A Fresh Look at Retiming via Clock Skew Optimization

Rahul B. Deokar and Sachin S. Sapatnekar

Department of Electrical and Computer Engineering
201 Coover Hall, Iowa State University, Ames, IA 50011.

Abstract

The introduction of clock skew at an edge-triggered flip-flop has an effect that is similar to the movement of the flip-flop across combinational logic module boundaries, and these are continuous and discrete optimizations with the same effect. While this fact has been recognized before, this work, for the first time, utilizes this information to find a minimum/specified period retiming efficiently. The clock period is guaranteed to be at most one gate delay larger than a tight lower bound on the optimal clock period; this bound is achievable using a combination of intentional skew and retiming. All ISCAS89 circuits can be retimed in a few minutes by this algorithm.

I Introduction

The importance of the issue of optimizing the timing behavior of VLSI circuits probably needs no introduction to any reader of this paper. This paper considers one method of timing optimization, retiming [1], which proceeds by relocating flip-flops (FF's) within a network to achieve faster clocking speeds. A novel approach to retiming that utilizes the solution of the clock skew optimization problem [2] forms the backbone of this work.

The introduction of clock skew at an FF has an effect that is similar to the movement of the FF across combinational logic module boundaries. This was observed (but not proved) in [2], which stated that clock skew and retiming are continuous and discrete optimizations with the same effect. Although the designer can choose between the two transformations, these methods can, in general, complement each other. The equivalence between retiming and skew has been observed and used in earlier work, e.g., in [3–5]. The contribution of this work is that it exploits this equivalence and presents a method that finds an optimal retiming efficiently, with a clock period that is guaranteed to be *at most* one gate delay larger than the optimal clock period. The method views the circuit hierarchically, first solving the clock skew problem at one level above the gate level, and then using local transformations at the gate level to perform retiming for the optimal clock period.

32nd ACM/IEEE Design Automation Conference ®

The clock skew problem is first solved using efficient graph-theoretic techniques [6]. The idea of using graph algorithms is to take advantage of the structure of the problem to arrive at an efficient solution. Like [2], our technique is illustrated on single-phase clocked circuits containing edge-triggered FF's. The advantage of using this graph algorithm is that it not only minimizes the clock period, but that unlike a simplex-based linear programming approach, it also ensures that the difference between the maximum and minimum skews is minimized at the optimal clock period, which may reduce the amount of work required in the relocation phase.

The complexity of solving the retiming problem using the technique in [1] is $O(|G|^2 \log |G|)$, where $|G|$ is the number of gates in the circuit; this could be phenomenally large; an efficient implementation has been reported very recently [7]. Our method has similar run-times, and solves the more general problem of retiming with skew optimization.

In our algorithm, named ASTRA[1] (A Skew-To-Retiming Algorithm), the gates in the circuit are assumed to have constant delays. The solution is divided into two phases. In Phase A, the clock skew optimization problem is solved with the objective of minimizing the clock period and ensuring that the difference between the maximum and the minimum skew is minimized. This difference provides an indication of how many gates have to be traversed in the next phase, and therefore, it is important that it be small. In Phase B, the skew solution is translated to retiming and some FF's are relocated across gates to reset the values of all skews to be as close to zero as possible. The designer may choose to achieve the optimal clock period by using a combination of clock skew and retiming; alternatively, any skews that could not be set exactly to zero may now be forced to zero. This could cause the clock period to increase; however, it is shown that this increase will be no greater than one gate delay.

The equivalence between retiming and clock skew is now shown in Sections II and III. The algorithm for the clock skew optimization phase is described in Section IV. Next, in Section V, the process of finding a retiming solution is described. Finally, we present experimental results in Section VI and conclude the paper in Section VII.

II Clock Skew Optimization 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.

[1] "astra" (अस्त्र): (Sanskrit) a sophisticated weapon.

Thus, there is a *skew* between the clock arrival times at different FF's. Several methods for achieving zero clock skew have been developed, e.g., [8].

Viewing clock skews as a manageable resource rather than a liability allows a designer to introduce intentional skews to improve the performance of the circuit. To illustrate this, consider the example in Figure 1, where each inverter has a unit delay. The combinational circuit blocks CC_1 and CC_2 have delays of 3.0 and 1.0 units, respectively; 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 [2].

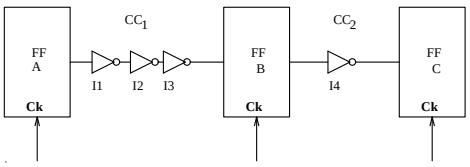

Figure 1: The advantages of nonzero clock skew.

A second approach to timing optimization is the procedure of retiming [1]. Retiming involves the relocation of FF's across logic gates to allow the circuit to be operated under a faster clock, without changing its functionality. For the circuit in Figure 1, the period of the original circuit can be minimized to 2.0 units by moving 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.

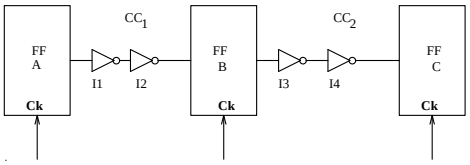

Figure 2: Retiming for clock period optimization.

If one were to imagine the circuit as being drawn with its inputs to the left and outputs to the right, then the conversion of a negative (positive) skew to zero skew would involve the relocation of FF's to the right (left). In this paper, we will use the terms "right" and "left" to denote the direction of signal propagation and the direction opposite to that of signal propagation, respectively.

III Equivalence between Clock Skew and Retiming

A more formal presentation of the equivalence between clock skew and retiming is presented here.

Theorem 1 [9] For a circuit that operates at a clock period P, satisfying long-path and short-path [2] delay constraints,

(a) retiming an FF by moving it to the left across a gate of delay d_1 is equivalent to decreasing its skew by d_1.

(b) retiming an FF by moving it to the right across a gate of delay d_2 is equivalent to increasing its skew by d_2.

Therefore, if one were to calculate the optimal clock skews corresponding to the minimum clock period, one could retime the circuit by moving FF's with positive(negative) skews to the left(right) until the skews at the FF's are nearly equal to zero. It must be noted that since gate delays take on discrete values, it is not possible to guarantee that the skew at an FF can always be reduced to zero through retiming operations.

An alternative view of the same procedure is as follows. Retiming may be thought of as a sequence of movements of FF's across gates. We may start from the final retimed circuit, where all of the skews are zero, and the zero-clocking and double-clocking constraints are met, and perform the sequence of movements in reverse order. This procedure can be used to move all FF's back to their initial locations, using Theorem 1 to keep track of the changed clock skews at each FF. Therefore, the optimal retiming is equivalent to applying skews at the inputs of FF's.

Note that the optimal clock period provided by the clock skew optimization procedure must, therefore, be no greater than the clock period for the set of clock skews thus obtained. Any differences arise due to the fact that clock skew optimization is a continuous optimization, while retiming is a discrete optimization.

Corollary 2 : The clock period obtained by an optimal retiming can be achieved via clock skew optimization. The clock period provided by the clock skew optimization procedure is less than or equal to that provided by the method of retiming.

IV Phase A: Optimizing Clock Skews

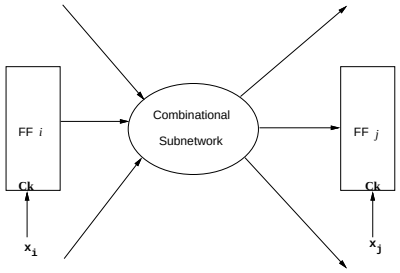

Figure 3: The clock skew optimization problem.

A Problem formulation

Only the barest essentials of this phase are described here; most of the ideas are similar to those published in [6]. Given a combinational circuit segment that lies between two FF's i and j, as shown in Figure 3, if x_i and x_j are the clock arrival times at the two FF's, then the following inequation must be satisfied:

$$x_i + \overline{d}(i,j) + T_{setup} \quad \leq \quad x_j + P \qquad (1)$$

where $\overline{d}(i,j)$ is the maximum delay of any combinational path between FF's i and j. The clock skew problem for

minimizing the clock period is solved via the following linear program:

$$\begin{array}{lll} \text{minimize} & P & \\ \text{subject to} & x_j - x_i + P & \geq T_{setup} + \overline{d}(i,j) \end{array} \quad (2)$$

for every pair, (i,j) of FF's such that there is at least one combinational path from FF i to FF j.

In this work, we will ignore the short path constraints. We thus obtain the clock skews that correspond to the minimum clock period. The short path and logic signal separation constraint violations [10] can be reconciled by using an algorithm for minimum padding [11].

Notice that for a constant value of P, the constraint matrix for the linear program in (2) reduces to a system of difference constraints [12]. A feasible solution to the linear program exists iff the corresponding constraint graph $G(P)$ contains no positive cycles. It is possible to derive the following upper and lower bounds on the optimal clock period, P_{opt} [6]:

$$P_{low} = \min_{e \in G(P)} w(e) \leq P_{opt} \leq \max_{e \in G(P)} w(e) = P_{high} \quad (3)$$

where $w(e)$ is the weight of a constraint graph edge. An even better lower bound can be obtained by additionally recognizing that the weight of any edge that begins and ends at the same node constrains the optimal clock period from below.

For any input i, the procedure for computing $\overline{d}(i,j)$ for all j involves setting the arrival time at input i to zero, and that at all other inputs to $-\infty$ [2]. The resulting signal arrival time at each output j, found using PERT [13], is the value of $\overline{d}(i,j)$. The procedure can be made more efficient by levelizing the circuit once, and propagating the flow of PERT only along the excited paths; details are omitted due to space constraints.

B The clock skew optimization procedure

The skeletal pseudo-code describing the algorithm for finding the optimal clock period proceeds as shown below. The theory behind this is described in [6].

```
Construct the constraint graph;
P_max = P_high;
P_min = P_low;
while (P_max − P_min) > ε {
    P = (P_max + P_min)/2;
    if G(P) has a positive cycle
        P_min = P;
    else
        P_max = P;
}
```

In the above algorithm, the presence of a positive cycle in $G(P)$ may be tested using the Bellman-Ford algorithm [12]. If the skews are initialized to 0, the Bellman-Ford solution minimizes $|x_{i,max} - x_{i,min}|$. On a graph with V vertices and E edges, the computational complexity of this algorithm is $O(V \cdot E)$. The number of iterations is $(P_{high} - P_{low})/\epsilon$. Therefore, the iterative procedure above, when carried to convergence, provides the solution to the linear program (2) in time

$$O\left(|F| \cdot E \cdot log_2\left[(P_{high} - P_{low})/\epsilon\right]\right) \sim O(|F| \cdot E) \quad (4)$$

where $|F|$ is the number of FF's in the circuit, E is the number of pairs of FF's connected by a combinational path, P_{high} and P_{low} are as defined in (3), and ϵ, defined in the pseudo-code above, corresponds to the degree of accuracy required. For real circuits, $E = O(|F|)$; therefore, for a fixed accuracy, the complexity of the procedure is $O(|F|^2)$. The efficient use of back-pointers [7] can provide further improvements in practice, so that these complexity formulae are not very meaningful.

In the solution found above, all skews must necessarily be positive, since the weights of each node in the Bellman-Ford algorithm was initialized to zero. Also, in general, the skew at the host node (corresponding to primary inputs and outputs) could be nonzero. Our objective is to ensure a zero skew at the primary input and output nodes since we do not have the flexibility of retiming these, and hence we modify this solution. Note that if $[x_1, \cdots x_n]$ is a solution to a system of difference constraints in $\mathbf{x}$, then so is $\mathbf{x}' = [(x_1 + k), (x_2 + k), \cdots, (x_n + k)]$. Therefore, by selecting k to be the negative of the skew at the host node, a solution $\mathbf{x}'$ with a zero skew at the host is found.

V Phase B: Skew Minimization by Retiming

In Phase B, the magnitudes of the clock skews obtained from Phase A are reduced to zero by relocating FF's across logic gates, while maintaining the optimal clock period. After the skew magnitudes have been reduced by as much as possible, the retimed circuit may be implemented by applying the requisite skews at each FF (to get the minimum achievable clock period) or by setting all skews to zero (to get a clock period that is, as will be shown in Section VI, no more than one gate delay above the optimum).

Since only FF's with nonzero skews are moved, we divide the relocations into the two following categories:
(a) Flip-flops with negative skews
(b) Flip-flops with positive skews
We describe the algorithm for the former case; the latter is similar and is described in detail in [9].

A Skew reduction

Consider the case of an FF j shown in the Figure 4(a) that has a negative skew at the conclusion of Phase A.[2] Through retiming operations, it is possible to transform the circuit in Figure 4(a) to the one in Figure 4(b); the equivalent skews at each FF in Figure 4(b) are calculated using a procedure that will be described later. At this point, it need only be noted that the equivalent skews for these FF's may be found without physically moving them to the gate inputs. The fact that this may be done is shown by the following theorem:

[2] We assume here that each FF fans out to exactly one gate. If the fanout of an FF is larger than one, then it is replicated at each fanout branch. The replicated FF's have exactly one fanout gate, and each such FF is considered in turn.

Theorem 3 [9] (a) Retiming transformations may be used to move flip-flops from all of the inputs of any combinational block to all of its outputs. The equivalent skew of the relocated flip-flop at output j, considering long path constraints only, is given by $x_j = \max_{1 \leq i \leq n}(x_i + \overline{d}(i,j))$ where the $x_i, 1 \leq i \leq n$ are the skews at the input flip-flops, and x_j is the equivalent skew at output j, and $\overline{d}(i,j)$ is the worst-case delay of any path from i to j.
(b) Similarly, flip-flops may be moved from all of the outputs of any combinational block to all of its inputs, and the equivalent skew at input k, considering long path constraints only, is given by $x_k = \min_{1 \leq j \leq m}(x_j - \overline{d}(k,j))$ where the $x_j, 1 \leq j \leq m$ are the skews at the input flip-flops, and x_k is the equivalent skew at input k, and $\overline{d}(k,j)$ is the worst-case delay of any path from k to j.

There now exists a set of n "virtual" FF's at the input to gate p, with skews $s_1, s_2, \cdots, s_n$, which must satisfy the constraints:

$$s_k + delay \quad \leq \quad s_i + P \ \forall \ 1 \leq k \leq n \tag{5}$$

where P is the clock period, s_i the skew at an output FF, FF_i, of the combinational block to which $FF_1 \cdots FF_n$ are input FF's, and $delay$ is the largest combinational delay from the input of gate p to FF_i.

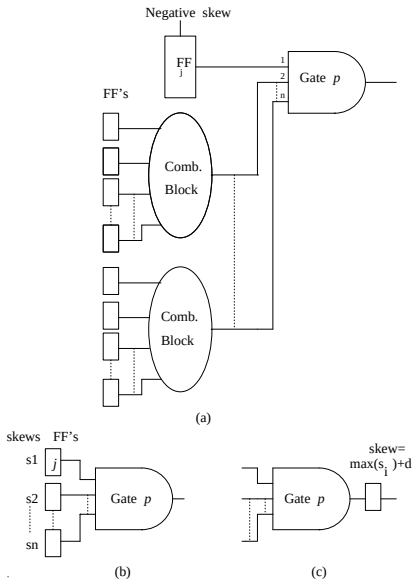

Figure 4: Retiming for a negative skew FF.

Obviously, from the above constraints:

$$\max_{1 \leq k \leq n}(s_k) + delay \leq s_i + P \tag{6}$$

We may now have one of two scenarios:

(1) All of the n FF's at the inputs have negative skews. In this case, the maximum of all the negative skew FF's is negative and hence the set of FF's may be moved across the gate p, as shown in Figure 4(c). If the sign of the skew were to change after the relocation, the relocation would be carried out only if it reduced the magnitude of the skew. If not, all FF's would be left in their current locations, and the skew of FF_j would be set to $\max_{1 \leq i \leq n}(s_i)$. For example, an FF with a skew of -0.75, to be moved across a gate with unit delay, would have a new skew value of 0.25; such a relocation would be desirable. However, it would be undesirable to move an FF with skew -0.25 across a unit delay gate. Therefore, in either case, this effects a reduction in the magnitudes of the skew values, as is desirable.

(2) One or more of the virtual FF's has a positive effective skew. In this case, the skew at the FF j under consideration may be set to zero without violating any timing constraints, i.e., the maximum skew at the input to gate p would be unchanged by this.

B Minimization procedure

The steps involved in minimizing the skew magnitudes for FF's with negative skews are outlined below:
Step 1 All FF's in the circuit with negative skews are placed on a queue, Q. For each FF, we consider one fanout at a time, using the transformation in footnote 2.
Step 2 Let j be the FF that is currently at the head of the queue, and p the gate that it fans out to. We will assume for now that p is a gate; the case where it is the input of an FF will be dealt with separately. The equivalent skew at every other fanin node of p is found. These FF's are not actually moved to the gate input at this time, and are hence referred to as virtual FF's.

The equivalent skews are found as follows. Consider an FF j with negative skew, as shown in Figure 5. The gate p to which it fans out to is added to the tail of a queue R.[3] A backward PERT is employed to trace back along the fanin cone of gate p. When a gate is encountered, it is added to the queue. In Figure 5, gate z is first added to R, and in the next step, gate y is added. The trace-back continues until an FF is encountered. In the example in Figure 5, the trace-back terminates when FF x is encountered. During this process, we keep track of the worst-case delay, d, to gate p. As a consequence of Theorem 3, if the skew (calculated in Phase A) at FF x is t units, then its equivalent skew at the input to gate p is $t + d$ units.
Step 3 If any equivalent skew at a virtual FF is positive, then the skew at j is set to zero and it is not retimed; if not, the skew after retiming is found using the criteria described earlier. If the magnitude of this skew is smaller than the current skew at j, then j and all of the virtual FF's at the input to p are retimed across p. (Notice that if the skew changes sign after retiming, then the magnitude of the retimed skew could become larger.) Note that the motion of the virtual FF's to their new location may entail replicating these FF's. For example, if FF j were to be moved across gate p, a new FF would have to be created at y_2 with an equivalent skew corresponding to the skew of FF x, retimed to position y_2. The new skews are found

[3] Note that the queue R is distinct from the queue Q.

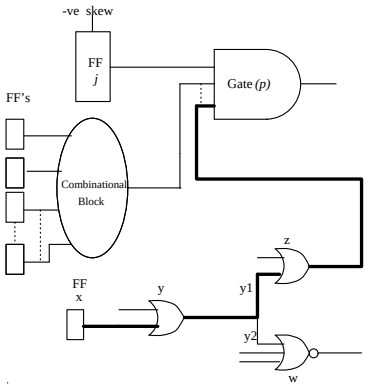

Figure 5: Reproduction of FF's during the backtrace.

as explained in the previous section. Any such FF's that have a negative skew (as will happen most of the time, unless relocation changed the sign of the skew) are now placed at the tail of the queue, Q, and are processed later.
Step 4 If the retimed FF has a negative skew, it is placed at the tail of Q.
Step 5 If Q is not empty, go to Step 1; if not, all negative skew FF's have been processed.

In Step 2 above, if the FF j fans out to another FF, which we shall call k, then since there is no combinational delay between the FF's, and retiming preserves the zero-clocking constraints, it must be true that

$$\begin{aligned} x_j + T_{setup} &\leq x_k + P \\ \text{i.e.,} \quad x_j &\leq x_k + P - T_{setup} \end{aligned}$$

If the right hand side is positive, then x_j can be set to zero without violating any constraints. If not, then $x_k \leq T_{setup} - P < 0$, which implies that k is an FF that will eventually move to the right, thereby allowing FF j the leeway to move as well. Therefore, if this is the case, the skew of FF j is set to

$$x_j = x_k + P - T_{setup} < 0 \tag{7}$$

and the FF j is reprocessed after FF k has been processed (i.e., its skew is set to the value calculated above, and it is placed at the tail of Q). All such FF's *will* eventually be processed, and their skews set to nearly zero. It is interesting to note that the latter case was rarely seen to happen in the ISCAS89 benchmarks.

VI Properties of the Retiming Procedure

Lemma 4 [9] At the end of the retiming procedure in Phase B, the skew at each FF is no more than half a gate delay.

Theorem 5 [9] If, at the end of the retiming procedure, all skews are set to zero, then the optimal clock period for this circuit is no more than $P_{opt} + d_{max}$, where P_{opt} is the optimal clock period found in Phase A, and d_{max} is the maximum delay of any gate in the circuit.

VII Experimental Results

The ASTRA algorithm was implemented as a 3000-line C program, and experimental results running from ASTRA on circuits in the ISCAS89 benchmark suite are presented in Table 1. It is interesting to note here that the versions of these circuits that we obtained from `mcnc.org` are different from those in [7]; we find that for the same circuit, our version has about twice as many gates as those in [7].

For each circuit, the table provides data that describes its size in terms of the number of Gates, $|G|$. All gates are assumed to have unit delays, and the setup and hold times are arbitrarily set to 0. P_{high} is the upper bound on the clock period provided by (3); note that it corresponds to the clock period in the original circuit. The next column shows change from P_{high} to the optimal value, P_{opt}, calculated at the end of Phase A of ASTRA, using clock skew optimization. At the end of retiming in Phase B of ASTRA, any skews that could not be set exactly to zero are now forced to zero and the new period P_{ret} is shown, along with the corresponding percentage improvement over the initial period. Note that as claimed, P_{ret} is always within one gate delay of P_{opt}. The total CPU times, T, for running ASTRA on an HP 735 is shown for all these circuits.[4] The last column shows the initial and final number of FF's in the circuit. For example, for s38584, a circuit with 1426

Table 1: Results of Clock Skew Optimization

Circuit	$\|G\|$	$P_{high} \rightarrow P_{ret}(P_{opt})$	Impr. (%)	T	$\|F\|_i \rightarrow \|F\|_f$
s1488	656	17.0 → 16.0(16.0)	6%	0s	6 → 17
s1494	653	17.0 → 16.0(16.0)	6%	0s	6 → 20
s3271	1572	28.0 → 14.7(15.0)	87%	2s	116 → 428
s3330	1789	29.0 → 14.0(14.0)	107%	3s	132 → 330
s3384	1685	60.0 → 26.5(27.0)	122%	19s	183 → 1697
s4863	2342	58.0 → 30.0(30.0)	93%	2s	104 → 241
s5378	2779	25.0 → 21.0(21.0)	19%	9s	179 → 553
s6669	3080	93.0 → 29.0(29.0)	221%	50s	239 → 2585
s9234	5597	58.0 → 38.0(38.0)	53%	10s	211 → 359
s13207	7951	59.0 → 51.0(51.0)	16%	2s	638 → 640
s15850	9772	82.0 → 63.0(63.0)	30%	7s	534 → 572
s35932	16065	29.0 → 27.0(27.0)	7%	24s	1728 → 1729
s38417	22179	47.0 → 32.0(31.5)	47%	77s	1636 → 1691
s38584	19253	56.0 → 48.0(48.0)	17%	5s	1426 → 1428

FF's, the value of P_{high} is found to be 56.0 units. At the end of Phase A (skew optimization), ASTRA calculates the value of P_{opt} to be 48.0 units, which is the same as the value of P_{ret} at the end of Phase B. The improvement in the clock period after the two phases is calculated as

$$\%change = \frac{P_{high} - P_{ret}}{P_{ret}} \times 100$$

and is found to be 16.7% for s38584. The number of FF's was increased in the process from 1426 to 1428.

[4] At the time of writing this paper, Phase A has been greatly optimized. For example, the total run-time (Phase A + Phase B) for the algorithm for s38417 has been brought down from 1 hour in the first implementation to 77s in the current implementation. Phase B, whose run-time was insignificant in the first implementation, is now often the bottleneck. The code for Phase B is currently not very efficient, and further run-time improvements are possible.

It was observed that in 36 out of 44 ISCAS89 circuits, ASTRA caused the clock period to improve, with the improvement being as much as 221% in the case of s6669. Although it is theoretically possible for retiming to reduce the number of FF's in the circuit, this was never seen to happen. The percentage increase in the number of FF's ranged from 0% to even about 1000% in two cases; it must be stressed, though, that minimizing the number of FF's is *not* incorporated in the objective here and is a direction for future research.

It is worth noting that the CPU times for ASTRA are rather small; even the largest circuit could be retimed in just over a minute. The run-times for ASTRA versus the circuit size ($|G|$) and the corresponding reduction in the clock period are shown in Figure 6.

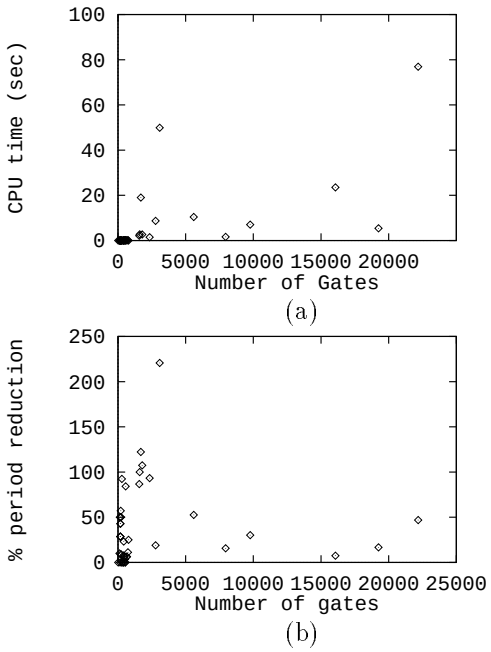

Figure 6: Performance vs. circuit complexity.

VIII Conclusion

An approach that takes advantage of the equivalence between retiming and clock skew is presented, and is used for gate-level retiming. Results on all of the circuits in the ISCAS89 benchmark suite have been presented and can easily be handled by this algorithm.

The chief reason for the improvement is that ASTRA takes a global view of retiming by first solving the clock skew problem in a smaller number of variables. In the second phase, local transformations are used to perform the retiming. The logic behind this approach is that a circuit would have to be very poorly designed indeed to require enormous computation time for the local transformations, and hence in most practical cases, the latter phase takes only a small amount of computation; this is borne out by our experimental results. It must be pointed out that the algorithm performs retiming only for timing optimization and does not take into account the fact that retiming may cause initial states to change. This may be offset by the use of methods such as [14].

The ASTRA algorithm may trivially be adapted to perform retiming to satisfy a given clock period, P_{spec}. Phase A consists of a single pass through the graph $G(P_{spec})$. If P_{spec} is infeasible, this will be reported immediately; else, the skew solution at the end of Phase A may be translated to a retiming solution using the methods of Phase B.

References

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

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

[3] H.-G. Martin, "Retiming by combination of relocation and clock delay adjustment," in *Proceedings of the European Design Automation Conference*, pp. 384–389, 1993.

[4] B. Lockyear and C. Ebeling, "Minimizing the effect of clock skew via circuit retiming," Tech. Rep. UW-CSE-93-05-04, Department of Computer Science and Engineering, University of Washington, Seattle, 1993.

[5] L.-F. Chao and E. H.-M. Sha, "Retiming and clock skew for synchronous systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1.283–1.286, 1994.

[6] R. B. Deokar and S. S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1.407–1.410, 1994.

[7] 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.

[8] R.-S. Tsay, "An exact zero-skew clock routing algorithm," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 242–249, Feb. 1993.

[9] S. S. Sapatnekar and R. B. Deokar, "Utilizing the retiming-skew equivalence in an efficient algorithm for retiming large circuits," Tech. Rep. ISU-CPRE-94-SS06, Iowa State University, Ames, IA, 1994.

[10] D. Joy and M. Ciesielski, "Clock period minimization with wave pipelining," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 461–472, Apr. 1993.

[11] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 156–161, 1993.

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

[13] S. Even, *Graph Algorithms*. Potomac, MD: Computer Science Press, 1979.

[14] H. J. Touati and R. K. Brayton, "Computing the initial states of retimed circuits," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 157–162, Jan. 1993.