implied by some of the other constraints. Any algorithm with pretensions to practicality must use techniques for pruning these redundant constraints. Note that the exactness of the solution is not sacrificed in doing so, since none of the essential constraints are removed. Our approach is to find tight bounds on the variable values, and to use these bounds to avoid generating the redundant constraints. By appropriate application of these bounds, we expect not only to prune the constraint set but also to reduce the number of variables. In this way, we simplify the problem and enable the LP to be solved more efficiently. We are also able to generate this set of reduced constraints efficiently.

B. The concept of restricted mobility

A modification of the procedure used in ASTRA can be used to identify how far FF's may possibly be moved. For the circuit in Figure 3, to achieve the minimum clock period of 4.0 units, one must move one copy of FF B to the output of gate G4. The possible locations for FF's along the other path to FF C are at the input to gate G8, or at the output of gate G8, or the inputs of gates (G9,G10) or the outputs of gates (G9,G10); no other locations are permissible

Therefore, it can be seen that the FF's cannot be sent to *just any* location in the circuit; rather, there is a restricted range of locations into which each FF may be moved, and the mobility of each FF is restricted. This restricted mobility may be used to reduce the search space, and hence the number of constraints.

This range of motion of FF's can be derived from the skews calculated by the Bellman-Ford procedure (which calculates the minimum allowable skew value at each FF) [7], and the corresponding slacks in the constraint graph.

The idea of this work is that the skew values can be used to reduce the search space for the minarea retiming algorithm using restricted mobility. This is seen to translate to a smaller LP.

We will now show the relation between the Leiserson-Saxe approach and the ASTRA approach, and how a modified version of ASTRA can be used to derive bounds on the r variables in the Leiserson-Saxe method. Next, we show how these bounds can be used to prune the number of constraints in minarea Leiserson-Saxe retiming. Finally, we present an example to illustrate the method.

C. Deriving bounds for the r variables

The concept of restricted mobility is related to the "nearest" and "farthest" location that any FF can occupy under the target clock period. This is relatively easy to map on to the clock skew optimization problem. To understand this, we provide a brief review of the clock skew optimization problem. Given a pair of FF's, i and j, if the maximum delay of any purely combinational path connecting them is D_{ij}, then the following long-path constraint must hold:

$$x_i + D_{ij} \leq x_j + P \tag{5}$$

where x_i and x_j are the clock skews at FF's i and j, respectively, and P is the target clock period. For a specified clock period, this may be written as a difference constraint [10] as follows:

$$x_j - x_i \geq P - D_{ij} \tag{6}$$

Note that the right hand side of the above equation is a constant since the clock period is a specified value. For a given circuit, one may build a set of difference constraints with one such constraint for every pair of FF's that have a purely combinational path connecting them, and these difference constraints may be represented by a constraint graph. The Bellman-Ford algorithm may be applied to this graph to find the longest path in the graph. The final value associated with each vertex provides the required skew at that vertex and gives one possible set of skews that can achieve the clock period P. Note that this is not the only allowable set of skews, since slacks [7] in the arcs of the constraint graph can lead to other allowable solutions. Therefore, the first order of business is to determine bounds on the allowable skews at each FF.

ASTRA initializes all skews to 0 to achieve the minimum range of skews. To obtain the bounding skews we need to initialize all skews to $-\infty$. Now when the the Bellman-Ford algorithm [10] is applied to the constraint graph for a specified clock period, the as-late-as-possible[1] (ALAP) skews are calculated for the network. The as-soon-as-possible (ASAP) skews can be obtained by running the Bellman-Ford algorithm on the transpose of this constraint graph [10] (i.e., a graph with the same vertex set as the original graph, but with the edge directions reversed).

These ASAP/ALAP skews can be translated to ASAP/ALAP locations for FF's. These locations can be used to obtain bounds on the retiming variables of the Leiserson-Saxe approach, r, associated with the gates in the circuit as illustrated by the following example. Here we use the terms "ASAP locations" to refer to the case when all FF's are as close to the primary input as possible. Similarly the set of ALAP locations has all FF's as close to the primary output as possible. For ASAP locations any available slacks are used to avoid moving a FF in the direction of signal flow, while for ALAP locations they are used to avoid FF motion against the direction of signal flow.

Example: For the circuit in Figure 3, the locations for the FF's in the retimed circuit corresponding to the ASAP and ALAP skew solutions are shown in Figure 3 (a) and (b), respectively. This implies that during retiming, no FF will move across gates G1, G2, G5, G6, G7, G11 and G12; one FF each will move from the input to the output of gates G3 and G4, and either 0 or 1 FF will move from the input to the output of gates G8, G9 and G10. Referring to Section II-B.1 for the definition of the r variables, this implies that one may set the following bounds on the r variables.

(1) $r(u) = 0$ for $u \in \{G1, G2, G5, G6, G7, G11, G12\}$
(2) $r(u) = -1$ for $u \in \{G3, G4\}$, and
(3) $-1 \leq r(u) \leq 0$ for $u \in \{G8, G9, G10\}$. □

As explained in [7], FF's that have positive skews are moved in the direction opposite to the signal flow direction, and FF's with negative skews are relocated in the direction of signal flow (see Section II-A.1 for a brief explanation). The procedure for finding the ASAP and ALAP locations proceeds along the same lines as in [7], with a few variations described below. During this procedure, we also generate the bounds on the r variables.

When we consider the ASAP locations for the retimed FF's, the aim is to push the FF's as far as possible in a direction opposite to the direction of signal propagation. Therefore, each positive skew FF is moved as far as possible in the direction opposite to the signal flow, and each negative skew FF is moved as little as possible in the direction of signal flow. Therefore,

(1) for an FF with positive skew s that is being moved across a single-fanout gate p against the direction of signal propagation, the skew value after the relocation at input i of p is set to $s - \text{delay}(p)$. If this value is nonpositive, then the ASAP location has been found. For gates with multiple fanouts, $s = \min_{\text{all outputs}}(s_i)$, where s_i is the skew of the FF at the i^{th} output, as shown in Figure 4(a).

(2) for an FF with negative skew s that is being moved across a single-fanin gate p in the direction of signal propagation, the skew value after the relocation at output i of p is set to $s + \text{delay}(p) + \text{slack(i)}$, where slack(i) is the slack associated with the output i. This slack is defined as the amount by which the delay at output i may be increased before it becomes the critical output of p; by definition, the critical output has a slack of 0. If the new skew is nonnegative, then the ASAP location has been found. For gates with multiple fanins, $s = \max_{\text{all inputs}}(s_i)$, where s_i is the effective skew of the FF at the i^{th} output, as shown in Figure 4(b).

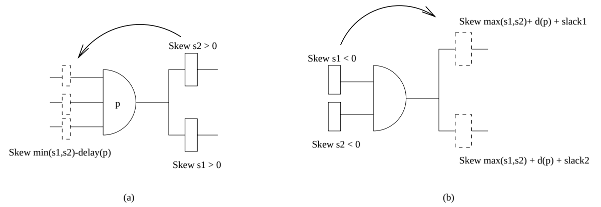


Fig. 4. Effective skews at FF's after ASAP retiming across a gate.

The ALAP locations can be found analogously with positive skew FF's being moved as little as possible in the direction opposite to the signal flow direction, and negative skew FF's being moved as much as possible in the signal flow direction.

While moving the FF's to ASAP and ALAP locations, subject to the specified clock period P, we count the number of FF's that traverse each gate; these lead us to upper and lower bounds, respectively, on the r variables for each gate. A FF moving from the inputs to the output of a gate decrements the count by one, while one moving from the output to the inputs increases it by one.

For the ASAP case, we move FF's as far as possible against the direction of signal propagation. In other words, we relocate the largest number of FF's possible from the output to the inputs of a gate. By the definition of the r

[1]The calculation of ASAP and ALAP times is a technique that is routinely used in scheduling in high-level synthesis; see, for example, [11] .

variables, this gives us an upper bound on r for the gates.

Similarly, the ALAP times are used to relocate the largest number of FF's that can move from the inputs of a gate towards its output, and this gives us a lower bound on the r values for the gates in the circuit. Therefore, this procedure provides upper and lower bounds on the r variable corresponding to each gate y of the form.

$$L_y \leq r(y) \leq U_y \qquad (7)$$

We will refer to L_y as the lower bound for gate y and to U_y as the upper bound of gate y. If $U_y = L_y = k_y$ we say that gate y is *fixed* or immobile since $r(y) = k_y$ is not really a variable any more. On the other hand if $U_y \neq L_y$ we say that gate y is flexible or mobile. Thus we can reduce the variable set V of the Leiserson-Saxe model to $V' \subseteq V$ the variable set of Minaret where

$$V' = \{v \in V | U_v \neq L_v\} \qquad (8)$$

D. Eliminating unnecessary constraints

In this section, we illustrate how the addition of bounds (derived previously) to the LP may be used to reduce the constraint set by removing redundant constraints. Let a constraint (i, j) in the constraint set C of the LP in Equation (4) for Leiserson-Saxe model be

$$\begin{array}{llll} & r_i - r_j & \leq & c_{ij} \\ \text{where} & c_{ij} & = & w(e_{ij}) \quad \forall e_{ij} \in E \\ & c_{ij} & = & W(i,j) - 1 \quad \forall\, D(i,j) > P \end{array} \qquad (9)$$

It can be seen from the bounds on $r(i)$ and $r(j)$ in Equation (7) that $r(i) - r(j) \leq U_i - L_j$. Therefore, if $U_i - L_j \leq c_{ij}$ then $r(i) - r(j) \leq c_{ij}$ is also true, and the constraint $((i, j))$ can be dropped. Thus the Leiserson-Saxe constraint set C can be reduced to the Minaret constraint set $C' \subseteq C$ where

$$C' = \{(i, j) \in C | \; U_i - L_j > c_{ij}\} \qquad (10)$$

Notice that constraints associated with fixed or immobile gates can be treated as bounds and need not be included in C'.

E. Reduced linear program

We use the Equations (8) and (10) to reduce the LP in Equation (4) to the following LP in Minaret

$$\begin{array}{rl} \min & \sum_{v \in V'} [(|FI(v)| - |FO(v)|) \cdot r(v)] \qquad (11) \\ \text{subject to} & r(u) - r(v) \leq c_{uv} \;\; \forall (u,v) \in C' \\ & L_u \leq r(u) \leq U_u \;\; \forall u \in V' \end{array}$$

where V' is the reduced variable set in Equation (8) and C' is the reduced constraint set in Equation (10). The contribution of the fixed gates to the objective function is a constant and can be ignored for optimization purposes.

F. An example

The following example illustrates the method and shows how the number of constraints can be reduced using our approach.

Consider the circuit example shown in Figure 5. As in the previous examples, we make the assumption that the gates have unit delays. We consider two possible clock periods of 2 units and 3 units in this example.

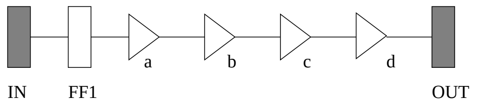


Fig. 5. Example illustrating the approach.

F.1 When $P = 2$ units

For a clock period of two units, the list of constraints generated by the approach in [6] is listed below.

$$\begin{array}{rll} \underline{\text{Circuit constraints}} & r(h) - r(a) & \leq 1 \\ & r(a) - r(b) & \leq 0 \\ & r(b) - r(c) & \leq 0 \\ & r(c) - r(d) & \leq 0 \\ & r(d) - r(h) & \leq 0 \\ \underline{\text{Period constraints}} & r(h) - r(c) & \leq 0 \\ & r(a) - r(c) & \leq -1 \\ & r(b) - r(d) & \leq -1 \end{array}$$

Note that
(a) the delay associated with the host node is zero, and
(b) the value of $r(h)$ is set to zero as a reference, so that it is not really a variable.
Therefore, this is a problem with <u>four</u> variables and <u>eight</u> linear constraints (of which three act as simple bounds).

In our approach, for a clock period of 2, we first find the bounding skews. The FF's at the input and output may not be moved, and therefore, the only movable FF is FF1, which is assigned an skew of -2 units. The correctness of this skew value is easy to verify since the only feasible location of FF1 under $c = 2$ is two delay units to the right of its current location. Therefore, we find that by using the concept of restricted mobility,

$$\begin{array}{rl} -1 \leq r(a) \leq -1 & \Rightarrow r(a) = -1 \\ -1 \leq r(b) \leq -1 & \Rightarrow r(b) = -1 \\ 0 \leq r(c) \leq 0 & \Rightarrow r(c) = 0 \\ 0 \leq r(d) \leq 0 & \Rightarrow r(d) = 0 \end{array}$$

Since all nodes are fixed, and all the constraints can be dropped, all of the constraints and variables have been eliminated!

F.2 When $P = 3$ units

With the clock period is set to 3 units, the list of constraints is

$$\begin{array}{rll} \underline{\text{Circuit constraints}} & r(h) - r(a) & \leq 1 \\ & r(a) - r(b) & \leq 0 \\ & r(b) - r(c) & \leq 0 \end{array}$$

$$\begin{array}{rrl} & r(c)-r(d) & \leq 0 \\ & r(d)-r(h) & \leq 0 \\ \underline{\text{Period constraints}} & r(h)-r(d) & \leq 0 \\ & r(a)-r(d) & \leq -1, \end{array}$$

As before, $r(h)=0$ is set as a reference, giving a problem with <u>four</u> variables (as before) and <u>seven</u> linear constraints (of which three act as simple bounds).

Under our approach, the relocated FF can reside either at the input of gate b, the output of gate b, or the output of gate c. Therefore, we have

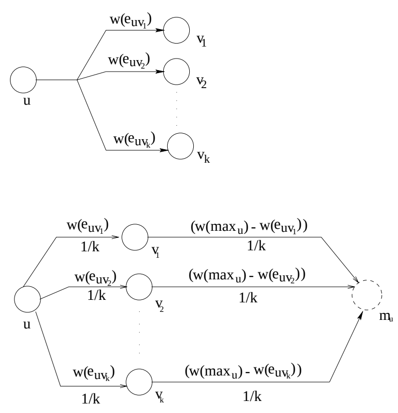
$$\begin{array}{rl} -1 \leq r(a) \leq -1 & \Rightarrow r(a)=-1 \\ -1 \leq r(b) \leq 0 & \\ -1 \leq r(c) \leq 0 & \\ 0 \leq r(d) \leq 0 & \Rightarrow r(d)=0 \end{array}$$

Using the bounds we drop all constraints but

$$r(b)-r(c) \leq 0$$

Therefore, we have reduced the problem complexity to <u>two</u> variables, each with fixed upper and lower bounds and <u>one</u> linear constraint. (Note that upper/lower bound constraints are typically much easier to handle in LP's than general linear constraints; in fact, in many cases, upper and lower bounds are actually helpful in solving the LP.)

IV. A more accurate area model



$w(e_{uv_1})$
$w(e_{uv_2})$
$w(e_{uv_k})$
u
v_1
v_2
v_k
$1/k$
$(w(max_u) - w(e_{uv_1}))$
$(w(max_u) - w(e_{uv_2}))$
$(w(max_u) - w(e_{uv_k}))$
m_u

Fig. 6. Maximum register sharing at multiple fanouts.

If a gate has more than one fanout with FF's, then we can combine (share) these FF's at the fanout to reduce the total number of FF's in the circuit. If the gates have constant delays this sharing does not change the clock period of the circuit. Therefore to accurately model the number of registers needed in a circuit, we need to take maximum register sharing into account. For this purpose we use the model presented by Leiserson-Saxe in [2].

This model, as shown in Figure 6, introduces a mirror vertex m_i for each gate i which has more than one fanout. Every edge e_{ij} in addition to having a weight $w(e_{ij})$, now also has a width $\beta(e_{ij})$. In Figure 6 the edge weights are shown above the edges, while the edge widths are shown below the edges. Let us consider a gate i which has k fanouts to gates $j=1\cdots k$. To model the maximum sharing of FF's, from each fanout gate $j=1\cdots k$, we add a extra edge to the mirror vertex, m_i with weight $w(e_{jm}) = w(max_i) - w(e_{ij})$. where $w(max_i) = \max_{\forall j \in FO(i)}(w(e_{ij}))$ is the maximum weight on any fanout edge of gate i. All the edges from the gate i to its fanouts j, and those from the fanouts to the mirror vertex have a width $1/k$, i.e., $\beta(e_{ij})=1/k$ and $\beta(e_{jm_i})=1/k$ for $j=1\cdots k$.

The original LP in Equation (4) is modified to include the effect of register sharing in this more accurate model. The LP now becomes

$$\min \sum_{v \in (V \cup M)} \left[\left(\sum_{\forall j \in FI(v)} \beta(e_{jv}) - \sum_{\forall j \in FO(v)} \beta(e_{vj}) \right) \cdot r(v) \right]$$

$$\begin{array}{rll} \text{subject to} & r(u)-r(v) \leq c_{uv} \;\; \forall (u,v) \in C & (12) \\ & r(u)-r(v) \leq c_{uv} \;\; \forall (u,v) \in C_m & \end{array}$$

where $M = \{m_i | i \in V \text{ and } |FO(i)| > 1\}$ is the set of all the mirror vertices, and $C_m = \{(j,m_i) | i \in M \text{ and } j \in FO(i)\}$ is the set of constraints due to the mirror vertices and is called the mirror constraint set. The weight c_{jm_i} on a constraint $(j,m_i) \in C_m$ of the form $r(j)-r(m_i) \leq c_{jm_i}$ is given by $c_{jm_i} = w(max_i) - w(e_{ij})$.

The objective function of the LP in Equation (12) now denotes the increase in the number of FF's assuming maximal sharing of FF's at the output of all gates. The terms for a gate together with the terms of its mirror vertex (if any) represents the number of FF's added when the gate is retimed. The mirror constraints C_m ensure that, even if the algorithm chooses to move one of the shared FF's across a gate, the cost of the remaining FF's will still count as a full FF.

A. Bounds on the mirror vertices

We will now show that the bounds on the r value of the mirror vertex m_i can easily be derived from the bounds on the fanout gates of i. Therefore the algorithm that calculates the bounds need only run on the original circuit model and the mirror vertices do not need be introduced explicitly in the graph. The *mirror constraints* for mirror m_i of gate i in Figure 6 are

$$\begin{array}{rcl} r(j)-r(m_i) & \leq & w(e_{jm_i}) \;\; \forall j=1\ldots k \\ \text{i.e. } r(m_i) & \geq & r(j)-w(e_{jm_i}) \;\; \forall j=1\ldots k \\ \text{i.e. } r(m_i) & \geq & r(j)-(w(max_i)-w(e_{ij})) \;\; \forall j=1\ldots k \\ \text{i.e. } r(m_i) & \geq & \max\limits_{j=1\cdots k}(r(j)+w(e_{ij}))-w(max_i) \end{array}$$

This gives us the lower bound, L_{m_i} on the values of $r(m_i)$ as

$$L_{m_i} = \max_{\forall j \in FO(i)} (L_j + w(e_{ij})) - w(max_i) \qquad (13)$$

To get the upper bound, U_{m_i} on the r value of the mirror vertex m_i we use the fact from [2] that after optimal retiming the weight on at least one of the edges to the mirror vertex will be zero, i.e., $\min_{j=1\cdots k}(w(e_{jm_i})) = 0$. Therefore,

$$\begin{aligned}\min_{j=1\cdots k}(w(e_{jm_i}) + r(m_i) - r(j)) &= 0\\ \text{i.e. } \min_{j=1\cdots k}(w(e_{jm_i}) - r(j)) &= -r(m_i)\\ \text{i.e. } r(m_i) &= \max_{j=1\cdots k}(r(j) - w(e_{jm_i}))\end{aligned}$$

Since $r(j) \le U_j \; \forall j = 1 \cdots k$ we have

$$\begin{aligned}\max_{j=1\cdots k}(r(j) - w(e_{jm_i})) &\le \max_{j=1\cdots k}(U_j - w(e_{jm_i}))\\ \text{which gives us } r(m_i) &\le \max_{j=1\cdots k}(U_j - w(e_{jm_i}))\end{aligned}$$

Thus the upper bound is

$$U_{m_i} = \max_{\forall j \in FO(i)}(U_j + w(e_{ij})) - w(max_i) \tag{14}$$

As in the case of gate variables we can reduce the Leiserson-Saxe mirror variables set M to $M' \subseteq M$ the Minaret mirror variable set where,

$$M' = \{m_i \in M | U_{m_i} \neq L_{m_i}\} \tag{15}$$

B. Eliminating unnecessary mirror constraints

As in Section III-D the Leiserson-Saxe mirror constraint set C_m can be reduced to the Minaret mirror constraint set $C'_m \subseteq C_m$ where

$$C'_m = \{(j, m_i) \in C_m \mid U_j - L_{m_i} > c_{jm_i}\} \tag{16}$$

C. Reduced linear program

As in Section III-E we can use the Equations (15) and (16) to obtain a reduced LP under the more accurate area model of this section.

$$\min \sum_{v \in \{V' \cup M'\}} \left[\left(\sum_{\forall j \in FI(v)} \beta(e_{jv}) - \sum_{\forall j \in FO(v)} \beta(e_{vj})\right) \cdot r(v)\right] \tag{17}$$

$$\begin{aligned}\text{subject to} \quad & r(u) - r(v) \le c_{uv} \;\; \forall (u,v) \in C'\\ & r(u) - r(v) \le c_{uv} \;\; \forall (u,v) \in C'_m\\ & L_u \le r(u) \le U_u \;\; \forall u \in (V' \cup M')\end{aligned}$$

V. Minarea retiming using Minaret

The ideas described so far have been encapsulated in Minaret (MINimum Area RETiming), a minimum area retiming algorithm for large sequential circuits. Minaret consists of three phases of finding the bounds, generating the LP and solving it. Each of these is described in detail in this section.

A. Deriving bounds on the r variables

As described in Section III-C the bounds are derived by finding the ASAP and ALAP locations of the FF's, using a modified form of ASTRA. An efficient method for calculating all FF-to-FF delays (D_{ij}'s) required by ASTRA, presented in [12], is used in Minaret. If the initial locations of FF's satisfy the target clock period all lower bounds must be nonpositive and all upper bounds must be nonnegative (i.e. $L_i \le 0$ and $U_i \ge 0 \;\forall i$), since $r(i) = 0 \;\forall i$ is a feasible solution. However, if the target clock period is smaller than the initial period we may be forced to move a FF from the inputs of a gate to its outputs to obtain any feasible (including the ASAP) locations. Thus it is possible to have a negative upper bound. Similarly it is possible to have a positive lower bound if the the target clock period is smaller than the initial period. The bounds on the mirror vertices for all gates with more than one fanout are derived from the circuit graph as described in Section IV-A.

B. Generating the linear program

We now describe how to obtain the reduced LP of Minaret given of Equation (17). The coefficients of the objective function, the constraint set C'_m and the circuit constraints in C' can be obtained directly from the circuit. Since for large circuits (with tens of thousand gates) $O(|V|^2)$ memory required by the Leiserson and Saxe method of generating period constraints [2] is not practical, we use the method from [6], which requires only $O(|V|)$ memory. We take advantage of the bounds obtained in Section V-A to modify this method to run faster, generating only the reduced constraint set C'.

The algorithm in [6] uses a combination of the Dijkstra's algorithm and the Bellman-Ford algorithm. The algorithm works by generating one (s^{th}) row of the W and the D matrix at a time. An ordered pair $(w(e_{ij}), -d(i))$, denoted by (a_i, b_i), is associated with each edge e_{ij} and is used to compute the shortest distance from vertex a source vertex s. A heap is maintained for each distinct value of a_i and is indexed by this value. Until all heaps are empty, we extract the node u at the top of the minimum index heap using the function pop-min(heap index). The fanouts of u are added to the appropriate heaps if their a_u or b_u values are updated (Bellman-Ford relaxation). At the end of this procedure $D(s,u) = -b_u$ and $W(s,u) = a_u$.

Note that to satisfy a clock period P, all we have to do is to ensure that each path whose delay is greater than P has at least one FF on it. The number of FF's on any path is monotonic with the path length because negative edge weights are not allowed. Due to the monotonicity of edge weights, if we ensure at least one FF on any sub-path, we are assured to have at least one FF on all paths containing this sub-path. This strategy can be used to prune the number of constraints generated as well as the gates examined.

Adding a period constraint from s to u is one way to ensure at least one FF on all paths from s to u. This observation presented by Leiserson-Saxe was used in [6] to prune the constraint set. The idea was to add a period edge to only the vertex v, reachable from s, that satisfies the following:

$$D(s,v) > P \text{ and } D(s,u) \leq P \;\forall\, u \text{ on } s \rightsquigarrow FI(v) \qquad (18)$$

where $s \rightsquigarrow FI(v)$ is a path from s to a fanin of v. Thus if the period constraint is added, the fanouts of u need not be relaxed. Similarly if the bounds on the r variables guarantee us at least one FF on any sub-path, we need not process any path containing this sub-path.

At the end of the ASTRA run to obtain the lower bounds all FF's are in the ALAP locations. If the delay of all the gates is not the same, it is possible that retimed circuit obtained by ASTRA with FF's in the ALAP locations may have some purely combinational paths with delays that are more than the target clock period P. However in practice, most of the other paths satisfy the target clock period. We will use this observation to further speed up the constraint generation process.

Consider a fixed gate a in the circuit at the end of the ALAP run. Now if none of the combinational paths starting at this gate violate the clock period, we have $W_{ALAP}(a,i) \geq 1$ if $D(a,i) > P \quad \forall i$. Since $W_{ALAP}(a,i) = W(a,i) + L_i - L_a$ we have $L_a - L_i \leq W(a,i) - 1$, or $L_a - L_i \leq c_{a,i}$. Since gate a is fixed $U_a = L_a$, we obtain $U_a - L_i \leq c_{a,i} \quad \forall i \in V$, which is guaranteed to be true, and hence all constraints starting from fixed gate a are redundant, and we do not need to generate them. Thus we must generate period constraints only from those fixed gates which have at least one purely combinational path starting from it with delay more than the clock period. Let us call this set V''.

The pseudo code presented bellow explains how we use the bounds on the r variables to generate the reduced constraint set C' efficiently.

```
P = target clock period;
$L_i \leq r(i) \leq U_i \;\; \forall i \in V$;
$S_k$= the $k^{th}$ heap;
$L_{min} = min(L_i) \;\; \forall i \in V$;
$\forall s \in V' \cup V''$
  {
    s = current vertex;
    $\forall v \in V, a_v = \infty$ and $b_v = 0$;
    $S_0 = \{s\}$, $a_s = 0$,  and  $b_s = -d(s)$;
    k = current register weight;
    do {
      $k = \min\{p \mid S_p \neq \emptyset\}$;
      if($k \geq U_s - L_{min} + 1$) break;
      $u$ = pop-min($S_k$) ;
      if($U_s - L_u \leq k - 1$) continue;
      if($-b_u > P$)
      add a period edge $c(s,u)$ with weight $a_u - 1$
      else {
          $\forall v \in FO(u)$ {
          if( $k - U_s + L_v < 1$)
              if(($a_v, b_v$) > ($a_u + s_{u,v}, b_v - d(v)$))
              heap-insert($S_{a(u)+s_{u,v}}, v$);
              }
          }
    } while($\exists\, p \mid S_p \neq \emptyset$)
  }
```

C. Solving the linear program

Like Equation (4) the LP in Equation (17) is also a dual of a minimum cost network flow problem. We found that it could be solved very efficiently using the network simplex algorithm from [13]. The network simplex method is a graph based adaptation of the LP simplex method which exploits the network structure to achieve very good efficiency. The upper and lower bounds on the r variables provide a initial feasible spanning tree. This tree has two levels only, with the host node as the root and all other nodes as leaves. To prevent cycling we construct the initial basis to be strongly feasible by using the appropriate bound (upper or lower) to connect a node to the root (host node). It is easy to maintain strongly feasible trees during the simplex operations, and details are given in [13].

Using the first eligible arc pivot rule with a wraparound arc list from [14] (page 417) gave us significant improvements in the run time. The dual variables (r variables) are directly available from the min cost flow solution. We could solve problems with more than 57,000 variables and 3.6 Million constraints in about 2.5 minutes.

VI. Experimental results

We now present area minimization results on circuits in the ISCAS89 benchmark suite, subject to a given clock period. We assume that all gates have a unit delay, although we emphasize that the algorithm is applicable when gates have non-unit delays. The target clock period is set to be the minimum achievable clock period for the circuit under retiming and is calculated using ASTRA. Therefore the results show the smallest number of FF's for the best clock period for all circuits. Since we did not have access to large circuits ($> 20,000$ gates) we created some large circuits (myex1 through myex5) by combining circuits from the ISCAS89 benchmark suite.

We present the results in two tables. Table I presents measures of the quality of minimum area retiming in the circuits. For each circuit, the number of gates $|G|$, the final number of FF's in the circuit from both ASTRA (A) and Minaret (M), and the CPU time T_{exec} of Minaret are shown. Also shown are two metrics on the circuits: F_{fx}, the percentage of gates found to be fixed and M_{avg}, the average mobility, i.e., the average value of $(U_y - L_y)$ over all gates in the circuit. Since $U_y - L_y$ gives the range in possible values (or mobility) of $r(y)$, M_{avg} is a measure of the average mobility in the circuit. The number of FF's both in ASTRA and Minaret are obtained under the more accurate area model of Section IV, after taking into account the maximum sharing of FF's at all nodes (including primary inputs) in the circuit. The execution times are in seconds on a DEC AXP system 3000/900 workstation, and includes the time spent in getting the bounds, generating the LP and solving it.

TABLE I

Minarea retiming using Minaret

Circuit	$\|G\|$	# FFs		F_{fx}	M_{avg}	T_{exec}
		A	M	(%)		(s)
s3271	1,572	306	168	49.38	0.81	0.25
prolog	1,601	358	122	49.77	0.55	0.27
s3384	1,685	438	167	14.31	3.15	2.44
s3330	1,789	331	110	63.46	0.39	0.22
s4863	2,342	201	138	28.46	0.97	5.24
s5378	2,779	555	173	36.12	0.85	1.28
s6669	3,080	719	305	40.02	0.76	2.20
s9234	3,270	205	134	14.62	1.55	6.18
s13207	7,791	629	446	21.49	2.96	10.38
s15850	9,617	571	525	24.15	1.52	38.81
s35932	16,065	1,729	1,729	55.27	0.54	7.56
s38584	19,253	1,428	1,427	14.22	2.13	65.07
s38417	21,370	1,616	1,370	0.88	4.35	146.92
myex1	25,717	5,146	2,293	4.75	2.51	169.10
myex2	28,946	5,655	2,022	8.73	2.26	160.47
myex3	35,353	8,052	3,279	5.22	2.65	489.52
myex4	40,661	11,591	2,803	1.80	4.12	421.50
myex5	56,751	11,488	3,378	4.95	3.98	799.64

TABLE II

Reduction in Problem Size with Minaret

Circuit	# Variables		# Constraints		$T_b/T_g/T_s$
	Minaret	Original	Minaret	Original	
s3271	1,079	2,038	5,492	43,506	0/0/0
prolog	1,039	1,992	5,304	37,319	0/0/0
s3384	1,870	2,166	47,916	49,487	0/2/0
s3330	858	2,212	4,595	30,409	0/0/0
s4863	2,170	2,995	92,873	597,323	0/5/0
s5378	2,385	3,664	19,170	168,530	0/1/0
s6669	2,539	4,100	20,041	341,750	0/2/0
s9234.1	3,366	3,893	54,610	137,962	0/6/0
s13207.1	7,303	9,180	38,630	491,561	1/9/1
s15850.1	8,740	11,332	38,318	1,046,108	1/36/2
s35932	10,306	21,716	53,087	389,647	1/5/2
s38584.1	20,486	23,390	97,268	11,450,472	2/54/8
s38417	25,731	25,923	1,507,162	1,628,544	3/91/52
myex1	31,476	32,922	812,872	3,275,567	3/146/19
myex2	31,704	34,493	398,697	17,185,252	4/131/24
myex3	42,604	44,812	5,693,689	16,978,788	5/404/81
myex4	48,415	49,214	2,635,127	8,186,340	7/312/104
myex5	57,488	60,241	3,600,681	24,316,717	10/638/152

For most ISCAS89 circuits M_{avg} was less than unity and the average over all ISCAS89 circuits was about 0.7. The percentage of fixed nodes F_{fx} varied from being as high as 95% to being below 1% (for s38417). We observed that circuits with a small critical part (perhaps a cycle in the retiming graph) and most gates being off the critical paths in the timing graph result in high values of M_{avg}. We note that these circuits are not very well suited for retiming since the small critical parts of the circuit restrict the rest of the circuit from achieving better clock periods. The CPU time T_{exec} depends on the the number of gates in the circuit $|G|$, the average mobility M_{avg} and F_{fx}.

In [6] the circuit s38584 needed 38 hours of CPU time while Minaret could retime it in about one minute. We point out, though, that such a comparison is not entirely fair since (a) the results are generated on different platforms and (b) the circuits used in [6] are modified ISCAS89 benchmarks and have a much smaller number of gates. For example s38584 has 7882 gates in [6] while it has 19,253 gates in this work.

In Table II we compare the size of the LP for Minaret and the original problem by presenting the number of variables and constraints for both methods. The number of variables include both the gate and mirror variables. The number of constraints in Minaret also include the upper and lower bounds. The number of constraints in the original method are obtained by using the pruning strategy suggested in [2]. The reduction in the number of constraints in Minaret depends on the average mobility M_{avg} and F_{fx}. However, since the original constraints are generated after some pruning, the reduction is affected by other factors as well. Table II also presents the breakup of the CPU time (in seconds) in terms of the time spent in using ASTRA to arrive at the bounds for the r variables (T_b), the time spent in generating the LP of Equation (17) (T_g) and the time needed to solve this LP by the network simplex method (T_s).

T_b depends on the number of gates and FF's in the circuit and the average mobility, M_{avg}, of the circuit. Phase A of ASTRA is dependent on the number of gates for obtaining the FF to FF delays and on the number of FF's for the Bellman-Ford runs. The CPU time taken for phase B of ASTRA depends on M_{avg}, since M_{avg} gives a measure of how many retiming (or movement of FF's across gates) are performed in phase B.

T_g is most strongly influenced by the number of flexible gates i.e. $(1 - F_{fx}) \cdot |G|$, which is equal to the number of rows of W and D matrices we need to generate. It is also influenced by M_{avg} in that it determines the number of gates processed for each row of W and D matrices. T_s depends on the size of the LP in terms of the number of variables and constraints.

VII. Conclusion

A fast algorithm for minarea retiming large circuits has been presented. The contributions of this work are twofold. Firstly, it reconciles the Leiserson-Saxe algorithm with the ASTRA algorithm and shows the relation between these two. Secondly, it utilizes this relationship to good purpose by modifying the ASTRA algorithm to make available information from the skew-retiming equivalence that is of great benefit in solving the minarea retiming problem under the Leiserson-Saxe framework.

Experimental results on benchmark circuits in the ISCAS89 benchmark suite have been presented, and the procedure is seen to give good benefits. The number of variables and the number of constraints was dramatically reduced in most cases. The entire ISCAS89 benchmark suite could be retimed in minutes. This works shows that it is possible to perform minarea retiming on large circuits in a reasonable amount of time.

Even though the average mobility M_{avg} is high and the fraction of fixed gates F_{fx} is low for the large circuits we created, we are still able to retime them in very reasonable amount of time. Because of the various pruning techniques used in Minaret the number of constraints in practical circuits grows at a far slower rate than $O(|G|^2)$.

Minaret also has a reduced memory requirement since a significant number of constraints are not stored. We found that for large circuits having constraints in millions, the memory requirement becomes a bottleneck. The reduction in the number of constraints also reduces both the problem generation and the problem solution time.

To the best of our knowledge, no other retiming algorithm incorporates pruning methods to reduce the number of variables. This reduction in the number of variables significantly reduces the problem generation time. Notice that due to the presence of mirror vertices the number of variables can be up to twice the number of gates in the circuit. Hence the reductions in the number of variables and constraints provided by Minaret are important to retime large circuits.

References

[1] C. Leiserson, F. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," in *Proceedings of the 3rd Caltech Conference on VLSI*, pp. 87–116, 1983.

[2] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, pp. 5–35, 1991.

[3] M. C. Papaefthymiou and K. H. Randall, "TIM: A timing package for two-phase, level-clocked circuitry," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 497–502, 1993.

[4] K. N. Lalgudi and M. Papaefthymiou, "DELAY: An efficient tool for retiming with realistic delay modeling," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 304–309, 1995.

[5] G. Even, I. Y. Spillinger, and L. Stok, "Retiming revisited and reversed," *IEEE Transactions on Computer-Aided Design*, vol. 15, pp. 348–357, Mar. 1996.

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

[7] 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*, vol. 15, pp. 1237–1248, Oct. 1996.

[8] N. Maheshwari and S. S. Sapatnekar, "A practical algorithm for retiming level-clocked circuits," in *Proceedings of the IEEE International Conference on Computer Design*, pp. 440–445, 1996.

[9] J. P. Fishburn, "Clock skew optimization," *IEEE Transactions on Computers*, vol. 39, pp. 945–951, July 1990.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. New York, NY: McGraw-Hill, 1990.

[11] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. New York, NY: McGraw-Hill, 1994.

[12] S. S. Sapatnekar, "Efficient calculation of all-pair input-to-output delays in synchronous sequential circuits," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. IV520–IV523, 1996.

[13] M. S. Bazaraa, J. J. Javis, and H. Sherali, *Linear Programming and Network Flows*. New York, NY: John Wiley, 1977.

[14] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows Theory, Algorithms and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1993.

Naresh Maheshwari received the B. E. degree from MNREC, Allahabad, India in 1992 and the M. Tech. degree from Indian Institute of Technology, Delhi in 1994. Currently he is a Ph.D. candidate at Iowa State University. He worked at the IBM Thomas J. Watson Research Center, Yorktown Heights in summer of 1996, and at IBM Rochester in summer of 1997. His research interests include high level and logic synthesis, particularly for large circuits. He has published several technical papers in this area, and he is a recipient of a best paper award at the 1997 Design Automation Conference, and a Lucent Technologies DAC Graduate Scholarship.

Sachin Sapatnekar received the B. Tech. degree from the Indian Institute of Technology, Bombay in 1987, the M. S. degree from Syracuse University in 1989, and the Ph. D. degree from the University of Illinois at Urbana-Champaign in 1992. He has had industrial experience at Texas Instruments Inc., Dallas during the summer of 1990 and at Intel Corporation, Santa Clara, during the summer of 1997. He was an assistant professor in the Department of Electrical and Computer Engineering at Iowa State University from 1992 to 1997. Since 1997, he has been an Associate Professor at the University of Minnesota. His current research interests lie in developing efficient techniques for the computer-aided design of integrated circuits, and are primarily centered around physical design, power, timing and simulation issues, and optimization algorithms.

He has published widely in the area of timing analysis and optimization and has coauthored a book, "*Design Automation for Timing-Driven Layout Synthesis*," (Kluwer Academic Publishers, Boston, MA). He has served as an Associate Editor for the *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, and has served on program committees for various conferences. He is a recipient of the NSF Career Award and a best paper award at the 1997 Design Automation Conference.