Buffer Sizing for Clock Power Minimization subject to General Skew Constraints

Kai Wang and Malgorzata Marek-Sadowska

Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106-9560, USA
{wk,mms}@ece.ucsb.edu

ABSTRACT

In this paper, we investigate the problem of buffer sizing for clock power minimization subject to general skew constraints. A novel approach based on sequential linear programming is presented. By taking the first-order Taylor's expansion of clock path delay with respect to buffer widths, the original nonlinear problem is transformed to a sequence of linear programs, which incorporate clock skew scheduling and buffer sizing to minimize clock power dissipation. For each linear program, the sensitivities of clock path delay with respect to buffer widths are efficiently updated by applying time-domain analysis to the clock network in a divide-and-conquer fashion. Our approach can take process variations and power supply noise into account. We demonstrate experimentally that the proposed technique is not only capable of effectively reducing clock power consumption, but also able to provide more accurate delay and skew results compared to the traditional approach.

Categories and Subject Descriptors

B.8.2[Performance and Reliability]: Performance Analysis and Design Aids

General Terms

Algorithms

Keywords

Clock skew scheduling, sizing, sequential linear programming

1. INTRODUCTION

In a synchronous digital design, the most common strategy for clock distribution is to insert a large number of buffers along the paths from clock source to sinks, forming a buffered tree structure. Carrying large loads and switching at the highest frequency, clock is one of the major sources of power dissipation in a digital integrated circuit. Reducing clock power dissipation is an effective technique for low-power designs. The clock skew, defined as the maximum difference between the arrival times of the signals at all the clock sinks, significantly impacts the performance of the system. Clock power and skew optimization have become the dominant objectives in clock design.

Traditional works in clock network design achieve zero or bounded skew, by adjusting either the wire lengths ([12], [2]) or the wire/buffer widths ([17]). Some other works focus on clock skew scheduling, which determines the intensional skews for clock sinks to improve the system performance or reliability. Existing clock skew scheduling techniques ([4], [10], [13]) generate a clock schedule without considering its impact on the layout of clock network, thus the resulting clock network based on the pre-defined skew may not achieve an optimal solution in terms of area or power. Recently, the UST-BP algorithm in [16] and the UST/DME algorithm in [11] have incorporated simultaneous skew scheduling and clock routing. The authors consider the routing problem of useful skew tree (UST) under general clock skew constraints. However, clock buffers are not considered in their formulations.

The majority of the existing methods of clock design are based on analytical delay models, such as Elmore delay [3] for interconnects, and simple RC model [14] for buffers. As the process technology scales down, many physical effects previously considered to be second-order, now are becoming prominent. Those effects may not be accurately captured by the analytical delay models. It is worth noting that the accuracy required to calculate delay differences is much greater than the accuracy required to calculate the absolute delay values[5]. Hence clock skew optimization that depends only on analytical delay models may not be sufficiently accurate in skew control.

Clock buffer sizing is an effective technique for skew control and power minimization. The problem of low-power clock buffer sizing for the bounded-skew constraint has been studied in [15]. In practice, general skew constraints introduce substantive flexibility in optimizing clock performance and power dissipation. In this paper, we consider the problem of buffer sizing for clock power minimization subject to general skew constraints. We propose a novel approach based on sequential linear programming (SLP). We transform the constrained nonlinear programming problem into a sequence of linear programs, by taking the first order Tay-

DAC'04, June 7–11, 2004, San Diego, California, USA.

lor's expansion of clock path delay with respect to buffer widths. Instead of relying on any analytical delay model, our technique applies time-domain analysis to the clock network to compute the sensitivities of clock path delay with respect to buffer widths. This calculation is efficient because we adopt a divide-and-conquer algorithm, which has a linear complexity with respect to the number of clock buffers. Our technique incorporates clock skew scheduling and buffer sizing, such that the resulting skew schedule is not only feasible, but is also optimal for clock power minimization. We compute accurate skew and delay because our technique applies time-domain analysis. We can also take into account the impact of process variations and power supply noise.

The rest of the paper is organized as follows. Section 2 describes the problem formulation. In section 3 we discuss the sequential linear programming approach. The sensitivity calculation algorithm is presented in section 4. In section 5 we show how to extend our technique to consider the impact of power supply noise. We give experimental results in section 6, and conclude the paper in section 7.

2. PROBLEM FORMULATION

To facilitate our discussion, we introduce the following notations. Let T be the given initial buffered clock tree with M clock sinks and K clock buffers. For each clock buffer $b_k(k = 1, \ldots, K)$, w_k is its width and w_k^t, w_k^b are the upper and lower bounds, respectively. We use $p(b_k)$ to denote b_k's direct parent buffer, $C(b_k)$ to denote the set of b_k's direct child buffers. $slew(b_k)$ denotes the slew rate of the clock signal at the b_k's input node, and $FC(b_k)$ denotes the set of the clock sinks that are in b_k's fan-out cone. For each clock sink $s_j(j = 1, \ldots, M)$, d_j is the delay from the clock tree root to s_j. For a sink $s_j \in FC(b_k)$, we use $b_{(k,j)}$ to denote the b_k's direct child buffer that is on the path from b_k to s_j, and $d^s(b_k, s_j)$ to denote the delay from the b_k's input node to s_j. Finally, $d^r(b_k)$ denotes the delay from clock tree root to the b_k's input node, and $d^b(b_k, b_j)$ denotes the delay from the input node of buffer b_k to the input node of another buffer b_j if there is a path from b_k to b_j.

Consider a synchronous circuit with positive edge-triggered flip-flops in a single-phase clocking scheme. Let FF_i and FF_j be two sequentially adjacent flip-flops, with FF_i feeding data to FF_j. We use t_i and t_j to denote the signal arrival time at the clock pins of FF_i and FF_j respectively. To ensure correct logic operations, we must bound the skew between FF_i and FF_j from above and below by the following local skew constraints:

$$t_i - t_j \geq t_{hold}^{max} - d_{pFF}^{min} - d_{logic}^{min} \quad (1)$$

$$t_i - t_j \leq C_P - d_{pFF}^{max} - d_{logic}^{max} - t_{setup}^{max} \quad (2)$$

where d_{logic}^{min} and d_{logic}^{max} are the minimum and maximum delays through the combinational logic; d_{pFF}^{min} and d_{pFF}^{max} are the minimum and maximum delay through the flip-flop; t_{hold}^{max} and t_{setup}^{max} are the maximum hold and setup time for flip-flop; and C_P is the clock period. In general, we use $SC = \{t_i - t_j \in [l_{i,j}, u_{i,j}]\}$ to represent the set of skew constraints for all sequentially adjacent clock sinks, which must be satisfied by any clock skew schedule.

The problem of buffer sizing for clock power minimization subject to general skew constraints is formulated as follows. Given an initial buffered clock tree T and a set of skew constraints SC, decide the buffer widths such that the total clock power dissipation is minimized:

$$Min \quad \sum_{k=1}^{K} \phi_k w_k \quad (3)$$

subject to skew, buffer width and delay constraints. ϕ_k is the constant decided by technology. We discuss the constraints below.

1) General skew constraints:
for a pair of sequentially adjacent clock sink s_i and s_j

$$l_{i,j} \leq d_i - d_j \leq u_{i,j} \qquad [l_{i,j}, u_{i,j}] \in SC \quad (4)$$

To consider process variation, we assume some uncertainty in the clock delays. There are two constants, $0 < \alpha \leq 1 \leq \beta$, having the property that if the nominal clock delay is d_j, the actual clock delay d_j' must always fall in the interval $[\alpha d_j, \beta d_j]$. For mature process technology, the values of α and β can be estimated from previous statistical data. Therefore, considering process variation, the skew constraint Eqn.(4) is replaced by:

$$\beta d_i - \alpha d_j \leq u_{i,j} \quad (5)$$

$$\alpha d_i - \beta d_j \geq l_{i,j} \qquad [l_{i,j}, u_{i,j}] \in SC \quad (6)$$

2) Buffer width bounds constraints:

$$w_k^b \leq w_k \leq w_k^t \qquad k = 1, \ldots, K \quad (7)$$

3) Maximal delay constraint:

$$d_j \leq D_{max} \qquad j = 1, \ldots, M \quad (8)$$

where D_{max} is the maximal delay specified by the designer.

The above formulation and the proposed technique can also be extended to handle the skew minimization and bounded-skew optimization problem. Specifically, the general skew constraints are replaced by

$$d_f \leq d_j \leq d_s \qquad j = 1, \ldots, M \quad (9)$$

where d_f is the least delay, and d_s is the largest delay among all d_js. For skew minimization, the cost function is the weighted sum of $d_s - d_f$ and the buffer area. For bounded-skew optimization, the skew bound constraint is added:

$$d_s - d_f \leq Skew_{max} \quad (10)$$

When two sequentially adjacent clock sinks are driven by the same clock buffer, buffer sizing cannot adjust the skew between them. This situation can be handled by adjusting wire lengths or widths. Therefore, for simplicity, we do not consider that case in this work. Although in this paper we deal only with buffer sizing, our technique can be easily extended to handle wire sizing as well.

In this paper, we consider only the tree topology clock network. Instead of using analytical delay models, we assume the SPICE [7] model for clock buffers. A wire is modeled as a lumped π-type circuit. We do not preclude using more advanced interconnect models such as the distributed RLC model.

3. SEQUENTIAL LINEAR PROGRAMMING APPROACH

We need to determine the clock path delay as functions of buffer widths, *i.e.*:

$$d_j = d_j(w_1, \ldots, w_K) \qquad j = 1, \ldots, M \quad (11)$$

The formula must be computationally simple and provide relatively high accuracy. In general, d_j is a nonlinear function with respect to all w_ks. Traditionally, clock skew scheduling is formulated as a linear programming problem. Therefore, to incorporate skew scheduling with buffer sizing, a linear expression is preferred. When we change a buffer size by a small amount Δw_k, the corresponding change of clock sink delay d_j is linear with respect to Δw_k. We take the first-order Taylor's expansion of Eqn.(11) at the nominal values $(w_1^0, \ldots, w_K^0)$ and yield:

$$d_j \approx d_j^0 + \sum_{k=1}^{K} \frac{\partial d_j}{\partial w_k}_{|w_k = w_k^0} (w_k - w_k^0) \tag{12}$$

where $d_j^0 = d_j(w_1^0, \ldots, w_K^0)$.

If we use Eqn.(12) to approximate d_j, all the constraints in the previous problem formulation become linear. We can transform the original nonlinear problem to a linear programming problem if the changes of variables are limited to a small range, *i.e.*, the buffer width bounds constraints in Eqn.(7) are replaced by the following constraints:

$$\begin{aligned} max(w_k^b, w_k^0 - \Delta w_k) \leq w_k &\leq min(w_k^t, w_k^0 + \Delta w_k) \\ k &= 1, \ldots, K \end{aligned} \tag{13}$$

The above linear formulation incorporates skew scheduling and buffer sizing. The solution resulting from the linear programming problem is used as the intermediate solution of the original nonlinear problem. To reduce further the objective function, after updating the buffer and wire widths according to the linear problem solution, we can form a new linear programming problem by updating the linear coefficients, *i.e.*$\frac{\partial d_j}{\partial w_k}$, at the new nominal values. In this way, a sequence of linear programs is formed. The clock skew schedule is gradually refined such that the clock power dissipation is reduced.

For each linear program we must carefully choose the Δw_k—that is, the ranges in which buffer widths are allowed to change. If this range is too large, the linear approximation may not be sufficient, and the result of the linear program may not reduce the objective function of the original nonlinear problem. On the other hand, if these ranges are too small, the reduction of the objective function may be very small; thus the sequence of linear programs may converge very slowly. Our experiments show that it is possible to choose such a range that the sequence of linear programs can converge in several iterations and still maintain a linearity.

The general method of the sequential linear programming method is also known as *the Kelly's cutting plane method* and *Stewart and Griffith's Method of Approximate Programming*. This method is shown to converge in finite steps [1]. Even though the idea of sequential linear programming (SLP) is considered unattractive by theoreticians due to its poor converging property for general nonlinear optimization problems, the concept has proved to be quite powerful and efficient for practical engineering problems. Our experimental results suggest that our clock buffer sizing problem is one case for which SLP is effective.

4. SENSITIVITY CALCULATION

For each linear program, it is essential to compute $\frac{\partial d_j}{\partial w_k}$, *i.e.*, the sensitivity of clock path delay with respect to buffer width. When analytical models are used, delay sensitivities can be determined using symbolic differentiation. However, their limited accuracy becomes a serious problem for the skew optimization. Delay sensitivity calculation based on time-domain analysis overcomes many of the above limitations. Not only can it provide accurate results, but it can also handle the impact of power supply variations. However, the size of clock trees can be very large in modern VLSI chips. Using time-domain simulator iteratively on a large clock tree is computationally expensive. To overcome this difficulty, we propose a divide-and-conquer approach, which efficiently applies time-domain analysis to calculate $\frac{\partial d_j}{\partial w_k}$.

4.1 Formulas

We consider a pair composed of a buffer b_k and a sink s_j. There are three possible cases:

1) $s_j \notin FC(p(b_k))$

If s_j is not in the fan-out cone of b_k's parent buffer, the change of w_k will not affect d_j due to buffer's shielding effect. Therefore in this case we have:

$$\frac{\partial d_j}{\partial w_k} = 0 \quad \text{if } s_j \notin FC(p(b_k)) \tag{14}$$

2) $s_j \in FC(p(b_k))$ and $s_j \in FC(b_k)$

In this case, s_j is in the fan-out cone of b_k. The delay from the clock tree root to the s_j can be decomposed into three parts via b_k's parent and child buffers. This is illustrated in Figure 1.

$$d_j = d^r(p(b_k)) + d^b(p(b_k), b_{(k,j)}) + d^s(b_{(k,j)}, s_j)$$

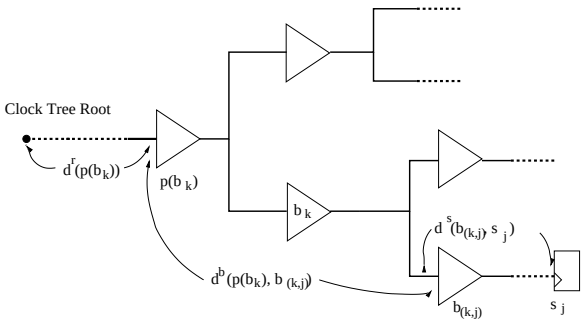

Figure 1: Case 2

In Figure 1, $\frac{\partial d^r(p(b_k))}{\partial w_k} = 0$ due to the shielding effect of $p(b_k)$. The change of w_k affects $slew(b_{(k,j)})$ which in turn affects $d^s(b_{(k,j)}, s_j)$ Then we apply the chain rule of derivative:

$$\frac{\partial d^s(b_{(k,j)}, s_j)}{\partial w_k} = \frac{\partial d^s(b_{(k,j)}, s_j)}{\partial slew(b_{(k,j)})} \frac{\partial slew(b_{(k,j)})}{\partial w_k} \tag{15}$$

Therefore when $s_j \in FC(b_k)$, $\frac{\partial d_j}{\partial w_k}$ can be expressed as:

$$\frac{\partial d_j}{\partial w_k} = \frac{\partial d^b(p(b_k), b_{(k,j)})}{\partial w_k} + \frac{\partial d^s(b_{(k,j)}, s_j)}{\partial slew(b_{(k,j)})} \frac{\partial slew(b_{(k,j)})}{\partial w_k} \tag{16}$$

3) $s_j \in FC(p(b_k))$ and $s_j \notin FC(b_k)$

In this case, the s_j is not in the fan-out cone of b_k, but it is in the fan-out cone of another child buffer of $p(b_k)$. d_j

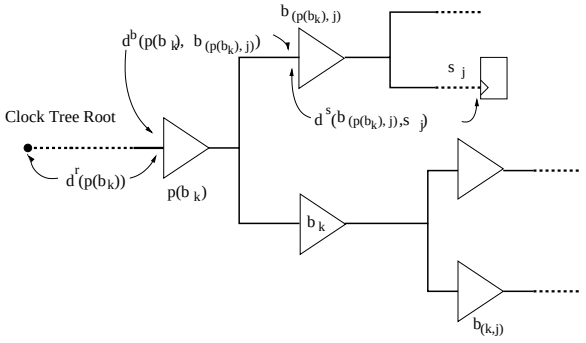

Figure 2: Case 3

can also be decomposed into three parts as illustrated in Figure 2:

$$d_j = d^r(p(b_k)) + d^b(p(b_k), b_{(p(b_k),j)}) + d^s(b_{(p(b_k),j)}, s_j)$$

Similar to the second case, we have:

$$\frac{\partial d^s(b_{(k,j)}, s_j)}{\partial w_k} = \frac{\partial d^s(b_{(p(b_k),j)}, s_j)}{\partial slew(b_{(p(b_k),j)})} \frac{\partial slew(b_{(p(b_k),j)})}{\partial w_k} \quad (17)$$

Therefore when $s_j \in FC(p(b_k))$ and $s_j \notin FC(b_k)$, $\frac{\partial d_j}{\partial w_k}$ can be expressed as follows:

$$\frac{\partial d_j}{\partial w_k} = \frac{\partial d^b(p(b_k), b_{(p(b_k),j)})}{\partial w_k} + \frac{\partial d^s(b_{(p(b_k),j)}, s_j)}{\partial slew(b_{(p(b_k),j)})} \frac{\partial slew(b_{(p(b_k),j)})}{\partial w_k} \quad (18)$$

4.2 The Two-pass Procedure

Based on Eqn.(14), Eqn.(16), and Eqn.(18), we design our sensitivity calculation algorithm as a two-pass procedure.

4.2.1 Pass One

In the first pass, we traverse all the clock buffers from the bottom. For each buffer b_k, we compute $\frac{\partial d^s(b_k, s_j)}{\partial slew(b_k)}$, *i.e.*, the sensitivity of delay from b_k's input node to every $s_j \in FC(b_k)$, with respect to the slew rate of clock signal at b_k's input node. A sub-tree ST_k for b_k is defined as shown in Figure 3. The root of ST_k is b_k's input node, the leaf of ST_k is either a clock sink or the output node of a child buffer in $C(b_k)$.

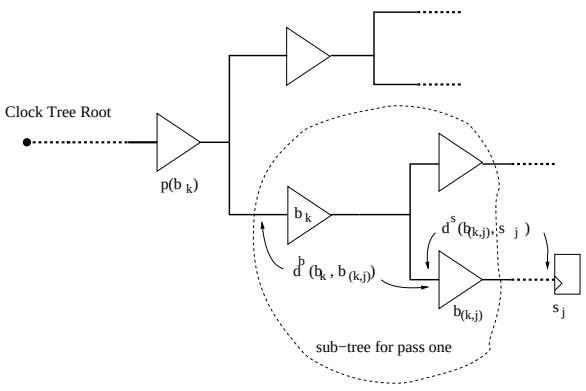

Figure 3: Pass one

In Figure 3, we observe that $d^s(b_k, s_j)$ can be decomposed into two parts:

$$d^s(b_k, s_j) = d^b(b_k, b_{(k,j)}) + d^s(b_{(k,j)}, s_j) \quad (19)$$

The change of $slew(b_k)$ affects $slew(b_{(k,j)})$, which in turn affects $d^s(b_{(k,j)}, s_j)$. Then we apply the chain rule of derivative and get:

$$\frac{\partial d^s(b_{(k,j)}, s_j)}{\partial slew(b_k)} = \frac{\partial d^s(b_{(k,j)}, s_j)}{\partial slew(b_{(k,j)})} \frac{\partial slew(b_{(k,j)})}{\partial slew(b_k)} \quad (20)$$

After some manipulations:

$$\frac{\partial d^s(b_k, s_j)}{\partial slew(b_k)} = \frac{\partial d^b(b_k, b_{(k,j)})}{\partial slew(b_k)} + \frac{\partial d^s(b_{(k,j)}, s_j)}{\partial slew(b_{(k,j)})} \frac{\partial slew(b_{(k,j)})}{\partial slew(b_k)} \quad (21)$$

In Eqn.(21), $\frac{\partial d^s(b_{(k,j)}, s_j)}{\partial slew(b_{(k,j)})}$ has been computed previously because we visit $b_{(k,j)}$ before b_k. $\frac{\partial d^b(b_k, b_{(k,j)})}{\partial slew(b_k)}$ and $\frac{\partial slew(b_{(k,j)})}{\partial slew(b_k)}$ can be computed by applying time-domain analysis to the sub-tree ST_k. Specifically, we apply a clock signal at the input node of b_k and simulate ST_k twice, first for the nominal value of $slew(b_k)$, and the second time for the value of $slew(b_k)$ with a small increment $\Delta slew(b_k)$. We are using the direct method to calculate sensitivity, and the adjoint network method would work just as well.

4.2.2 Pass Two

In the second pass, we traverse all the clock buffers again, but the order here is not important. For every b_k and every $s_j \in FC(p(b_k))$, we compute $\frac{\partial d_j}{\partial w_k}$ based on Eqn.(16) and Eqn.(18). Again, we define a sub-tree ST_k for b_k as shown in Figure 4. Please note that the sub-tree definition in pass two is different from that in pass one. Here the root of ST_k is the input node of $p(b_k)$, the leaf of ST_k is either a clock sink or the output node of a buffer that is a direct child of either b_k or $p(b_k)$.

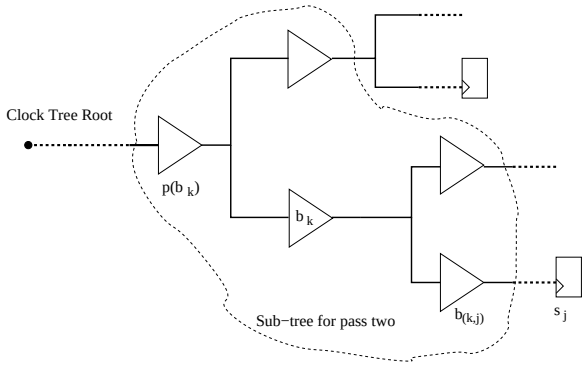

Figure 4: Pass two

As previously mentioned, for a clock sink $s_j \in FC(p(b_k))$, there are two possibilities: $s_j \in FC(b_k)$ or $s_j \notin FC(b_k)$. The formulas for $\frac{\partial d_j}{\partial w_k}$ are given by Eqn.(16) and Eqn.(18), respectively. In either case, the values of $\frac{\partial d^b(p(b_k), b_{(k,j)})}{\partial w_k}$, $\frac{\partial slew(b_{(k,j)})}{\partial w_k}$, $\frac{\partial d^b(p(b_k), b_{(p(b_k),j)})}{\partial w_k}$ and $\frac{\partial slew(b_{(p(b_k),j)})}{\partial w_k}$ can be computed by applying time-domain analysis to the sub-tree ST_k. In addition, the values of $\frac{\partial d^s(b_{(k,j)}, s_j)}{\partial slew(b_{(k,j)})}$ and $\frac{\partial d^s(b_{(p(b_k),j)}, s_j)}{\partial slew(b_{(p(b_k),j)})}$

have already been obtained in the first pass. In this way, $\frac{\partial d_j}{\partial w_k}$ is finally computed.

The above sensitivity calculation algorithm has a linear complexity with respect to the number of clock buffers. In the two-pass procedure, we visit each clock buffer twice. For each visit, we perform time-domain analysis for the sub-tree locally defined around this buffer. Because the scales of all these sub-trees are the same, no matter how large the whole clock tree might be, the complexity of simulating a sub-tree can be considered constant. Therefore, the complexity of computing delay sensitivities with respect to buffer sizes is $O(K)$.

5. POWER SUPPLY VARIATION

Power supply variation (noise) is one of the main causes of the clock skew. The existing clock network design techniques don't consider power supply noise. They are usually applied at the early design stages, when it is difficult to estimate the power supply variation. However, at the later design stages, when the placement of logic blocks, global routing, and pre-synthesis of clock tree have been finished, the power supply variations can be estimated by simulating the full chip. At this point, we are in a position to address the impact of power supply variation on clock timing.

We assume that in addition to the clock tree, we are also given the power supply network. The power grid is modeled as a linear, passive, time-invariant network, consisting of resistive, capacitive and inductive elements. Power sources are modeled as ideal voltage sources. The switching currents drawn by combinational logic blocks are modeled as time-varying piece-wise-linear waveforms obtained using off-line simulation and waveform compression techniques [8].

A common technique applied in the analysis of on-chip power networks is to separate the linear and non-linear components of the problem and analyze them individually. The clock skew verification methodology [9] proposed by Saleh, *et al.* falls into this category of techniques and therefore requires an iterative analysis of each network individually. We apply the same idea when we consider the impact of power supply noise in our optimization problem. We first assume there is no power supply voltage variation in the clock tree and we simulate the initial clock tree to obtain the switching current waveforms drawn from the power supply network. Then the switching current information and the current waveforms of logic blocks are fed into a power network analysis tool to obtain the power supply voltage waveforms at the Vdd pin of every clock buffer. With the updated power supply voltage waveforms, we use the proposed technique to size the clock buffers. Specifically, when we calculate sensitivities using time-domain analysis, instead of assuming an ideal power supply, a voltage source with the updated time-varying waveform is attached to the Vdd pin of the corresponding clock buffer. After sizing, clock tree simulation is carried out again to update switching current waveforms, which again will be fed into the power network analysis tool to update voltage waveforms, and so on. This iterative process will end when the simulations converge.

6. EXPERIMENTAL RESULTS

We have developed our prototype tool based on the proposed technique. Experiments are carried out for five test cases on a P4 2.4GHz PC running Linux. The linear programs are solved using the IBM optimization package [6]. The time-domain analysis is performed using Hspice. The test circuits have been implemented in $0.18\mu m$ technology, and the clock period is 5ns.

We first compare our useful-skew buffer sizing (USBS) approach against the previous bounded-skew buffer sizing (BSBS) technique in [15], which uses Elmore delay and simple RC buffer models. To construct the initial buffered clock trees, we first generate and embed the tree topology using the DME algorithm [2], then we follow the buffer insertion scheme described in [15] to insert the clock buffers. The results are summarized in Table 1. Columns 1 through 4 give the circuit name, the number of clock sinks, the number of clock buffers, and the number of general skew constraints. Column 5 lists the maximum global skew bound for the BSBS such that all general skew constraints are satisfied if this global skew bound is met by the BSBS. Column 6, 7, and 8 give the results for the BSBS obtained from simulating the clock tree using Hspice and show the actual maximal delay, actual maximal skew and the number of skew constraints violated. The power dissipations for the BSBS are normalized to 1. Columns 9 through 12 show the clock power dissipation, actual maximal delay, number of SLP iteration and CPU time, for the proposed USBS technique. From Table 1, we make the following observations:

1) The USBS technique can effectively reduce the clock power dissipation on average by 22.2% compared to the previous BSBS approach. This is because our technique combines clock skew scheduling and buffer sizing into a linear formulation, which allows us to explore the flexibility of general skew constraints compared to bounded skew constraints.

2) The actual maximal skews of the clock trees resulting from the BSBS method all exceed the given skew bounds and there are many skew constraints violations. This is because the previous BSBS technique is based on the analytical delay model with limited accuracy. On the other hand, there are no skew constraint violations with the clock trees produced by our USBS technique, because the USBS technique applies efficient time-domain analysis and provides very accurate delay and skew values.

3) The SLP-based technique converges very quickly, usually in fewer than 6 iterations. Figure 5 shows the average CPU times per iteration with respect to clock buffer numbers, which suggests a good scalability of our technique.

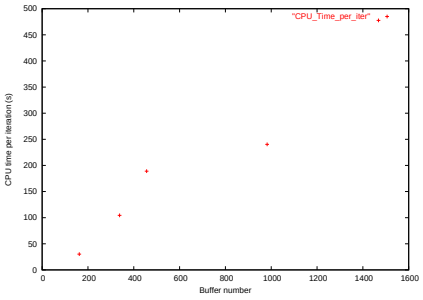

Figure 5: CPU Time per Iteration vs. Buffer Number

In the second experiment, we apply our buffer sizing technique to the problem of clock skew minimization under power supply variations. The initial clock trees have zero skew un-

Circuit	Sink Num.	Buffer Num.	Skew Cnstr. Num.	BSBS				USBS			
				Skew Bound (ps)	Actual Max. Delay (ns)	Actual Max. Skew (ps)	Cnstr. Vio. Num.	Clock Power	Actual Max. Delay (ns)	SLP Iter. Num.	CPU Time (s)
r1	267	162	289	130	1.212	178	28	0.91	1.191	4	121
r2	598	338	623	85	1.689	197	135	0.72	1.674	5	522
r3	862	456	847	165	1.936	248	253	0.63	1.894	3	567
r4	1903	982	1876	110	1.901	324	479	0.78	1.867	6	1442
r5	3101	1468	3357	125	2.027	369	501	0.85	2.012	5	2388

Table 1: BSBS vs. USBS

Circuit	Before Optimization					After Optimization		
	Elmore Delay(ns)	Max. Delay w/o Noise (ns)	Skew w/o Noise (ps)	Max. delay with Noise (ns)	Skew with Noise (ps)	Max Delay(ns)	Skew (ps)	Buffer Area Increase
r1	1.185	1.448	202	1.552	291	1.188	25	7.21%
r2	1.335	1.675	471	1.873	563	1.294	19	9.52%
r3	1.772	2.295	485	2.415	614	1.738	56	6.70%
r4	1.540	1.940	457	2.131	587	1.494	42	8.22%
r5	1.834	2.363	499	2.498	632	1.818	87	10.07%

Table 2: Skew Minimization under Power Supply Variation

der Elmore delay without considering power supply noise. The results are summarized in Table 2. Columns 3 and 4 show the actual maximal delay and skew without power supply noise before optimization. Columns 5 and 6 show the actual maximal delay and skew with power supply noise before optimization. The optimization results determined by the divide-and-conquer approach are shown in columns 7 to 9. The maximal voltage drop is about 5% of power supply voltage. From Table 2, we can see that the inaccuracy of the Elmore delay model and the presence of power supply noise may cause large skews, as we observed in simulation. The proposed technique can efficiently reduce clock skew, with insignificant area overhead.

7. CONCLUSIONS

In this paper, we have formulated the problem of buffer sizing for clock power minimization subject to general skew constraints, and presented a novel SLP-based approach. We have shown that by incorporating clock skew scheduling and buffer sizing, we can effectively reduce the clock power dissipation. Our technique is also capable of taking practical design issues such as process variations and power supply noise into account, and providing very accurate delay and skew results.

Acknowledgment

This work is supported in part by the SRC grant #1069 and in part by the California MICRO program through IBM.

8. REFERENCES

[1] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 1995.

[2] J. Cong, A. B. Kahng, C. K. Koh, and C. W. A. Tsao. Bounded-skew clock and steiner routing. *ACM Trans. on Design Automation of Electronic Systems*, 4(1):1–46, Jan 1999.

[3] W. C. Elmore. The transient response of damped linear networks with particular regard to wide-band amplifiers. *Journal of Applied Physics*, pages 55–63, 1948.

[4] J. P. Fishburn. Clock skew optimization. *IEEE Trans. on Computer*, pages 39(7):945–951, July 1990.

[5] E. G. Friedman. Clock distribution networks in synchronous digital integrated circuits. In *Proceedings of IEEE*, volume 89, pages 665–692, 2001.

[6] http://www.ibm.com/software/data/bi/osl. Ibm optimization solutions and library.

[7] L. W. Nagel. Spice2: a computer program to simulate semiconductor circuits. Technical Report ERL-M520, UC Berkeley, May 1975.

[8] R.Chaudhry, D. Blaauw, R. Panda, and T. Edwards. Current signature compression for ir-drop analysis. In *Proc. Design Automation Conf.*, 2000.

[9] R. Saleh, S. Z. Hussain, S. Rochel, and D. Overhauser. Clock skew verification in the presence of ir-drop in the power distribution network. *IEEE Trans. Computer-Aided Design*, 19:635–644, 2000.

[10] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Graph algorithms for clock schedules optimization. In *Proc. Int. Conf. on Computer Aided Design*, pages 132–136, 1992.

[11] C.-W. A. Tsao and C.-K. Koh. Ust/dme: A clock tree router for general skew constraints. *ACM(TODATES)*, pages 7:359–379, 2002.

[12] R. S. Tsay. An exact zero-skew clock routing algorithm. *IEEE Trans. Computer-Aided Design*, 12:242–249, 1993.

[13] P. Vuillod, L. Benini, A. Bogliolo, and G. D. Micheli. Clock skew optimization for peak current reduction. In *Proc. Int'l. Symp. on Low Power Electronics and Design*, pages 265–270, Aug. 1996.

[14] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: a Systems Perspective*. Addison-Wesley, MA, 2 edition, 1993.

[15] J. G. Xi and W. W.-M. Dai. Buffer insertion and sizing under process variations for low power clock distribution. In *Proc. Design Automation Conf.*, 1995.

[16] J. G. Xi and W. W.-M. Dai. Useful-skew clock routing with gate sizing for low power design. *Journal of VLSI Signal Processing Systems*, pages 16(2/3):163–170, 1997.

[17] Q. Zhu and W. W. M. Dai. High-speed clock network sizing optimization based on distributed rc and lossy rlc interconnect models. *IEEE Trans. Computer-Aided Design*, 15:1106–1118, 1996.