An Improved Algorithm for Minimum-Area Retiming

Naresh Maheshwari and Sachin S. Sapatnekar
Department of Electrical & Computer Engineering
Iowa State University, Ames IA 50010
naresh@iastate.edu *sachin@iastate.edu*

Abstract

The concept of improving the timing behavior of a circuit by relocating flip-flops is called retiming and was first presented by Leiserson and Saxe. The ASTRA algorithm proposed an alternative view of retiming using the equivalence between retiming and clock skew optimization. This work defines the relationship between the Leiserson-Saxe and the ASTRA approaches and utilizes it to solve the problem of retiming for minimum area. The new algorithm, Minaret, uses the linear programming formulation of the Leiserson-Saxe approach. The underlying philosophy of the ASTRA approach is incorporated to reduce the number of variables and constraints in the linear program. This reduction in the size of the linear program makes Minaret space and time efficient, enabling minimum area retiming of circuits with over 56,000 gates in under 15 minutes.

1 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, extending the Leiserson-Saxe method to handle variations of the original problem, for example, retiming level-clocked circuits [3, 4], improving the delay model [5], retiming with equivalent initial states [6] and retiming for low power [7]. It was only recently that algorithms for handling large VLSI circuits were introduced [8, 9]. The work in [8] presented efficient and clever implementations of the Leiserson-Saxe algorithms for both minarea and minperiod retiming. At about the same time, the ASTRA algorithm [9, 10] was published, displaying a different view of retiming by using an equivalence between clock skew and retiming. It proposed an minperiod retiming algorithm that was quite different from the Leiserson-Saxe approach. A similar approach for retiming level-clocked circuits was presented in [11].

For digital circuit design the most useful problem is that of minarea retiming. However, due to the high computational expense of this optimization its use has been limited. In this work we approach the problem of constrained minarea retiming through an amalgamation of the Leiserson-Saxe approach and the ASTRA approach. By utilizing the merits of both, we develop a much faster algorithm for minarea retiming.

As in the Leiserson-Saxe approach, Minaret also formulates the minarea retiming problem as a linear program (LP). The ASTRA approach is utilized to obtain reliable bounds on the variables in this LP. These bounds are then used to reduce the problem size by reducing both the number of variables and the number of constraints. Thus 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. Experimental results show that large circuits (with tens of thousands of gates can be retimed for minimum area in minutes). As in most of the references on retiming listed above, this paper assumes the circuit to be composed of gates with constant delays.

The paper is organized as follows. In Section 2, the outline of the ASTRA and Leiserson-Saxe approaches is presented. Next, in Section 3, we show the relationship between these two, and utilize it to prune the constraint set for minarea retiming. Experimental results are presented in Section 4, followed by concluding remarks in Section 5.

[1] This work was supported in part by the National Science Foundation under award MIP-9502556.

0-89791-847-9/97/0006/$3.50 DAC 97 - 06/97 Anaheim, CA, USA

2 Background

We first briefly describe the ASTRA approach for minperiod retiming, and the Leiserson-Saxe approach for minarea retiming. These approaches will later be combined to form “Minaret”, an improved algorithm for minimum area retiming.

2.1 The ASTRA Algorithm

The introduction of clock skew at a FF has an effect that is similar to moving it across combinational logic module boundaries (gates) [12]. The effect of applying a positive skew on a FF is equivalent to moving it from the inputs of a gate to the outputs. Similarly application of a negative clock skew is equivalent to moving it from the output to the inputs of a gate. Hence both retiming and clock skew are equivalent and can be used for timing optimization of sequential circuits. Since clock skew is a continuous optimization while retiming is a discrete one, the minimum clock period achievable by application of clock skews may not be obtained by retiming. This relationship between skew and retiming motivates the following two-phase solution to the minperiod retiming problem in the ASTRA approach [9].

Phase A : The clock skew optimization problem is solved to find the optimal values 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 on a constraint graph.

Phase B : The skew solution is translated to a retiming by relocating FF's 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 [10].

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 that 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 optimization [12].

2.2 The Leiserson-Saxe Algorithm

2.2.1 Notation

A sequential circuit can be represented by a directed graph $G(V, E)$, where each vertex v corresponds to a gate, and a directed edge e_{uv} represents a connection from the output of gate u to the input of gate v, through zero, one 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 retiming label $r(v)$ for a vertex v represents the number of registers moved from its output towards its inputs. The weight of an edge e_{uv} after retiming, denoted by $w_r(e_{uv})$ is given by

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

One may define the weight $w(p)$ of any path p originating at vertex u and terminating at vertex v (represented as $u \rightsquigarrow v$), 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

$$c = \max_{\forall\ p | w(p)=0} \{d(p)\} \quad (2)$$

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

$$W(u, v) = \min_{\forall\ p: u \rightsquigarrow v} \{w(p)\} \quad (3)$$

$$D(u, v) = \max_{\forall\ p: u \rightsquigarrow v \text{ and } w(p) = W(u,v)} \{d(p)\} \quad (4)$$

The matrices are defined for all pairs of vertices (u, v) such that there exists a path $p : u \rightsquigarrow v$ 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.

2.2.2 The Minarea Retiming Problem

The minarea retiming problem can be formulated as the following linear program:

$$\begin{array}{llll} \text{minimize} & \sum_{v \in V} (|FI(v)| - |FO(v)|) \cdot r(v) & & (5) \\ \text{subject to} & r(u) - r(v) \leq w(e_{uv}) & \forall e_{uv} \in E & \\ & r(u) - r(v) \leq W(u, v) - 1 & \forall D(u, v) > c & \end{array}$$

where $|FI(v)|$ and $|FO(v)|$ represent the number of fanins and fanouts of a gate v.

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 nonnegative. We will refer to these constraints as *nonnegativity 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*.

This formulation assumes that all FF's fanout to exactly one gate. However, in physical circuits a FF can fanout to several gates. Thus FF's at the fanouts of a gate can be combined or shared. To accurately model the number of FFs in a circuit we need to take this sharing into account. For this purpose we use the model given by Leiserson-Saxe in [2], which introduces for every gate u with multiple fanouts a mirror vertex m_u. The objective function of the LP is also modified as described in [2].

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

3 The Reduced Linear Program

On practical circuits, it is found that the number of period constraints is phenomenally large; considerably more than the number of nonnegativity constraints. However, it is also true that a large number of these constraints are redundant as they are implied by some of the other constraints, and 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 reliable bounds on the variable values, and to use these bounds to prune the 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 by simplifying the problem we can generate the LP efficiently and also solve it faster.

3.1 The Concept of Restricted Mobility of FF's

Consider the circuits in Figure 1(a) and 1(b). Assuming unit gate delays, the minimum achievable value of the clock period is 4.0

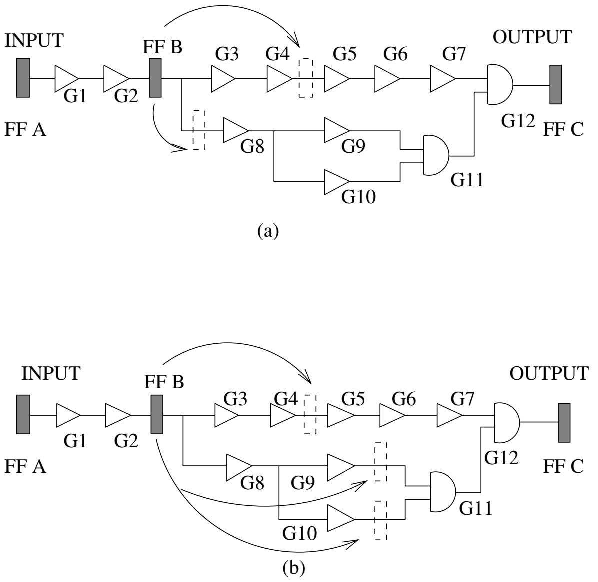

Figure 1: Possible FF locations after retiming.

units; each of these two circuits achieves this period, but the latter utilizes more FF's. In order 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 paths to FF C are at the input to gate G8, or at the output of gate G8 or the outputs of gates (G9,G10); no other locations are permissible[1].

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 range can be derived from the skews calculated by the Bellman-Ford procedure (which calculates the minimum allowable skew value at each FF), and the corresponding slacks in the constraint graph. Since the motion of the FF's is closely related to the retiming labels (r) of the gates, this restriction on the mobility of FF's also restricted the range of values the retiming labels can take.

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 into a smaller linear program.

We will now show the relation between the Leiserson-Saxe approach and the ASTRA approach, and how the a modified version of ASTRA can be used to obtain the 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 LP of Equation (5). Finally, we present an example to illustrate the method.

3.2 Deriving Bounds for the r Variables

The concept of restricted mobility is related to the "earliest" and "latest" 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 $\overline{d}_{ij}$, then the following long-path constraint must hold:

$$x_i + \overline{d}_{ij} + T_{setup} \leq x_j + c \tag{6}$$

where x_i and x_j are the skews at FF's i and j, respectively, c is the clock period as before, and T_{setup} is the setup time for FF j. For a specified clock period, this may be written as a difference constraint [13] as follows:

$$x_j - x_i \geq c - \overline{d}_{ij} - T_{setup} \tag{7}$$

Note that the right hand side of the above equation is a constant. 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 may be represented by a constraint graph. It is shown in [10] that the Bellman-Ford algorithm may be applied to this graph to find the longest path in the graph; the final values associated with each vertex provides the required skew at that vertex and gives *one* possible set of skews that can achieve the clock period c. Note that this is not the only allowable set of skews, since slacks 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 skew at each FF.

When the version of the Bellman-Ford algorithm in [13] is applied to the constraint graph for a specified clock period, the as-soon-as-possible (ASAP) skews are calculated for the network. It is possible to easily modify the Bellman-Ford algorithm to calculate the as-late-as-possible (ALAP) skew for the constraint graph[2]. Note that these skews could be either negative or positive.

These ASAP/ALAP skews can be translated to ASAP/ALAP locations for FF's, which in turn can are translated to bounds on the retiming variables from the Leiserson-Saxe approach, r, associated with the gates in the circuit as illustrated by the following example.

Example: For the circuit in Figure 1, the locations for the FF's in the retimed circuit corresponding to the ASAP and ALAP skew solutions are shown in Figure 1 (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 2.2.1 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 [10], 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 2.1 for a brief explanation). The procedure for relocation here to find the ASAP and ALAP locations proceeds along the same lines as in [10], with a few variations described below. During this procedure, we also obtain 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.

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 c, we count the number of FF's that

[1] Unlike this specific example, it is not always possible to say which FF was relocated to exactly which location. However, our method does not require this information.

[2] The calculation of ASAP and ALAP times is a technique that is routinely used in scheduling in high-level synthesis; see, for example, [14].

traverse each gate; this count gives us the upper and lower bound, respectively, on the r variable for the gate.

For the ASAP locations, 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 that move from the output to the input of any gate. By the definition of the r 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 move from the input of a gate towards the 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.

$$a_y \leq r(y) \leq b_y \tag{8}$$

We will refer to a_y as the lower bound of gate y and to b_y as the upper bound of gate y. If $a_y = b_y = k$ we say that gate y is *fixed* since $r(y) = k$ is not really a variable anymore. Thus we can reduce the number of variables in the linear program. The gates for which $a_y \neq b_y$ are said to be *flexible*.

The bounds on the r value of the mirror vertex m_u of a gate u can be easily derived from the bounds on the fanouts of gate u. Therefore the algorithm that calculates the bounds need only run on the original circuit model and the mirror vertices need not be introduced explicitly in the circuit graph. The bounds on the mirror vertex are

$$a_{m_u} = \max_{\forall\ j \in FO(u)} (a_j + w_{uj}) - w_{max_u}$$
$$b_{m_u} = \max_{\forall\ j \in FO(u)} (b_j + w_{uj}) - w_{max_u}$$

where $w_{max_u} = \max_{\forall\ j \in FO(u)} (w_{uj})$.

3.3 Eliminating Redundant Constraints

For large circuits, generation of the period constraints of Equation (5) is expensive in terms of both memory and computation requirements. We will now show how the bounds obtained in Section 3.2 can be used to reduce both these requirements.

From Equation (2) it is clear that in edge-triggered circuits the requirement for correct operation under a given clock period P is that all paths with delay greater than P must have at least one FF on them. Since path weights are monotonic, if a subpath has at least one FF on it, then any path containing this subpath also has at least one FF on it.

The period constraints in Equation (5) ensure that this condition is satisfied after retiming. While generating period constraints from a gate s if we add a period constraint from gate s to gate u to ensure that there will be at least one FF on the path $s \rightsquigarrow u$ after retiming. Then we do not need to consider any path via gate u to its fanouts for period constraints. Since this period constraint from gate s to gate u assures us that any path from gate s passing through gate u at least one FF on it. This idea was used in [8] to prune the period constraints by adding a period constraint only to a vertex u, reachable from s, that satisfies the following:

$$D(s,u) > c \text{ and } D(s,v) \leq c\ \forall\ v \text{ lying on } s \rightsquigarrow FI(u) \tag{9}$$

In the presence of the bounds suppose for a vertex v, reachable from w we have $W(w,v) + a_v - b_w > 1$, then we are assured that any retiming will have at least on FF on any path from w to v. This makes any period edge between gate w and v redundant. Thus while generating the period constraints from a gate w we do not process the fanouts of any gate v for which the inequality $W(w,v) + a_v - b_w > 1$ is satisfied. We observed that incorporating this rule in the period constraint generation algorithm of [8] gave us significant improvements in the CPU time.

To keep the memory requirements of the LP in Equation (5) low we must identify and drop the redundant constraints as they are generated. Any constraint of the form $r(u) - r(v) \leq c_{uv}$ is redundant if $b_u - a_v \leq c_{uv}$. Hence when generating the period constraints we only keep the constraints for which $b_u - a_v > c_{uv}$. Note that the constraints associated with fixed gates are redundant since the r value of fixed gate is not a variable.

The techniques presented in this section yield a significant reduction in the number of constraints (both period and nonnegativity) in the LP. Thus by adding a upper bound and a lower bound on each retiming variable in the LP of Equation (5) we can reduce the total number of constraints. These upper/lower bounds are typically much easier to handle in linear programs than general linear constraints; in fact, for the network simplex algorithm [15] used by Minaret these, upper and lower bounds are actually helpful in solving the LP.

3.4 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 2. 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.

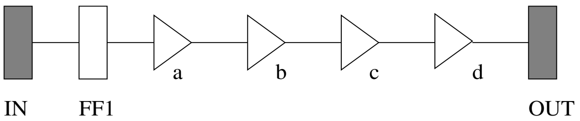

Figure 2: Example illustrating the approach.

When $c = 2$ units

For a clock period of two units, the list of constraints generated by the approach in [8] is listed below.

<u>Nonnegativity constraints</u>	$r(h) - r(a)$	≤ 1
	$r(a) - r(b)$	≤ 0
	$r(b) - r(c)$	≤ 0
	$r(c) - r(d)$	≤ 0
	$r(d) - r(h)$	≤ 0
<u>Period constraints</u>	$r(h) - r(c)$	≤ 0
	$r(a) - r(c)$	≤ -1
	$r(b) - r(d)$	≤ -1

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.

In our approach, for a clock period of 2, we first calculate the ASAP and ALAP skews and then translate them into the ASAP and ALAP locations getting the bounds on the r variables in the process. Since the FF's at the input and output may not be moved, therefore, the only movable FF is FF1. The ASAP skew at FF1 is -2.0 units and hence the ASAP location of FF1 is at the output of gate b. Since to arrive at this ASAP location FF1 must move forward across gate a and gate b the upper bounds on the r variables of both the gate a and the gate b is -1. Since no FF's move across gate c and gate d their upper bounds are both 0.

It so happens in this circuit that the ALAP skew of FF1 is also -2.0 units and the ALAP location of FF1 is also at the the output of gate b. Thus the lower bound on the r variables of gate a and gate b is -1, while it is 0 for gate c and gate d. Therefore, we find that by using the concept of restricted mobility of FF's we can derive these bounds

$$\begin{aligned} -1 \le r(a) \le -1 &\quad \Rightarrow r(a) = -1 \\ -1 \le r(b) \le -1 &\quad \Rightarrow r(b) = -1 \\ 0 \le r(c) \le 0 &\quad \Rightarrow r(c) = 0 \\ 0 \le r(d) \le 0 &\quad \Rightarrow r(d) = 0 \end{aligned}$$

In other words, all of the constraints and variables have been eliminated ! The correctness of these bounds is easily verified since to obtain a clock period of 2.0 units the only solution is to locate FF1 at the output of gate b, and hence there is only one feasible set of r values for the gates.

When $c = 3$ units

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

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

As before, $r(h) = 0$ is set as a reference, giving a problem with four variables (as before) and seven linear constraints.

The relocated FF can reside either at the input of gate b, the output of gate b, or the output of gate c. To arrive at the ASAP location, FF1 must move across only gate a, therefore, the upper bound on gate a is -1, while on gate b, gate c and gate d it is 0 since no FF's moved across them. The ALAP location of FF1 is at the output of gate c, to arrive at which FF1 has to move across gate a, gate b and gate c. This gives us the following bounds

$$\begin{aligned} -1 \le r(a) \le -1 &\quad \Rightarrow r(a) = -1 \\ -1 \le r(b) \le 0 & \\ -1 \le r(c) \le 0 & \\ 0 \le r(d) \le 0 &\quad \Rightarrow r(d) = 0 \end{aligned}$$

These bounds may now be applied to eliminate all of the nonnegativity and period constraints except

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

Therefore, we have reduced the problem complexity from four variables and seven constraints to two variables, each with fixed upper and lower bounds one linear constraint.

4 Experimental Results

We performed minarea retiming using Minaret on the complete ISCAS89 benchmark suite but present results only on the larger circuits. The results presented are for unit delay gates, 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. Thus we get the smallest number of FFs for the best clock period. We artificially created some large circuits (myex1,myex2 and myex3) which are derived from the circuits in the ISCAS89 benchmark suite.

We present the results in two tables. Table 1 presents the quality of minimum area retiming in the circuits. For each circuit, the number of gates $|G|$, the target clock period P, the number of FF's from applying ASTRA for minperiod retiming, the number of final FF's provided by Minaret, and CPU time T_{exec} on a DEC AXP system 3000/900 workstation are shown. The number of FFs both for ASTRA and Minaret are obtained after taking into account the maximum sharing of FFs at all nodes in the circuit.

Table 1: Minarea Retiming using Minaret

Circuit	$\lvert G\rvert$	P	# FFs		T_{exec}
			ASTRA	Minaret	
s3330	1,789	14.0	331	110	0.22s
s5378	2,779	21.0	555	173	1.28s
s6669	3,080	26.0	719	305	2.20s
s9234.1	3,270	38.0	205	134	6.18s
s13207.1	7,791	51.0	629	446	10.38s
s15850.1	9,617	63.0	571	525	38.81s
s35932	16,065	27.0	1,729	1,729	7.56s
s38584.1	19,253	48.0	1,428	1,427	65.07s
s38417	21,370	32.0	1,616	1,370	146.92s
myex1	28,946	45.0	5,655	2,022	160.47s
myex2	40,661	35.0	11,591	2,803	421.50s
myex3	56,751	47.0	11,488	3,378	799.64s

The significant decrease in the number of FF's obtained by Minaret proves the importance of minarea retiming. The results also show that Minaret can perform minarea retiming for large circuits in very reasonable time. About 75% of the total execution time T_{exec} was spent in computing the period constraints, and the rest in solving the LP.

Table 2: Reduction of Variables and Constraints in Minaret

Circuit	F_0	F_{avg}	Reduction	
			N	A
s3330	63.46%	0.39	61.21%	84.84%
s5378	36.12%	0.85	34.91%	88.63%
s6669	40.02%	0.76	38.07%	94.14%
s9234.1	14.62%	1.55	13.54%	60.42%
s13207.1	21.49%	2.96	20.45%	92.14%
s15850.1	24.15%	1.52	22.87%	96.34%
s35932	55.27%	0.54	52.54%	86.38%
s38584.1	14.22%	2.13	12.42%	99.18%
s38417	0.88%	4.35	0.74%	7.45%
myex1	8.73%	2.26	8.09%	97.68%
myex2	1.80%	4.12	1.62%	67.81%
myex3	4.95%	3.98	4.57%	85.20%

Table 2 presents the reduction in the number of variables (N) and the number of constraints (A), obtained by Minaret. The basis of this percentage calculation is the LP obtained after applying Equation (9) only. Also shown are two metrics on the circuits: F_0, the percentage of gates found to be fixed, and F_{avg}, the average flexibility, i.e., the average value of $(b_y - a_y)$. The metric $(b_y - a_y)$ gives the possible range in r value of gate y and hence is a measure of its flexibility. The reduction in problem size was seen to be closely related to the flexibility in the circuit, which is as expected since Minaret uses restricted mobility to reduce the problem size. As is clear from the table Minaret obtains significant reduction in the number of constraints over the work in [8]. For example, for the circuit s38584.1 the number of constraints. reduced from 11 million to less than 0.1 million. Although not shown in the results, the time spent in computing the bounds was found to be insignificant at about 1% to 2% of the total execution time. Thus Minaret would be useful even if a circuit has a unusually high flexibility.

We also note that in [8] the circuit s38584 needed 39 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 [8] are modified ISCAS89 benchmarks and have a much smaller number of gates. (for example, s38584.1 has 7882 gates in [8] while the original benchmark used here has 19,253 gates).

Table 3: Area-Delay Tradeoff

Circuit	P	F_{avg}	F_0	Reduction	
				N	A
s1269	19.0	0.49	54.77%	51.50%	93.27%
s1269	22.0	0.69	36.52%	32.03%	79.36%
s1269	25.0	0.79	25.96%	22.88%	74.14%
s1269	28.0	0.87	18.43%	16.34%	72.36%
s1269	32.0	0.92	13.57%	11.50%	72.55%
s3330	14.0	0.39	63.46%	61.21%	84.89%
s3330	16.0	0.69	35.59%	33.36%	64.23%
s3330	18.0	0.87	22.40%	20.98%	53.10%
s3330	20.0	0.94	19.14%	17.86%	50.99%
s3330	22.0	0.99	15.98%	15.05%	52.84%
prolog	13.0	0.55	49.77%	47.84%	85.79%
prolog	15.0	0.82	27.37%	25.50%	73.63%
prolog	17.0	0.94	21.23%	19.73%	67.02%
prolog	19.0	1.00	17.72%	16.62%	64.09%
prolog	21.0	1.06	15.67%	14.86%	59.17%

Table 3 shows the variation in problem size reduction and circuit flexibility with respect to different clock periods. As expected the circuit flexibility increases with the target clock period. Thus, Minaret gives better problem size reduction on the more useful problem of retiming circuits under tight clock periods.

5 Conclusion

An improved algorithm for minarea retiming 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 them. 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. The bounds obtained by the ASTRA algorithm provide a efficient way to significantly reduce the size of the linear program in 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 were dramatically reduced. The entire ISCAS89 benchmark suite could be retimed for minarea in less than five minutes. Minaret could also retime a circuit (myex3) with over 56,000 gates in under 15 minutes.

Minaret also has a reduced memory requirement since a significant number of constraints are not stored. In our implementation of [8] we found that large circuits can have constraints in millions, which makes the memory requirement a bottleneck. 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 is important because (a) it significantly reduces the problem solution time and (b) it helps in pruning the number of constraints significantly.

We also attempted to reduce the problem size by other means like use of transitivity to eliminate constraints. For example, given the constraints $x_k - x_j \leq c_{kj}$,$x_j - x_i \leq c_{ji}$, and $x_k - x_i \leq c_{ki}$, if $c_{kj} + c_{ji} \leq c_{ki}$, then the last of the three constraints is redundant. However the reduction in constraints by this method was much less than that achieved by Minaret even for circuits like s38417 where Minaret did not perform well. Furthermore, the computational complexity of redundant constraint removal by transitivity is orders of magnitude more than the one used in Minaret, and unlike Minaret, transitivity can only reduce the number of constraints but not the number of variables in the problem.

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] A. Ishii, C. E. Leiserson, and M. C. Papaefthymiou, "Optimizing two-phase, level-clocked circuitry," in *Advanced Research in VLSI and Parallel Systems: Proceedings of the 1992 Brown/MIT Conference*, pp. 246–264, 1992.

[4] B. Lockyear and C. Ebeling, "Optimal retiming of level-clocked circuits using symmetric clock schedules," *IEEE Transactions on Computer-Aided Design*, vol. 13, pp. 1097–1109, Sept. 1994.

[5] 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.

[6] 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.

[7] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 398–402, 1993.

[8] 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.

[9] R. B. Deokar and S. S. Sapatnekar, "A fresh look at retiming via clock skew optimization," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 310–315, 1995.

[10] 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.

[11] 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.

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

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

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

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