Retiming Level-Clocked Circuits for Latch Count Minimization

Naresh Maheshwari
Dept. of Electrical & Computer Engineering
Iowa State University, Ames IA 50011
naresh@iastate.edu

Sachin S. Sapatnekar
Dept. of Electrical & Computer Engineering
University of Minnesota, Minneapolis, MN 55455
sachin@ee.umn.edu

Abstract—
Retiming is a powerful transformation that can minimize the number of memory elements in a sequential circuit under clock period constraints. Recent research has led to the development of extremely fast algorithms for retiming edge-triggered circuits. However, level-clocked circuits have the potential to operate faster and require less memory elements than edge-triggered circuits. This paper addresses the harder problem of retiming level-clocked circuits, and presents a space and time efficient retiming algorithm, called Minaret-L, whose performance compares well with the state of art edge-triggered retiming methods. Minaret-L can perform latch count minimization for large circuits (over 56,000 gates) that use multi-phase symmetric clock schedules very efficiently (under 1.5 hours).

I. Introduction

Retiming [1], [2] relocates the memory elements in a circuit, without changing its functionality, to optimize some cost function, e.g, clock period [3], number of latches [4], power [5] or testability [6]. The problem of finding the minimum clock period without regard to area is called *minperiod retiming*, and that of minimizing the number of memory elements while satisfying a target clock period is called *minarea retiming.*

The memory elements in a circuit may be either edge-triggered *flip-flops* (FF's) or level-sensitive *latches.* In a level-clocked circuit the latch is transparent during the active period of the clock, and the delay through a combinational logic path can be longer than one clock cycle, as long as it is compensated by shorter paths delays in the subsequent cycles. This transparent nature of the latch provides more flexibility both in terms of the minimum clock period achievable and the minimum number of memory elements needed. Unfortunately this transparency also perplexes the analysis of level-clocked circuitry because data can ripple through several stages of memory elements before its propagation is complete. This complicates the design of level-clocked circuits, making the need of good automation tools acute.

As a result several efforts have been made to retime circuits with level-triggered latches based on the Leiserson-Saxe approach [2], e.g. [7], [8], [9], [10]. Since minperiod retiming algorithms do not pay any regard to the number of latches in the retimed circuit, they tend to significantly increase the number of latches. Hence there is a real need for constrained minarea retiming methods. TIM [9] provides one such minarea retiming method for two-phase symmetric clocking schedules. The minarea retiming problem for level-clocked circuits can be formulated as a linear program (LP) similar to the one in [2]. Due to the transparent nature of the latches the number of constraints in this LP is extremely large, almost $\frac{|G|^2}{2}$ for a circuit with $|G|$ gates. This places heavy time and space requirements on minarea retiming and TIM is not capable of retiming large circuits (with tens of thousands of gates) for minimum area. The objective of this work, called Minaret-L, is to be able to retime level-clocked circuits with tens of thousands of gates in reasonable time.

An efficient method for minarea retiming of edge-triggered circuits was presented in [12]. This was achieved by utilizing the observation that in edge-triggered circuits, if a sub-path satisfies the timing constraints, then any path containing this sub-path will also satisfy the timing constraints. Unfortunately this is not true in level-clocked circuits, because of the the transparent nature of latches. Therefore the techniques of [12] cannot be applied to level-clocked circuits.

The ASTRA algorithm in [3], presented a different approach to retiming of edge-triggered circuits utilizing the retiming-skew relation. This relation was extended to level-clocked circuits in [11], which presented an algorithm for minperiod retiming of level-clocked circuits under a general multi-phase clocking scheme. Both of these methods are capable of minperiod retiming of large circuits but do not address the harder problem of minarea retiming.

The work in [4], presented efficient techniques to obtain bounds on the variables of the minarea LP for edge-triggered circuits. It then used these bounds to further reduce the size of the LP and the time required for generating the LP. Utilizing the retiming-skew relation for level-clocked circuits from [11], bounds on the variables of the minarea LP are obtained in this

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. Reprinted from TAU 97

work, along the lines of [4]. However due to the transparent nature of latches, unlike edge-triggered circuits, the techniques of [4] cannot be used to reduce the time required to generate the LP in level-clocked circuits. This presents a major hurdle in retiming large level-clocked circuits for minimum area, because in the absence of any efficiency-enhancing technique, the minarea LP can not be generated in a reasonable time. This work also presents new techniques for reducing the time taken to generate the minarea LP in level-clocked circuits.

The rest of the paper is organized as follows: Section II presents the required background, followed by techniques for reducing the linear program in Section III. We then present efficient techniques for generating this reduced LP in Section IV, and solving it in Section V. Section VI presents experimental results on the ISCAS-89 benchmarks, and Section VII concludes the paper.

II. Background

A. The Clock Model

As in [9] a k-phase clock is a set of k periodic signals $\Pi = \langle \phi_1, \gamma_1, \phi_2, \gamma_2, \ldots \phi_k, \gamma_k \rangle$, where ϕ_i is the active duration of phase i and γ_i is the gap between the falling edge of phase i and the rising edge of phase $(i+1)$. We denote the duration of phase i by $\pi_i = \phi_i + \gamma_i$. We overload the symbol Π to also denote the clock period $\Pi = \sum_{i=1}^{k} \pi_i$. A clocking scheme is symmetric if all phases have the same duration and active intervals, i.e. if $\phi_i = \phi \; \forall i = 1 \cdots k$ and $\gamma_i = \gamma \; \forall i = 1 \cdots k$. Thus for a k phase symmetric clocking scheme $\Pi = k \cdot \pi$ and $\pi = \phi + \gamma$. In this work we consider only symmetric clocking schemes.

B. The Circuit Model

The graph model used here is identical to the one in [2]. A sequential circuit is 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 or more latches. Each vertex has a fixed delay $d(v)$. Each edge has associated with it a weight $w(e_{uv})$ and a width $\beta(e_{ij})$. The weight is the number of latches between the output of gate u and the input of gate v. The width of an edge is the area cost of placing one latch on it.

To accurately model the number of latches needed in a circuit we take into account maximum latch sharing. To achieve this each gate with multiple fanouts has a mirror vertex of zero delay, associated with it [2].

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 $w_r(e_{uv}) = w(e_{uv}) + r(v) - r(u)$.

The retiming label $r(v)$ for a vertex v represents the number of latches moved from its output towards its inputs. One may define the weight $w(p)$ of any path $p : u \rightsquigarrow v$, originating at vertex u and terminating at vertex 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. Similarly $w_r(p)$ is the sum of the weights on the edges on p after retiming, and is given by

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

For a level-clocked circuit to be properly clocked the delay between any two gates should be less than the time available [10], i.e.,

$$d(p) \leq (w_r(p) + 1) \cdot \pi + \phi \tag{2}$$

After substituting Equation 1, this constraint can be rewritten as

$$r(u) - r(v) \leq w(p) - \frac{d(p)}{\pi} + 1 + \frac{\phi}{\pi} \tag{3}$$

Clearly if there are multiple paths from u to v only the tightest constraint (minimum right hand side) is irredundant. We denote the minimum value of $\left[w(p) - \frac{d(p)}{\pi} \right]$ over all paths from u to v by $\delta(u, v)$, i.e.

$$\delta(u, v) = \min_{\forall p : u \rightsquigarrow v} \left(w(p) - \frac{d(p)}{\pi} \right) \tag{4}$$

Let us define $\Delta(u, v)$ as

$$\Delta(u, v) = \left\lfloor \delta(u, v) + \frac{\phi}{\pi} + 1 \right\rfloor \tag{5}$$

Since the retiming variables $r(u)$ and $r(v)$ are integers, we can rewrite Equation (3) as

$$r(u) - r(v) \leq \Delta(u, v) \tag{6}$$

C. The Minarea Retiming Problem

The constrained minarea retiming problem for a target period Π is formulated as the following LP [9]:

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

$$\begin{aligned} \text{subject to} \quad & r(u) - r(v) \leq w(e_{uv}) \quad \forall e_{uv} \in E \\ & r(u) - r(v) \leq \Delta(u, v) \quad \forall u, v \;\exists\; p : u \rightsquigarrow v \end{aligned} \tag{7}$$

The objective function represents the number of latches added to the retimed circuit in relation to the original circuit, taking into account maximal latch sharing at the output of a gate. $\beta(e_{ij})$ is the cost of a latch on edge e_{ij} as defined in Section II-B. The first set of constraints ensures that the weight e_{uv} of each edge (i.e., the number of latches between the output of gate u and the input of gate v) after retiming is non-negative. We will refer to this set as the *circuit constraint* set C_c. The second set of constraints ensures that after retiming, each path satisfies the proper timing constraint of Equation (2). This set, being dependent on the clock period, is referred to as the *period constraint* set C_p.

D. Alternate View of Retiming

In [11] the term *Global Departure Time (GDT)* is defined for each latch, as the latest departure time of data signal from that latch, with reference to the arrival time at the primary inputs, in a global time frame. GDT is an abstract term and can have either positive or negative values. A relation between GDT and retiming, similar to the one between skew and retiming for edge-triggered circuits [3] is also presented in [11]. This relation is used to solve the minperiod retiming problem for level-clocked circuits, by mapping it to the clock skew optimization problem. The GDT's so obtained are then used to get the actual retiming. Moving a latch from the inputs of a gate to its outputs is equivalent to increasing the GDT of that latch by an amount equal to the delay of the gate. Likewise, a motion from outputs to the inputs is equivalent to reducing the GDT by the gate delay. A value of GDT between $-\phi$ and 0 is considered allowable, since it corresponds to zero skew. The GDT's are reduced by relocating latches across gates obtaining the retimed circuit.

III. The Reduced Linear Program

In this section, we will show how reliable bounds on the retiming variables together with other techniques can be used to reduce the size of the LP in Equation (7), in terms of both the number of variables and constraints.

A. Reducing the Variable Set

The work in [4] modified the methods in [3] to obtain bounds on the r variables for edge-triggered circuits. In this work we modify the methods in [11] along the same lines to obtain bounds on the r variables for level-clocked circuits.

Consider the simple circuit in Figure 1 with unit gate delays and a single-phase clocking scheme with 50% duty cycle. We assume that the data signals are available at the primary inputs at the falling edge of the clock, and must arrive at the primary outputs before the falling edge. This assumption gives any signal exactly two clock period to reach the primary outputs from the primary inputs. To achieve the minimum clock period of 2.0 units, one must move the latch $L1$ either to the output of gate $G1$ or to the output of gate $G2$; no other position of the latch $L1$ is feasible. Therefore, it can be seen that the latches cannot be sent to *just any* location in the circuit; rather, there is a restricted range of locations into which a latch may be moved.

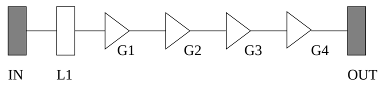

Fig. 1. Example circuit.

As in [4] this concept of restricted mobility of latches is related to the "nearest" and "farthest" location from the primary inputs, that any latch can occupy under the target clock. These locations can be thought of as the boundaries to the range within which the latch may be moved. Here we use the term ASAP (as-soon-as-possible) locations to refer to the locations in which all latches are as close to the primary inputs as possible. Similarly the ALAP (as-late-as-possible) locations has all latches as close to the primary outputs as possible.

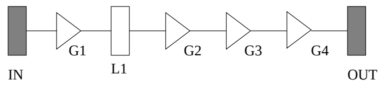

Fig. 2. ASAP latch locations after retiming.

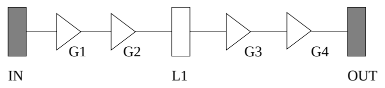

Fig. 3. ALAP latch locations after retiming.

The algorithm for minperiod retiming in [11] is modified along the lines of [4] to give ASAP or ALAP locations. This modified algorithm is run twice on the original circuit, once for ASAP and once for ALAP, to generate the bounds on the r variables. While moving the latches, we count the number of latches that traverse each gate. The count of latches moved across a gate in arriving at the ASAP [ALAP] locations is the upper [lower] bound on the r variable for that gate.

The bounds on the r value of a mirror vertex m_i for the gate i can easily be derived from the bounds on the fanouts of gate i as explained in [4]. Thus we obtain upper and lower bounds on the r variables corresponding to *each* gate y (including mirror vertices) of the form.

$$L_y \leq r(y) \leq U_y \qquad (8)$$

If $U_y = L_y = k_y$ we say that gate y is *fixed* or immobile, otherwise the gate is called *flexible*. These bounds are with respect to a fixed host vertex, i.e., $L_H = U_H = 0$. We can now reduce the variable set V of the LP in Equation (7) to $V' \subseteq V$ the variable set of Minaret-L where

$$V' = \{v \in V | U_v \neq L_v\} \qquad (9)$$

Example: For the circuit in Figure 1, the ASAP location for the latch $L1$ is at the output of gate $G1$

as shown in Figure 2. The number of latches moved across each gate in arriving at this ASAP location, and hence the upper bounds are: $U_{G1} = -1, U_{G2} = 0$, $U_{G3} = 0$, and $U_{G4} = 0$. The ALAP location of latch $L1$ is at the output of gate $G2$, as shown in Figure 3. The number of latches moved across each gate in arriving at this ALAP location, and hence the lower bounds are: $L_{G1} = -1$, $L_{G2} = -1$, $L_{G3} = 0$, and $L_{G4} = 0$.

B. Pruning the Constraint Set

As in [4] a constraint $c(i,j)$ of the type $r(i) - r(j) \leq c_{ij}$ is always satisfied if $U_i - L_j \leq c_{ij}$ is true. This leads us to the following rule

Rule 1: Any constraint (i,j) of the form $r(i) - r(j) \leq c_{ij}$ is redundant in the presence of the bounds and can be dropped if $U_i - L_j \leq c_{ij}$.

Consider three gates a, b and c, such that gate b lies on the path from gate a to gate c. If gate b is a fanin of gate c then we have

$$\begin{aligned} C1: \quad r(a) - r(b) \leq& \quad \Delta(a,b) \\ C2: \quad r(b) - r(c) \leq& \quad w(e_{bc}) \\ C3: \quad r(a) - r(c) \leq& \quad \Delta(a,c) \end{aligned}$$

If $\Delta(a,b) + w(e_{bc}) \leq \Delta(a,c)$ then constraint $C3$ is redundant and can be dropped. This leads us to the following rule

Rule 2: If b and c are two gates reachable from gate a, such that gate b is a fanin of gate c and $\Delta(a,b) + w(e_{bc}) \leq \Delta(a,c)$ then the period constraint from gate a to gate c is redundant and can be dropped.

If gate b is a fanout of gate a then we have

$$\begin{aligned} C4: \quad r(a) - r(b) \leq& \quad w(e_{ab}) \\ C5: \quad r(b) - r(c) \leq& \quad \Delta(b,c) \\ C6: \quad r(a) - r(c) \leq& \quad \Delta(a,c) \end{aligned}$$

If $w(e_{ab}) + \Delta(b,c) \leq \Delta(a,c)$ then constraint $C6$ is redundant and can be dropped. This leads us to the following rule.

Rule 3: If gate b is a fanout of gate a and gate c is some gate reachable from gate a, then if $w(e_{ab}) + \Delta(b,c) \leq \Delta(a,c)$ then the period constraint from gate a to gate c is redundant and can be dropped.

Rule 1 prunes the constraints because the information in the bounds on r variables makes some constraints redundant. Rule 2 and Rule 3 on the other hand prune the constraints because of the discrete nature of the Δ values; these rules can be generalized to include implication by more than two constraints; these generalized rules will, however, be computationally expensive to apply. Rules 2 and 3 on the other hand, can be efficiently applied as explained in Section IV.

Together these rules can reduce the circuit constraint set (C_c) to obtain the reduced circuit constraint set $C'_c \subset C_c$, and the period circuit constraint set (C_p) to get the reduced period constraint set $C'_p \subseteq C_p$. This gives us the reduced LP of Minaret-L as follows:

$$\min \sum_{v \in V'} \left[\left(\sum_{\forall j \in FI(v)} \beta(e_{jv}) - \sum_{\forall j \in FO(v)} \beta(e_{vj}) \right) \cdot r(v) \right] \tag{10}$$

$$\begin{aligned} \text{subject to} \quad r(u) - r(v) \leq w(e_{uv}) & \quad \forall (u,v) \in C'_c \\ r(u) - r(v) \leq \Delta(u,v) & \quad \forall (u,v) \in C'_p \\ L_u \leq r(u) \leq U_u & \quad \forall u \in V' \end{aligned}$$

IV. Generating the Reduced Linear Program

The reduced circuit constraint set C'_c is easily obtained by dropping redundant constraints using Rule 1. A major portion of the computational effort in retiming a level-clocked circuit for minimum area is spent in generating the period constraints set C_p. We now describe efficient techniques for generating the reduced period constraint set C'_p.

A. Computing the δ Values

The generation of period constraints requires computation of $\delta(u,v)$ for all-pairs of gates in the circuit. These δ values can be obtained by re-weighting each edge e_{ij} with $w'(e_{ij}) = \left\lceil w(e_{ij}) - \frac{d(i)}{\pi} \right\rceil$ and computing all-pair shortest paths. We use Johnson's algorithm [13] which has $O(|V|)$ memory requirement, since $O(|V|^2)$ memory is not practical for large circuits with tens of thousand of gates. Johnson's algorithm first re-weights all edges to ensure nonnegative edge weights. The shortest paths between all pair of gates are then be computed by running Dijkstra's algorithm for each gate as source.

Let us consider a particular run of Dijkstra's algorithm with gate a as the source, and let b be a gate to which the shortest path $\delta(a,b)$ has been obtained. Let c be any other gate in the circuit, reachable from gate b.

$$\begin{aligned} \text{By definition,} \quad r(a) - r(b) \leq& \quad U_a - L_b \\ \text{If} \quad U_a - L_b \quad \leq& \quad \delta(a,b), \\ \text{then} \quad r(a) - r(b) \leq& \quad \delta(a,b). \end{aligned} \tag{11}$$

From Equation (3) and Equation (4)

$$r(b) - r(c) \leq \delta(b,c) + \frac{\phi}{\pi} + 1,$$

which when combined with Equation (11) gives

$$r(a) - r(c) \leq \delta(a,b) + \delta(b,c) + \frac{\phi}{\pi} + 1 \tag{12}$$

If the shortest path from gate a to gate c does not go through gate b then $\delta(a,b)+\delta(b,c) \geq \delta(a,c)$ and we do not need to process the fanouts of gate b to obtain $\delta(a,c)$. On the other hand, if the shortest path from gate a to gate c is indeed through gate b then $\delta(a,b)+\delta(b,c) = \delta(a,c)$ and Equation (12) is same as the period constraint $r(a)-r(c) \leq \Delta(a,c)$. If $U_a - L_b \leq \delta(a,b)$ then this period constraint is redundant. In either case we need not process the fanouts of gate b. Since this is true for any c, reachable from gate b, we get the following rule.

Rule 4: If during the shortest path calculations from source a using the Dijkstra's algorithm, for any gate b we have $U_a - L_b \leq \delta(a,b)$, we do not need to process the fanouts of gate b.

Notice that unlike TIM, our approach does not compute the full (all-pairs) shortest path matrix, because of the pruning provided by Rule 4. Since we generate the period constraints from one gate (say gate a) at a time, both $\Delta(a,b)$ and $\Delta(a,c)$ are available in the same iteration, and Rule 2 can be efficiently applied to drop redundant period constraints as they are generated.

B. Reusing δ Computations

We now describe how to reuse some of the computations performed in obtaining the δ values to further speed up the generation of period constraints. The idea is motivated by the fact that in many practical circuits (e.g, ISCAS-89) a high percentage of gates are single-fanout gates. Consider one such gate a with the single fanout b. For gate a, the shortest paths to all other gates must be via gate b, which implies that $\delta(a,c) = w'(e_{a,b}) + \delta(b,c)$. Therefore we can obtain the shortest paths from gate a by simply adding $w'(e_{a,b})$ to the shortest paths from gate b. Thus if we somehow ensure that shortest paths from gate b are obtained before those from gate a, we will save one complete execution of Dijkstra's algorithm (for gate a as source). We call this approach "chaining" and the set of gates for which only one δ computations is performed as "chains." We use Dijkstra's algorithm to obtain the δ values for the gate at the head of a chain, and then simple additions give the δ values for all of the other gates in the chain. This chaining technique can be generalized to store δ values from multiple gates. We found that by using a simple chaining technique, with no storage overhead we could reduce the time spent in generating the period constraints on average by about 50%.

These chains are obtained by a fast preprocessing step which maps each gate to exactly one chain. As a side note, Rule 4 must be modified for use with chaining to ensure that it holds for all gates that reuse δ computations. This reuse of δ computations also enables efficient application of Rule 3. This is possible because for a gate a with only one fanout (gate b), $\Delta(a,c)$ is derived from $\Delta(b,c)$, and hence both are available when the period constraint from a to c is being generated.

V. Solving the Linear Program

Like Equation (7), the LP in Equation (10) is also the dual of a min-cost flow problem [2]. We found that it could be solved very efficiently using the network simplex algorithm from [14]. The network simplex method is a graph based adaptation of the LP simplex method that 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 [14].

Using the first eligible arc pivot rule with a wraparound arc list from [15, page 417] gave us significant improvements in the run time. The r variables (duals of the flow variables) are directly available from the min-cost flow solution. Using these techniques we could solve min-cost flow problems with 70,000 variables and more than 17 million constraints in under 9 minutes.

VI. Experimental Results

We present results for the larger circuits in ISCAS-89 benchmark suite, and some other large circuits (myex1 through myex5) created by combining ISCAS-89 circuits. As in [9], to obtain level-clocked circuits, we replaced each edge-triggered FF in the ISCAS-89 circuit by two level-sensitive latches.

In Table I we show for each circuit, the number of gates $|G|$, the number of latches, the problem size, and the execution times. The circuits are retimed for the minimum possible period, and the number of latches for both minperiod retiming [11] and minarea retiming by Minaret-L (for the same clock period) obtained after taking into account the maximum register sharing [2] are shown. For almost all circuits minarea retiming reduces the number of latches in the circuit by a factor of two to three as compared to minperiod retiming, even though both retime the circuit for the same clock period. This underscores the importance of minarea retiming.

We compare the size of the LP in Minaret-L given in Equation (10) and the original LP in Equation (7) by presenting the number of variables and constraints in both. It can be seen that up to three orders of magnitude reduction is obtained in the number of constraints by using Minaret-L. The number of unpruned constraints grow at the rate of $O(|G|^2)$ and our pruning techniques reduce this rate of growth significantly.

Also presented are the CPU time required to compute the bounds on the r variables (T_b), generate the LP (T_p), and to solve it (T_s). All execution times are in seconds on a DEC AXP system 3000/900 workstation with 256M RAM. It can be seen that even though the reduction in size of the LP due to the bounds is significant, the extra time spent in obtaining them is an insignificant fraction of the total CPU time for minarea retiming (T_{total}). The time required to generate the LP dominates the total execution time signifying the importance of chaining and Rule 4. The small execution time of Minaret-L highlights the effectiveness of our efficiency enhancing techniques.

VII. Conclusion

A fast algorithm for minarea retiming of large level-clocked circuits has been presented. The entire ISCAS-89 benchmark suite could be retimed in minutes. This work unifies the two approaches to retiming of level-clocked circuits, i.e., the TIM [9] and the ASTRA [11] approaches. This unification together with the other techniques (Rule 2, Rule 3, Rule 4 and chaining) leads to an efficient method of generating a much smaller LP, with two to three orders of magnitude less constraints. It makes it feasible to retime large level-clocked circuits (over 56,000 gates) for minimum area in very reasonable time (under 1.5 hours). To put this in perspective, the largest level-clocked circuit for which minarea retiming results had been published in the past had less than 400 gates [9].

Minaret-L can retime large level-clocked circuit for minimum area in time similar to that required for retiming edge-triggered circuits, thus bringing the techniques for retiming level-clocked circuits at par with the state of art in retiming techniques for edge-triggered 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] 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.

[4] N. Maheshwari and S. S. Sapatnekar, “An improved algorithm for minimum-area retiming,” in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 2–7, 1997.

[5] 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.

[6] S. Dey and S. Chakradhar, “Retiming sequential circuits to enhance testability,” in *Proceedings of the IEEE VLSI Test Symposium*, pp. 28–33, 1994.

[7] N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, “Retiming of circuits with single phase transparent latches,” in *Proceedings of the IEEE International Conference on Computer Design*, pp. 86–89, 1991.

[8] 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.

[9] 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.

[10] 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.

[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] 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.

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

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

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

TABLE I
Minimum Area Retiming using Minaret-L

Circuit	$\|G\|$	# Latches		# Variables		# Constraints		T_b	T_p	T_s	T_{total}
		Minperiod	Minarea	Minaret-L	Original	Minaret-L	Original	(sec)	(sec)	(sec)	(sec)
s3384	1,754	685	322	1,961	2,166	54,216	761,365	0.19	1.95	0.57	2.71
s4863	2,407	588	237	2,693	2,995	83,752	5,482,144	0.15	4.96	0.46	5.57
s5378	2,863	660	314	2,960	3,664	31,497	4,595,422	0.15	2.93	0.25	3.32
s6669	3,218	1,168	530	3,787	4,100	59,372	1,924,119	0.24	5.99	0.46	6.69
s9234.1	3,337	590	260	3,543	3,893	136,675	6,443,183	0.24	12.40	0.72	13.36
s13207.1	8,004	2,831	890	7,656	9,180	73,796	22,908,612	1.05	19.29	1.36	21.70
s15850.1	9,843	1,718	1,033	9,013	11,332	93,423	39,493,359	1.42	49.27	2.90	53.60
s35932	16,420	4,629	3,457	17,144	21,716	110,019	129,482,466	1.85	42.93	4.73	49.51
s38584.1	19,595	7,399	2,852	20,590	23,390	161,638	293,482,466	3.55	118.21	10.48	132.24
s38417	21,504	6,861	2,722	25,735	25,923	4,390,018	149,498,368	4.93	350.44	147.60	502.97
myex1	26,113	8,503	4,559	32,509	32,922	4,550,735	313,110,465	5.59	484.69	61.01	551.29
myex2	29,134	9,288	3,883	30,489	34,417	213,914	504,058,462	7.16	394.72	20.25	422.13
myex3	35,711	13,309	6,370	43,837	44,812	11,325,336	725,691,302	8.16	1613.56	187.48	1809.20
myex4	40,993	15,087	5,531	48,708	49,214	7,400,927	819,727,160	10.39	1926.41	234.11	2170.92
myex5	56,895	19,307	9,039	70,000	70,414	17,269,129	1,624,722,880	16.84	4623.62	535.62	5176.07