A Graph-theoretic Approach to Clock Skew Optimization

Rahul B. Deokar and Sachin S. Sapatnekar
Department of Electrical Engineering and Computer Engineering
Iowa State University
Ames, IA 50011

Abstract

This paper addresses the problem of minimizing the clock period of a circuit by optimizing the clock skews. We incorporate uncertainty factors and present a formulation that ensures that the optimization will be safe. In [1], the problem of clock period optimization is formulated as a linear program. We first propose an efficient graph-based solution that takes advantage of the structure of the problem. We also show that the results of [1] may result in exceedingly large skews, and propose a method to reduce these skews without sacrificing the optimality of the clock period. Experimental results on several ISCAS89 benchmark circuits are provided.

1 Introduction

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 flip-flops (FF's) at the same time. Thus, there is a *skew* between the clock arrival times at different latches.

One approach that has been followed by several researchers is to design the clock distribution network so at to ensure zero clock skew ([2, 3] etc.). An alternative approach views clock skews as a manageable resource rather than a liability, and manipulates clock skews to advantage by intentionally introducing skews to improve the performance of the circuit. To illustrate how this may be done, consider the following example. In Figure 1, if the combinational subcircuits CC_1 and CC_2 have delays of 6 and 14 units, respectively, the fastest allowable clock has a period of 14 units. However, if a skew of -4 units is applied to the clock line to latch B, the circuit can run with a clock period of 10 units. This approach was formalized in the work by Fishburn [1] where clock skew optimization problem was formulated as a linear program (LP).

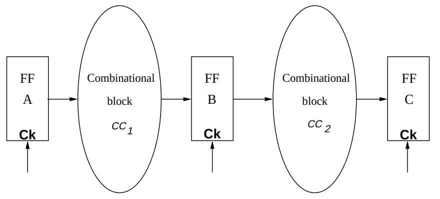

Figure 1: The advantages of nonzero clock skew.

One of the common misconceptions of circuit designers about changing clock skews is that it is an "unsafe" optimization, in that a small change in the gate delays may cause a precariously balanced circuit to malfunction. In fact, this is not so; one can build in safety margins, δ, that ensure that skewing errors of $\pm\delta/2$ $(\delta > 0)$ do not disrupt circuit functionality.

In this work, we show that the clock skew problem may be solved using efficient graph-theoretic techniques in polynomial time. The idea of using graph methods is to take advantage of the structure of the problem to arrive at an efficient solution. Like [1], we illustrate our technique on single-phase clocked circuits containing edge-triggered FF's. The problem is solved in two phases. In Phase I, we solve a small variation of the problem formulated in [1] to find the optimal clock period. The result of this optimization may lead to inordinately large skews. Hence, in Phase II, we formulate a second problem that minimizes the skews for the given clock period; note that this minimization is not carried out in [1]. Thus, not only does the algorithm minimize the clock period, but it also ensures that the magnitudes of the skews required to run the circuit at the optimal period are minimized.

2 Phase I: Clock Period Optimization

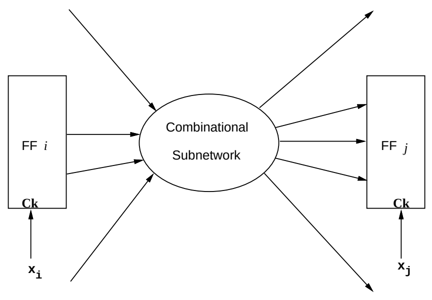

Figure 2: The clock skew optimization problem.

Given a combinational circuit segment that lies between two FF's i and j, as shown in Figure 2, let x_i and x_j are the clock arrival times at the two FF's. We denote the maximum (minimum) combinational delay between FF's i and j by $\overline{d}(i,j)$ ($\underline{d}(i,j)$), the clock period by P, and the FF setup (hold) time by T_{setup} (T_{hold}). $\overline{d}(i,j)$ and $\underline{d}(i,j)$ may be efficiently calculated using the modified PERT method proposed in [1].

To model the uncertainty in the skews, suppose one can guarantee that in the manufactured circuit, the skew at FF k will be within the range $[x_k - \delta/2, x_k + \delta/2]$, where x_k is the designed value of the skew. In that case, the difference between any skews, $(x_i - x_j)$ in the manufactured circuit, must be within δ of the designed value of $(x_i - x_j)$. We formulate the following LP that is guaranteed to produce a functional manufactured circuit:

$$\begin{array}{lll} \text{minimize} & P & \\ \text{subject to} & |x_k| & < P \quad \forall \text{ FF's } k \text{ in the circuit} \\ & x_i - x_j & \geq T_{hold} - \underline{d}(i,j) + \delta \\ & x_j - x_i & \geq T_{setup} + \overline{d}(i,j) - P + \delta \end{array} \quad (1)$$

for every pair, (i,j) of FF's such that there is at least one combinational path from FF i to FF j. This formulation is similar to that in [1], except that [1] omits the $|x_k| < P$ constraints, and that we introduce the safety factor, δ. This LP has a constraint matrix with a special structure, which we take advantage of to solve the problem using graph-theoretic techniques.

If the value of P were constant, the constraint matrix of the above LP would reduce to a system of difference constraints [4]. When $\overline{d}(i,j)$ and $\underline{d}(i,j))$ are finite, a pair of directed edges are between x_i and x_j is constructed in $G_1(P)$. Note that the weight of one of the edges in the pair is dependent on P. A feasible solution to the LP exists if the corresponding constraint graph $G_1(P)$ contains no positive cycles.

The following theoretical results are stated without proof due to space limitations:

Lemma 1: If $P = P_1$ does not permit a feasible solution to (1), then nor does $P = P_2 < P_1$.

Corollary 2: If $P = P_1$ permits a feasible solution to (1), then so does $P = P_2 > P_1$.

Lemma 3: Given a pair of inequalities for each pair of FF's (i,j) in Equation (1), define

$$P_{ij} = \overline{d}(i,j) - \underline{d}(i,j) + T_{setup} + T_{hold} + 2\delta \quad (2)$$

for an edge $q = (i,j) \in G_1(P)$. A lower bound, P_{low}, on the value of the optimal clock period, P, is

$$P \geq P_{low} = \max_{q=(i,j) \in G_1(P)} P_{i,j} \geq 0. \quad (3)$$

Lemma 4: An upper bound, P_{high} on the clock period, P, is given by

$$P \leq P_{high} = \max_{q=(i,j) \in G_1(P)} (\overline{d}(i,j) + T_{setup} + \delta) < \infty \quad (4)$$

where $q = (i,j)$ is an edge in $G_1(P)$.

Theorem 5: A finite solution to the clock period minimization problem exists.

The pseudocode for Phase I is as follows:

```
$P_{min} = P_{low}$;          ...(Lemma 3)
$P_{max} = P_{high}$;         ...(Lemma 4)
while $(P_{max} - P_{min}) > \epsilon$
    $P = (P_{max} + P_{min})/2$;
    if $G_1(P)$ has a positive cycle
        $P_{min} = P$;
    else
        $P_{max} = P$;
```

In the above algorithm, the presence of a positive cycle in $G_1(P)$ may be tested using the Bellman-Ford algorithm [4]. On a graph with V vertices and E edges, the computational complexity of this algorithm is $O(V \cdot E)$.

Theorem 6: The iterative procedure above, when carried to convergence, provides the solution to the LP in Equation (1) in time

$$O\left(F \cdot E \cdot log_2\left[(P_{high} - P_{low})/\epsilon\right]\right) \sim O(F \cdot E) \quad (5)$$

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, and P_{high} and P_{low} are as defined in Lemmas 3 and 4, 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 $O(F^2)$.

3 Phase II: Skew Minimization

At the end of the first phase, the optimal clock period and a corresponding clock skew for each FF are found. However, it may so happen that the skews computed here are very large, as in the example below:

Example: In Figure 3, the final constraint graph for a circuit is shown, with the final weights associated with its edges, and the skews calculated for each FF. Notice that the skew of the largest magnitude, associated with FF3, is -6; however, due to the existence of a slack on the path FFIO-FF3-FF4, one would have a functional circuit even if the skew on FF3 were reduced to -2. □

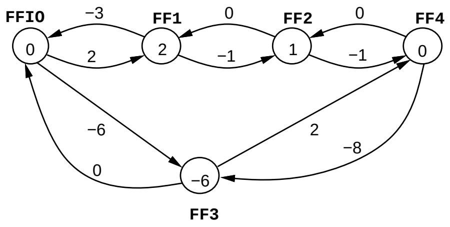

Figure 3: Example: Skew minimization at P_{opt}.

Clearly, a clock network that generates a skew of -6 would take up a larger area than one that generates a skew of -2, and hence it is a good idea to perform an optimization that reduces the magnitude of the maximum clock skew.

The problem of minimizing the magnitude of the maximum value of the clock skew, given a clock period, can be stated as follows:

$$\begin{array}{lc} \text{minimize} & r \\ \text{subject to} & -r \leq x_k \leq r \quad \forall \text{ FF's } k \text{ in the circuit} \\ & r \geq 0 \\ & x_i - x_j \geq T_{hold} - \underline{d}(i,j) + \delta \\ & x_j - x_i \geq T_{setup} + \overline{d}(i,j) - P_{opt} + \delta \end{array} \quad (6)$$

for every pair, (i,j) of FF's such that there is at least one combinational path from FF i to FF j. P_{opt} above is the minimum clock period calculated in Phase I. Note that since $x_k < P_{opt}$ $\forall$ $1 \leq k \leq F$ in the solution of Phase I, which is a feasible solution for the

LP in Equation (7), it is unnecessary to specify this constraint in Phase II. Note that the above is *not* a system of difference constraints due to the existence of the $x_k \geq -r$ constraints.

As in Phase I, a constraint graph, G_2, may be generated, and the minimum value of r that ensures the absence of a positive cycle in the constraint graph would provide the solution to the LP in Equation (7).

We observe that G_2 has the following properties:

(1) G_2 is identical to $G_1(P_{opt})$ except that for each node i in G_2, there is an arc from i to the root node s (whose skew is set to 0) of weight $-r$, and an arc from s to i of weight $-r$.
(2) The arcs with weights $-r$ are the only arcs in G_2 with variable weights.
(3) Since $G_1(P_{opt})$ has no positive cycles, and because of property (2) above, no cycle in G_2 with constant weights can be a positive cycle. As a result, if one ensures that all cycles through the arcs with weight $-r$ in G_2 are nonpositive, it is equivalent to ensuring that G_2 has no positive cycles.

One may take advantage of the last property in solving the LP in Equation (7).

The cycles that pass through arcs of weight $-r$ can be divided into three classes as shown in Figure 4. Type 1 cycles contain exactly one arc of weight $-r$ terminating at the root node, type 2 cycles contain two arcs of weight $-r$, while type 3 cycles contain one arc of weight $-r$ originating at the root node.

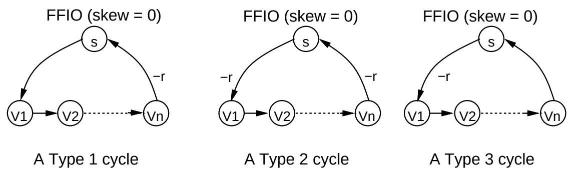

Figure 4: Types of cycles whose weights depend on r.

Lemma 7: Let $\mathcal{S} = x_1, \cdots, x_F$ be the values of the skews found in Phase I. Let $M = \max_{1 \leq i \leq F} \mid x_i \mid$.

(1) Decreasing the skew at any FF i in the circuit to less than x_i corresponds to an infeasible solution to the LP in Equation (1).
(2) As a result of (1), if $\exists x_i \in \mathcal{S}$ such that $x_i = M$, no reduction in M is possible in Phase II.

Type 1 Cycles

Let r_1 be the smallest positive value of r at which G' contains no positive Type 1 cycle. The following procedure finds the optimal value of r_1:
(1) Remove all arcs of weight $-r$ from G_2 to form G'.
(2) Find the optimal value of x_i in G' $\forall$ $1 \leq i \leq F$.
(3) Set $r_1 = \max_{1 \leq i \leq F} x_i$.

Notice that by definition, G' is identical to $G_1(P_{opt})$. Hence, the process of finding the optimal values of the x_i's corresponds to the results of the last application of the Bellman-Ford algorithm to $G_1(P_{opt})$.

Each cycle of Type 1 with k edges has $k-1$ edges from G' and one edge of weight $-r$. Therefore, to ensure that G_2 has no positive cycle of Type 1, it is enough to ensure that r is larger than the sum of the weights of all paths originating at the root node. Since the sum of arc weights from the root node to a node i is equal to x_i, the correctness of step 3 follows.

Note that since the x_i values are already available to us, the above algorithm corresponds to finding $\max_{1 \leq i \leq F} x_i$, which corresponds to the largest positive value of x_i in Phase I. Therefore, the value of r_1 is available with zero computational expense in Phase II.

Type 2 Cycles

Let G' be the graph formed by removing all arcs of weight $-r$ and all arcs terminating on the root node from graph G_2. Let r_2 be the smallest positive value of r at which G_2 contains no positive cycle of Type 2.
Lemma 8: Let the *critical* Type 2 cycle, i.e., the Type 2 cycle (if it exists) whose weight is zero when $r = r_2$, be $s, v_1, v_2, \cdots, v_n, s$, where s is the root node. Then

(1) the weight of any path from v_1 to $v_i, 2 \leq i \leq n$ cannot be negative, and
(2) the weight of any path from $v_i, 1 \leq i \leq (n-1)$ to v_n cannot be negative.

Based on this result, we propose the following procedure that finds the optimal value of r_2 for a type 2 cycle:

(1) Remove all arcs of weight $-r$ and all arcs terminating on the root node from graph G_2 to form graph G'.
(2) Initialize $x_i = 0$ $\forall$ vertices $1 \leq i \leq F$ in G'.
(3) Apply the Bellman-Ford algorithm to graph G' with the node weights initialized as in Step (2).
(4) Set $r_2 = \frac{1}{2} \max_{1 \leq i \leq F} x_i$.

Type 3 Cycles

Lemma 9: Let the *critical* Type 3 cycle, i.e., the Type 3 cycle (if it exists) whose weight is zero when $r = r_3$, be $s, v_1, v_2, \cdots, v_n, s$, where s is the root node. Then the weight of any path from v_1 to $v_i, 2 \leq i \leq n$ cannot be negative.

The smallest positive value of r, r_3, that ensures that G_2 has no positive cycles of Type 3 can be found by carrying out the procedure outlined in the previous subsection, and replacing Step (4) by

(4) If the weight of the arc from node i to the root node, s, is w_{is}, set $r_3 = \max_{1 \leq i \leq F, w_{is} < \infty} (x_i + w_{is})$.

Finding the Optimal Value of r

In the previous three subsections, techniques for calculating the values of r_1, r_2 and r_3 have been outlined. The calculation of r_1 involves finding the largest positive skew at the end of Phase I. To calculate r_2 and r_3, the Bellman-Ford algorithm is run once with all $x_i's$ initialized to zero instead of $-\infty$; at the end of this procedure, the formulæ provided earlier may be used to calculate r_2 and r_3.

To ensure that the graph G_2 contains no positive cycles, it is enough to show that no positive cycles of Types 1, 2, or 3 exist. This may be achieved by setting

$$r = \max(r_1, r_2, r_3) \tag{7}$$

Notice that as a consequence of Lemma 8 (2), if r_1 corresponds to the maximum value of $|x_i|, 1 \leq i \leq F$, then $r = r_1$, and it is not necessary to calculate r_2 and r_3. Thus, the additional expense incurred in Phase II is, at most, one execution of the Bellman-Ford algorithm.

4 Experimental Results

Experimental results on several circuits in the ISCAS89 benchmark suite are presented in Table 1. For each circuit, the the values of P_{low} and P_{high}, which are, respectively, the lower bound from Lemma 3, and the upper bound from Lemma 4, on the optimal clock period, P_{opt}. Note that P_{high} corresponds to the clock period in a circuit where all skews are set to 0. The optimal value of the clock period, P_{opt}, which is available at the end of Phase I is shown in the next column. The percentage figure denotes the improvement in the clock period over the zero-skew case corresponding to P_{high}. The next column shows the change in the value of $|x_i|_{max}$ after Phase II. In some of the circuits, the maximum magnitude skew value corresponds to a positive skew, and there is no reduction of the skew value in Phase II. This was particularly seen to be true for the smaller circuits, where the number of flip-flops is smaller. The execution time on a DECstation 5000/133 is shown for all of these circuits.

Table 1: Results of Clock Skew Optimization

Circuit	P_{min}/ P_{max}	P_{opt}	$\|x_i\|_{max}$ Change	CPU Time
s298	36/41	36.6(12%)	0.0	0.1s
s349	40/91	82.2(11%)	0.0	0.2s
s382	22/65	45.5(43%)	0.0	0.2s
s400	22/69	47.7(45%)	0.0	0.2s
s444	22/69	45.1(53%)	0.0	0.2s
s526	36/53	41.0(29%)	0.0	0.2s
s713	259/202	249.2(4%)	19.2	1.1s
s9234	150/179	165.1(8%)	41.4	1.8m
s9234.1	150/179	173.1(3%)	138.2	1.9m
s15850	292/373	314.0(19%)	276.0	24.1m

For example, for s15850, a circuit with 9772 gates and 597 FF's, the values of P_{low} and P_{high} were found to be 292 and 373 units, respectively. At the end of Phase I the value of the optimal clock period was found to be 314 units, a reduction of 19% over P_{high}. $|x_i|_{max}$ was 335 units; this was improved in Phase II to 59 units, corresponding to a reduction in the maximum skew value by 276 units. The algorithm in [1] does not account for this second optimization. The run time for the algorithm on this circuit was about 24 minutes.

In many cases, the clock period was found to improve. In some of the ISCAS benchmark circuits, however, no improvement in the clock period was possible through clock skew optimization; examples include s420.1, s5378 and s13207, perhaps due to the special structure of these circuits.

This method is a viable addition to the delay reduction tool-box of a designer, to be used in addition to techniques such as logic synthesis and sizing. In many cases, it is likely to help the designer arrive at an improved circuit with a reduced clock period.

The $|x_i|_{max}$ column provides a measure of the amount of work that the clock routing algorithm must perform; a larger skew variation implies that a larger number of buffers must be inserted into the clock distribution network. Therefore, the proposed algorithm provides a significant improvement over the results of [1]. While the skew values are still large, it must be remembered that the area overhead required to increase delays in the clock skew network is manageable; in fact, adding delays to the clock skew network may well require less overhead than sizing the circuit to achieve the amount of clock period reduction that we aim for. The reason for this is that the area increases at a very rapid rate as the delay constraint for a combinational circuit is made tighter, beyond a certain point [5]. However, the tradeoff lies in the fact that the task of clock network design is now harder. The improvement figures shown here indicate that this is a worthwhile task, considering the possible benefits.

5 Conclusion

We have examined the clock skew optimization problem and have exploited the special structure of the LP that describes the problem. An efficient graph-theoretic algorithm for minimizing the clock period has been presented. Next, the problem of achieving the minimum clock skew at this clock period is addressed, with the amount of additional computation required for this step being minimal. This is an improvement over previous approaches which neglected to address the problem of minimizing the clock skews. Results on several ISCAS89 benchmark circuits are presented, and substantial improvements in the clock period are found to be achievable in many cases.

REFERENCES

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

[2] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high-performance IC's," *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 573–579, 1990.

[3] R.-S. Tsay, "Exact zero clock skew," *Proceedings of the 1991 International Conference on Computer-Aided Design*, pp. 336–339, Nov. 1991.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw-Hill, 1990.

[5] S. S. Sapatnekar, P. M. Vaidya, and V. B. Rao, "A convex programming approach to transistor sizing for CMOS circuits," *Proceedings of the 1991 IEEE International Conference On Computer-Aided Design*, pp. 482–485, 1991.