Speeding up Pipelined Circuits through a Combination of Gate Sizing and Clock Skew Optimization

Harsha Sathyamurthy
Mentor Graphics
1001 Ridder Park Drive
San Jose, CA 95131.

Sachin S. Sapatnekar
Department of ECE
Iowa State University
Ames, IA 50011.

John P. Fishburn
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974.

Abstract

An algorithm for unifying the techniques of gate sizing and clock skew optimization for acyclic pipelines is presented in this paper. In the design of circuits under very tight timing specifications, the area overhead of gate sizing can be considerable. The procedure utilizes the idea of cycle-borrowing using clock skew optimization to relax the stringency of the timing specification on the critical stages of the pipeline. Experimental results verify that cycle-borrowing using sizing+skew results in a better overall area-delay tradeoff than with sizing alone.

1 Introduction

The problem of optimizing acyclic pipelines has attracted considerable interest of late. This paper presents a method for speeding up acyclic pipelined circuits through a combination of two techniques: gate sizing and clock skew optimization. Each technique has been utilized in isolation for speeding up circuits; we unify them to arrive at an optimal design.

Gate sizing is a well-known technique and several CAD tools have been developed to perform this optimization [1, 2]. Given the circuit topology, the delay of a combinational circuit can be controlled by varying the sizes of transistors in the circuit. In coarse terms, the circuit delay can usually be reduced by increasing the sizes of gates in the circuit, entailing the penalty of increased circuit area and power dissipation; this trade-off is the gate sizing problem. We stress here that this method is applied individually to each combinational subcircuit of a sequential circuit. The problem is commonly formulated as

$$\text{minimize} \quad Area \quad \text{subject to } Delay \leq P_{spec} \qquad (1)$$
$$\text{or minimize} \quad P \quad \text{subject to } Area \leq A_{spec} \qquad (2)$$

where P is the clock period. In the former case, a goal period is specified, which restricts the delay of each combinational segment. In the latter formulation, the clock period is minimized subject to a specification, A_{spec}, on the area. $Area$ is typically measured as the sum of all the transistor channel widths.

To understand the problem of clock skew optimization, it is important to recognize that due to differences in interconnect delays on the clock distribution network of integrated circuits, there is typically a skew between the arrival times of clock signals at the flip-flops (FF's). One approach that has been followed by several researchers is to design the clock distribution network so as to ensure zero clock skew. An alternative approach [3,4] 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. This process of selecting the optimal skews is the problem of clock skew optimization.

To illustrate the benefits of clock skew optimization, consider the following example. In Figure 1, if the combinational subcircuits CC_1 and CC_2 have delays of 6 and 14 units, respectively, then with zero skew, the fastest allowable clock has a period of 14 units. If a skew of -4 units is applied to the clock line to latch B, the circuit can run at a clock period of 10 units. Algorithms for skew optimization were presented in [3,4], where clock skew optimization problem was formulated as a linear program (LP).

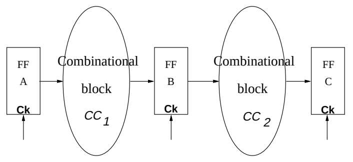

Figure 1: The advantages of nonzero clock skew.

We now motivate the need for using skew optimization in conjunction with sizing. The general character of the area-delay tradeoff for sizing alone (without using skew optimization) is shown by the curve in Figure 2. For a loose delay specification, the area penalty is not very large, but for tighter specifications, it becomes extremely large. Referring to Figure 1, the use of skew optimization in conjunction with sizing would allow CC_2 to *steal* a part of the clock cycle for CC_1, thereby allowing it a larger timing budget and a correspondingly smaller area penalty. For example, in Figure 2, this may allow CC_2 to move from position A to position B. Correspondingly, CC_1 may move from position C to position D. The nonlinearity of the area-delay tradeoff curve ensures that the area corresponding to moving CC_1 from C to D is smaller than the area savings corresponding to moving CC_2 from A to B.

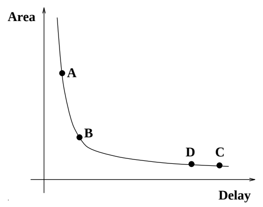

Figure 2: Using sizing in conjunction with clock skew.

In this paper, we present an algorithm for unifying clock skew optimization and transistor sizing. The algorithm is directed towards circuits with edge-triggered FF's. The solution may be arrived at in two steps. In Step 1, the circuit delay is minimized and the long path constraints satisfied, ignoring the short path constraints. Next, in Step 2, the short path constraint violations are resolved by padding the circuit with buffers as necessary. This paper focuses on the solution of the Step 1, and adaptations of techniques such as [5] may be used to reconcile short path violations. The technique is illustrated on single-phase clocked circuits containing edge-triggered FF's.

It was previously thought that it was extremely hard to achieve a good solution to the sizing+skew problem under Elmore delays. To the best of of our knowledge, this is the first piece of work that shows that Step 1 corresponds to a convex optimization problem, which implies that it is easy to find a solution to this subproblem. Therefore, if, as is likely for practical circuits, the number of short path violations in the final circuit is relatively small, Step 2 will perturb the solution by only a small amount, and the above technique will work well for practical circuits. Another contribution of this work is in the application of PERT to acyclic sequential circuits to detect long path constraint violations.

The organization of this paper is as follows. Section 2 formally states the problem to be solved, followed by an overview of the algorithm in Section 3. The procedure for detecting long path constraint violations is described in Section 4. The entire algorithm is then presented in Section 5. Finally, experimental results are provided in Sections 6.

2 Statement of the Clock Skew Optimization Problem

2.1 The Clock Skew Optimization Problem

For each FF pair (FF_i,FF_j), let x_i, x_j be the skews at the FF's FF_i and FF_j respectively and $d(i,j)$ be the delay of the combinational block between them (with $\underline{d}(i,j)$ being the minimum delay and $\overline{d}(i,j)$ being the maximum delay). Let T_{hold} be the FF hold time and T_{setup} be the FF setup time. Two types of timing errors may exist, corresponding to double-clocking (short path violations) and zero-clocking (long-path violations). It has been shown in [3] that the minimization of the clock period can be written as an LP, with constraints reflecting the absence of zero-clocking and double-clocking conditions. The LP may be stated as:

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

where the factor δ models the uncertainty in the skews. If one can guarantee that in the manufactured circuit, the skew at each FF k, $\hat{x}_k$, will be within the range $[x_k - \delta/2, x_k + \delta/2]$, where x_k is the design value of the skew, then the difference between any skews, $(\hat{x}_i - \hat{x}_j)$ in the manufactured circuit, must be within δ of the design value of $(x_i - x_j)$.

In the case where all gate delays are constant, the above optimization problem is an LP in the skew variables and the clock period [3]. If we consider the gate delays as functions of the gate sizes, then $\underline{d}(i,j)$ and $\overline{d}(i,j)$ are functions of the gate sizes, and the optimization problem is considerably more complex. We will elaborate on this in Section 3.

2.2 Is Clock Skew Optimization Safe?

A common misconception that persists among circuit designers about changing clock skews is that it is believed to be 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 [3, 4] that ensure that skewing errors do not disrupt circuit functionality. These safety margins ensure that the circuit will operate in the presence of unintentional process-dependent skew variations. Introducing deliberate delays within the clocking network has been a tactic that has long been used by designers [6], and this may be adapted to build fixed-skew clock networks [7].

In fact, it is a misconception to believe that zero skew is entirely safe. To see this, consider a shift register consisting of register A whose output is connected to register B with *no* combinational logic between the two. Even for a circuit designed for zero skew, a small unintentional positive skew at register B will cause double-clocking. Such problems may be avoided by the use of these safety margins and the introduction of deliberate nonzero skew: a small amount of deliberate positive skew at A provides an effective safety margin against double-clocking.

3 Formulation of the Problem

The combination of clock skew optimization and sizing into a single framework was thought to be too intractable a problem to solve [3]. A recent approach in [8] used piecewise linear models to arrive at a solution, setting up the combined problem as an LP. However, the accuracy of such models is limited, and hence it is desirable to investigate techniques that use the more accurate Elmore delay model directly.

The problem of sizing with skew optimization is stated as:

$$\begin{array}{rcll} \text{minimize} & Area & & \\ \text{subject to } x_i + \overline{d}(i,j) + T_{setup} & \leq & x_j + P + \delta & (3) \\ -X_{max} & \leq & x_i \leq X_{max} & (4) \end{array}$$

where $\overline{d}(i,j)$ is a function of the gate sizes in the circuit, P is the specified clock period, and X_{max} is the maximum allowable skew magnitude. The $Area$ objective function is approximated as the sum of all transistor sizes. Note that only long-path constraints have been considered.

The area of the clocking network is not included here for two reasons. Firstly, it is difficult to derive a relation between the skews and the clocking network area (however, as we will show, safeguards may be added to prevent the clocking network area from becoming exceedingly large). Secondly, the complexity of routing the clock network is such that it is not possible to predict whether a nonzero skew clock tree will necessarily utilize more routing resources than a zero-skew tree.

However, we make the observation that for large clock skew magnitudes, the clock tree is likely to consume large routing resources. Therefore, to ensure that the expense of clock routing does not run amuck, we introduce the constraint (4). This constraint may easily be incorporated into our solution.

Under the Elmore delay model, it can be shown [1] that the gate delays are posynomial functions [9] of the gate sizes. A posynomial function in $\mathbf{x}$ can be transformed into a convex function in $\mathbf{z}$ using the mapping $x_i = e^{z_i}$. Based on this fact, it was pointed out in [3] that the above optimization problem is a *signomial* programming problem [9], which makes it difficult to arrive at a good solution to the problem.

Each long path constraint is of the form

$$\overline{d}(i,j) + x_i - x_j \leq K, \quad (5)$$

where $\overline{d}(i,j)$ is some posynomial function of the transistor sizes, x_i and x_j are clock delays to the source and destination FF's, and K is a constant. The left-hand side of this inequality is not a posynomial because of the negative coefficient of x_j. If the logarithmic substitution, $x = e^z$, were performed for each clock skew and transistor size variable x in this inequality, the result would not be a convex constraint.

However, if we substitute $w = e^z$ for each transistor width w appearing in $\overline{d}(i,j)$, while leaving x_i and x_j alone, then the result is a convex constraint: the left-hand side of (5) is the sum of $\overline{d}(i,j)$ (which is convex in the z variables) and $x_i - x_j$, which is linear (hence convex), in x_i and x_j. *This result holds for general sequential circuits, and not just acyclic pipelines.*

More generally, if the variables in an optimization problem can be divided into two separate classes $w_1, \cdots, w_n$, and $x_1, \cdots, x_m$, and if each constraint is of the form

$$P(w_1, ..., w_n) + C(x_1, ..., x_m) \le K, \qquad (6)$$

where P is posynomial, C is convex, and K is a constant, then the problem can be transformed into a convex program by substituting $w_i = e^{z_i}$ while leaving the x_i variables alone.

The solution of the problem proceeds as follows. The optimization approach is divided into two stages that are repeated iteratively. In the first, violations of the clocking constraints must be detected. We present a modification of the PERT procedure for delay estimation, generalized to handle sequential circuits, in Section 4. Next, a critical path is defined and identified, and the size of the most sensitive gate on this path is bumped up by a small amount. The iterations continue until all long path-constraints are satisfied for the given clock period.

4 Detection of Long-Path Constraint Violations

4.1 The "Delay" of a Flip-Flop

PERT is a method for finding the longest/shortest path in a directed acyclic graph (DAG) that arises out of a system of difference constraints. Here, we present a generalization to sequential circuits of the PERT method that has been used extensively for delay estimation in combinational circuits.

The technique for representing combinational blocks by difference constraints, and therefore, by DAG's is well-known. For sequential circuits, one also has to deal with the problem of representing FF's. In this section, it is shown that the input delay-output "delay" for FF's can also be represented by a difference constraint. Note that the word "delay," when used in reference to FF's, is applied in a loose sense here. It refers not to the propagation delay of the gates within the FF, but to a mathematical tool used to apply PERT.

For acyclic pipelines, the composite set of these and the difference constraints for each gate can be represented by a DAG, on which PERT may be applied to identify the gates on the critical path in the sequential circuit. It is noteworthy that the circuit will remain a DAG only when the pipeline is acyclic; if not, methods such as loop unrolling will have to be utilized. It will now be shown that in an acyclic sequential circuit, a memory element is equivalent to a "delay" of $T_{setup} - P - \delta$.

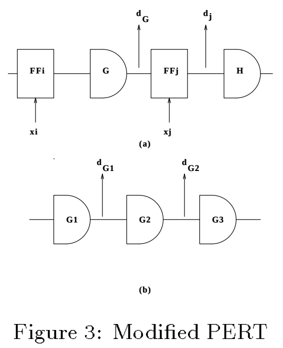

Figure 3: Modified PERT

Consider Figure 3(a). From (3) we have $x_i + G_{maxdelay} + T_{setup} \le x_j + P + \delta$. where x_i is the latest arrival time at the input of gate G, x_j is the latest arrival time at the input of gate H. Note that $G_{maxdelay}$ is not necessarily the delay of a single gate, but can, in general, be the delay of a path from FF_i to FF_j. Writing $x_i + G_{maxdelay}$ as d_G, the latest arrival time at the input of FF_j, we have the following difference constraint

$$\begin{array}{lll} & d_G + T_{setup} & \le x_j + P + \delta \\ \text{i.e.} & x_j - d_G & \ge T_{setup} - P - \delta \end{array} \qquad (7)$$

For a regular combinational gate, as shown in Figure 3(b), if d_{G1} and d_{G2} are the latest arrival times at the input of $G2$ and $G3$, then we have the difference constraint, $d_{G2} - d_{G1} \ge delay(G2)$. From this, we see that $G2$ and FF_j behave analogously, i.e., FF_j behaves like a gate with a delay of $T_{setup} - P - \delta$.

The value x_j also represents the skew associated with the FF. Therefore, a violation occurs if at the end of PERT, the arrival time at a primary output is greater than zero, since the skew at each primary input or output is zero and unchangeable. (Notice that if the arrival time at a primary output is ≤ 0, we have a nonnegative slack for the long path constraints.)

4.2 The Delay of a Gate

The method used here for modeling the delay of a gate has been used extensively and successfully in several gate sizing algorithms, and is not original. The delay of each gate is calculated by replacing it by an equivalent inverter with parameters W_N and W_P corresponding to the equivalent n and p transistor sizes, respectively. Each gate is replaced by an RC tree whose Elmore delay is taken to be the delay of the gate, as in [1].

5 Unifying Sizing and Clock Skew Optimization

This section presents an overview of the algorithm. In the delay model presented in Section 4, a gate is represented by an equivalent inverter. The n- and p-transistors of the gate are sized independently. Initially, all gate sizes are set to the minimum value. Each iteration performs the following steps:
(1) A timing analysis is conducted on the sequential circuit using PERT to identify paths that fail to satisfy the delay constraint. A violation occurs if the rise/fall delay at a PO is ≥ 0. Starting from the PO with the maximum violation, a critical path is traced back to a PI. For circuits with inverting gates, the critical path is an alternating sequence of rise and fall transitions.
(2) The sensitivity of each gate along the critical path is computed, and the most sensitive gate is identified. If this gate is undergoing a rise (fall) transition, the p-(n-)channel width for this gate is increased by a multiplicative factor, bumpsize.

In each iteration, the delay of the critical path is expected to be reduced by a small amount. However, any change on the current critical path is liable to have repercussions on the delays of other paths in the circuit, and hence, as in [1], we use a bumpsize that is just above 1.0. The algorithm continues until all timing requirements are met, or until the sensitivity of the most sensitive gate in the most critical path is ≥ 0. In the latter case, any increase in channel widths results only in an increase in delay, and the specified clock period is unachievable. The optimal skew values fall out as a natural consequence of this procedure: the arrival time at an FF output, as calculated by PERT, is the optimal skew to be applied to that FF.

6 Experimental Results

The algorithm has been implemented in a C program, SACSOFON (Sizing And Clock Skew Optimization For One-phase clocked Networks). Experimental results are provided on several circuits, described in Table 1. The circuits marked with a "†" are single-stage pipelines; the results on these circuits with and without skew optimization must be identical since skew adjustments can only be made on internal FF's.

Table 1: Input Circuit Description

Circuit	# Gates	# Stages	# FF's	# PI's	# PO's
add2†	15	1	5	3	2
inv10†	10	1	2	1	1
r500†	520	1	40	20	20
mcnc	744	3	24	10	8
r_2	279	2	26	15	5
r_4	496	4	35	15	4
r_6	1370	6	78	25	10
r_10	1687	10	91	25	6
r_15	2986	15	142	25	9
r_20	4043	20	176	20	7

† single-stage pipelines

Experimental results on these circuits, using transistor sizing with and without clock skew optimization, are presented in Table 2. The column labeled P_{max} corresponds to the clock period of an unsized circuit where all clock skews are set to 0, and P_{spec} is the specified clock period to be achieved. The percentage increases in circuit area as a result of sizing are, respectively, Δ^{nsk}_{Area} and Δ^{sk}_{Area}. The corresponding CPU times for SACSOFON on a Sun Sparcstation 10 are also shown.

Table 2: Combining Clock Skew Optimization and Sizing

Circuit	P_{max} (ns)	P_{spec} (ns)	Δ^{nsk}_{Area} (0skew)	Δ^{sk}_{Area} (skew)	CPU (0skew)	CPU (skew)
add2†	5.38	3.55	46.00%		0.36s	0.36s
inv10†	5.76	4.25	55.13%		0.32s	0.32s
r500†	48.59	30.00	3.79%		15.94s	15.94s
mcnc	77.74	18.0	- ‡	42.75%	-	107.6s
r_2	28.95	12.0	31.09%	23.69%	15.9s	14.8s
r_4	26.31	12.0	32.68%	23.98%	45.1s	55.9s
r_6	36.11	14.0	45.88%	32.10%	482.2s	396.7s
r_10	36.40	14.0	36.22%	26.51%	618.6s	488.6s
r_15	38.76	14.0	44.71%	31.82%	2885.5s	2305.5s
r_20	40.38	14.0	38.00%	26.51%	3656.9s	2830.8s

† single-stage pipelines ‡ unachievable specification

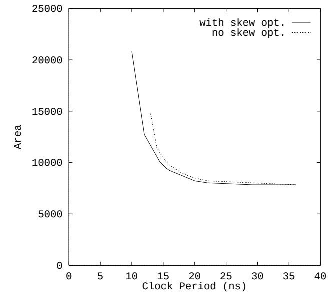

Figure 4: Increase in area vs. clock period for r_6

In Table 2, for both sizing with and without clock skew optimization, a bump size of 1.25 is used and no limit on the increase in area is imposed. (It may be recalled that the bump size is the factor by which the size of the most sensitive transistor is increased in each iteration. It was found experimentally that a bumpsize between 1.2 and 1.75 provides a good tradeoff between accuracy and execution time.) It can be seen that in the case where clock skew optimization has been applied, the clock periods achieved are smaller than those achieved by not introducing clock skews. As expected, in the cases of the single-stage pipelines, clock skew optimization cannot be applied and the results for each case are the same. For the remaining circuits, it can be seen that wherever the specification, P_{spec}, is achieved by both the methods, $\Delta^{nsk}_{A} > \Delta^{sk}_{A}$. In other words, the increase in area is greater when clock skew optimization is not applied. It may also be observed that the period P_{spec} is sometimes unachievable using sizing alone, but is achieved by the use of skew optimization in addition to sizing.

For example, for r_20, a 20-stage pipeline with 4043 gates, the clock period of the unsized circuit is 40.38 ns. The specified clock period is 14 ns. The increase in area due to sizing without skew is 38%, as against about 26.5% for sizing+skew. The CPU times are seen to be similar in all cases. For some specifications for the circuits (as is shown later), the clock period specified here was not achievable by transistor sizing alone, demonstrating the utility of unifying transistor sizing and clock skew optimization.

Figure 4 indicates the improvement achieved when clock skew optimization is applied along with transistor sizing for circuit r_6. Not only is the increase in area less, but the clock periods are also reduced further. For example, in Figure 4, even the best achievable clock period with sizing alone can be achieved by sizing+skew optimization at about half the cost, and that significantly lower clock periods are also achievable at a reasonable cost. The curves for other circuits are similar.

The curves show us that for a given timing specification, the area utilized by the sizing-only solution is always larger than that for the sizing+skew solution. We caution the reader not to be misled by the fact that the two curves in each figure seem to be very close to each other; the area differences are acute when the circuit is designed for very tight timing constraints, where we try to push the limits of the achievable circuit speed.

References

[1] J. Fishburn and A. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," *Proc. ICCAD*, pp. 326–328, 1985.

[2] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Trans. CAD*, pp. 1621–1634, Nov. 1993.

[3] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Comp.*, pp. 945–951, July 1990.

[4] R. B. Deokar and S. S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," *Proc. ISCAS*, pp. 1.407–1.410, 1994.

[5] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," *Proc. ICCAD*, pp. 156–161, 1993.

[6] K. D. Wagner, "Clock system design," *IEEE Design and Test of Computers*, pp. 9–27, Oct. 1988.

[7] R.-S. Tsay, "An exact zero-skew clock routing algorithm," *IEEE Trans. CAD*, pp. 242–249, Feb. 1993.

[8] W. Chuang, S. S. Sapatnekar, and I. N. Hajj, "A unified algorithm for gate sizing and clock skew optimization to minimize sequential circuit area," *Proc. ICCAD*, pp. 220–223, 1993.

[9] J. Ecker, "Geometric programming: methods, computations and applications," *SIAM Review*, vol. 22, pp. 338–362, July 1980.