

Clock Skew Optimization Via Wiresizing For Timing Sign-off Covering All Process Corners*

Sari Onaissi, Khaled R. Heloue, Farid N. Najm
Department of ECE, University of Toronto, Toronto, Ontario, Canada
{sari, khaled, najm}@eecg.utoronto.ca

ABSTRACT

Manufacturing process variability impacts the performance of synchronous logic circuits by means of its effect on both clock network and functional block delays. Typically, variability in clock networks is either handled early in the design flow by assigning margins to clock network delays, or at a later stage through post-processing steps that *only* focus on achieving minimal skew, without regard to functional block variability. In this work, we present a technique that alters clock network lines so that the circuit meets its timing constraints at all process corners. This is done near the end of the design flow while considering delay variability in *both* the clock network and the functional blocks. Our method operates at the *physical level* and provides designers with the required changes in clock network line widths and/or lengths. This can be formulated as a Linear Programming (LP) problem, and thus can be solved efficiently. Empirical results for a set of ISCAS-89 benchmark circuits show that our approach can considerably reduce the effect of process variations on circuit performance.

Categories and Subject Descriptors

B.7.2 [Hardware]: Integrated Circuits—*Design Aids*

General Terms

Algorithms, performance, verification, reliability

Keywords

Clock skew optimization, variability, sign-off, parameterized timing analysis, wiresizing

1. INTRODUCTION

The performance of a synchronous logic circuit depends on both circuit delays and clock skews introduced by the clock distribution network. Clock skew is defined as the difference between the arrival times of the clock signal at different clocked circuit elements. Typically, clock networks are designed to minimize skew between the various clocked elements, where the objective in this case is to achieve a *zero-skew clock tree*, as in [1] and [2]. On the other hand, the more aggressive techniques of *skew optimization* or *clock scheduling* (e.g. [3], [4], and [5]) actually assign skews to different clocked elements in order to obtain better performance. These usually formulate optimization problems to find clock network path delays that improve circuit performance. However, in practice, this requires interaction among different stages of the design flow [6] and extends to beyond finding clock network delay values.

With the scaling of VLSI technology, the effect of manufacturing process variations on circuit and clock network delays has increased. Typically the effect of this on clock networks is seen in the form of *unintended skew*, which degrades circuit performance. In the case of zero-skew trees the focus has been on minimizing the maximum skew resulting from process variations. This is typically

*This project was supported in part by Intel Corp.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC2009, July 26–31, 2009, San Francisco, California, USA.

Copyright 2009 ACM 978-1-60558-497-3-6/08/0006 ...\$5.00.

done by wire sizing and/or introduction of buffers (e.g. [7] and [8]). Such approaches try to “cap” the maximum skew expected as a result of process variability. However, these approaches deal with variability in the clock network without regard to that in the functional blocks, and are focused on achieving zero-skew rather than timing closure. On the other hand, skew optimization techniques usually deal with variability by incorporating margins [3], or permissible ranges [5], into their optimization formulation. However, this is done at an early stage in the design flow when accurate variability information of the clock network and functional blocks is typically not available and is thus problematic.

In this work we present a linear programming (LP) formulation in order to reduce the effect of process variability on circuit performance. Our approach specifies required changes at the physical level in clock network wire widths or lengths rather than required clock network delay values. It is applied near the end of the design flow as a post-processing step to account for process variability when accurate *variational* clock network and functional path delays are available. As stated earlier, modifying clock line widths or lengths as a post-processing step has been proposed to minimize the unintended skew of zero-skew clock trees. However, in this work we formulate an LP that varies skews to try to meet circuit timing constraints at all process corners rather than try to eliminate skew.

2. PRELIMINARIES

In this section, we first present the process parameter model assumed in our work. Then, we present some definitions and preliminary concepts that are used throughout the paper.

2.1 Variability Model

In our approach, all logic cell and interconnect delays are modeled as linear functions of normalized process parameters, whose values vary between -1 and $+1$. Hence the delay of a logic cell or of an interconnect RC -tree (also referred to as an interconnect structure) can be written as an affine function of these process parameters. Because the delays of individual paths, in both clock networks and functional blocks, are sums of gate and interconnect delays, they also become such affine functions of the process parameters. Let the number of process parameters under consideration be p . Thus, a timing quantity t , representing a timing arc, interconnect, or path delay, can be written as follows:

$$t = \bar{t} + \sum_{l=1}^p \delta_l X_l \quad (1)$$

where $X = (X_1, \dots, X_p)$ is the set of normalized process parameters, \bar{t} is the *nominal value* of t , and δ_l is its *sensitivity* to process parameter X_l . Such an affine function represents a *hyperplane* in $(p+1)$ -dimensional space, and will often be referred to as, simply, a hyperplane. In this work, it is often required to find the maximum or the minimum value of a hyperplane over the process parameter space. It is trivial to see that the maximum value of t over all process corners is:

$$t_{max} = \bar{t} + \sum_{l=1}^p |\delta_l| \quad (2)$$

which is equivalent to computing t at the process corner $\hat{X} = (\hat{X}_1, \dots, \hat{X}_p)$, such that $\hat{X}_l = +1$ if $\delta_l \geq 0$ and $\hat{X}_l = -1$ otherwise. Minimizing t involves a very similar operation where the hyperplane is instead computed at $\hat{x} = (\hat{x}_1, \dots, \hat{x}_p)$, where $\hat{x}_l = -1$ if $\delta_l \geq 0$ and $\hat{x}_l = +1$ otherwise.

2.2 Circuit Model

We focus on synchronous sequential circuits with edge-triggered registers, but the work can be extended to circuits with level-sensitive latches. In such circuits, every combinational logic block receives its inputs from its *input registers* and stores its outputs in its *output registers*. We denote by S the set of combinational blocks in a given circuit. For example, Fig. 1, shows a simple sequential circuit, where in this case $S = \{s_1, s_2\}$. A clock signal is connected to all the registers of a sequential circuit by means of a clock network that consists of buffers and interconnect. Such a clock network can be designed to minimize skew between the clock signals at the various registers, or it can be designed such that it intentionally introduces skew at carefully chosen registers in order to help meet timing constraints. In our approach, we refer to the clock signal received at a register as the register's *clock phase*, and we call the delay of the path from the clock source to the register its *clock phase arrival time*. In general, for a sequential circuit with m registers, the clock phase arrival times are referred to as $r_q, 1 \leq q \leq m$. For a combinational logic block $s \in S$, D_s^{ij} refers to the largest delay between the input register controlled by clock phase i and the output register controlled by clock phase j . The term d_s^{ij} is used to refer to the smallest such delay. For example, in Fig. 1, D_1^{24} is the largest delay between reg₂ and reg₄ through s_1 whereas d_1^{24} is the smallest such delay.

The clock phase arrival times r_q of a sequential circuit are path delays and hence, in our formulation, are hyperplanes in the set of process parameters X . Consider the set P_s^{ij} of all paths through the combinational logic block s between the source register controlled by clock phase i and the sink register controlled by clock phase j . The delays of these paths are, each, a hyperplane in the process parameters, for which D_s^{ij} is their “max” and d_s^{ij} is their “min”. Being the maximum and minimum of a set of hyperplanes, it follows that D_s^{ij} and d_s^{ij} are, in general, piecewise planar surfaces and *not* hyperplanes. The notion of piecewise planar delays has been used in the context of parameterized static timing analysis (PSTA) and there are existing works (e.g., [9] and [10]) on how these can be found and represented. Typically, such a surface is represented by a set or a “list” of hyperplanes, each of which corresponds to a particular path. Thus D_s^{ij} is the set of hyperplane delays of “potentially longest paths” and d_s^{ij} is the set of hyperplane delays of “potentially shortest paths”. However, in our method, it is also possible to approximate D_s^{ij} and d_s^{ij} by single hyperplanes, obtained by finding hyperplane bounds for these two sets, using the approach in [11]. That is, D_s^{ij} would be replaced by a hyperplane approximation of the “max” of the delays of the “potentially longest paths”, and similarly d_s^{ij} would be replaced by a hyperplane approximation of the “min” of the “potentially shortest paths” delays. These approximations are conservative, i.e., they never underestimate the true maximum delay or overestimate the true minimum delay for any point in the process space.

3. BACKGROUND

In our approach, we modify the delays of the clock network in order for the circuit to meet its timing constraints in the presence of process variations. This would be applicable near the end of the design flow, as part of physical design while trying to achieve timing sign-off, by modifying the clock network line widths and/or lengths. In this section, we first present a short overview of standard clock tree synthesis and a brief description of how clock skew optimization could be used to improve the performance of a circuit in a typical design flow. We then discuss the effect of process variability on circuit performance and how a circuit's timing constraints can be verified under variability. This will provide some background understanding of the problem and set the stage for the presentation of our proposed clock network optimization method.

3.1 Clock Tree Synthesis

In order to synthesize a clock network, a user can specify certain parameters such as maximum skew, maximum and minimum

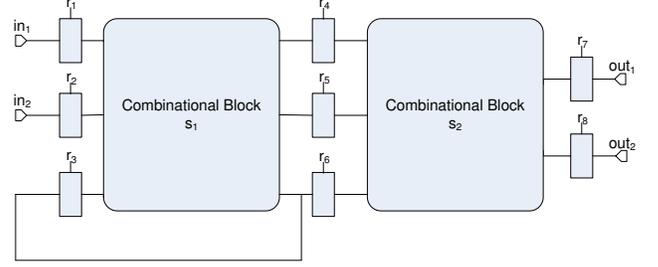


Figure 1: A Simple Sequential Circuit

clock phase arrival times, and a host of other parameters. These are then used to generate a clock network consisting of buffers and interconnect, where the objective would be to have a “balanced” clock tree. That is, designers usually strive to minimize skews between different leaves of the clock tree.

In order to achieve circuits with higher frequencies, or higher slacks, a user can use skew optimization techniques such as those presented in [3] and [4] to find an “optimal” set of clock phase arrival times. These techniques lead to a set of *nominal* clock phase arrival times or a *clock schedule*, which are then translated into a physical structure consisting of buffers and interconnect. The method in [3] is an LP formulation that provides the clock phase delays that maximize the clock frequency for circuits with edge-triggered registers, as follows. Let $\bar{r}_q, 1 \leq q \leq m$, be the *nominal* clock phase arrival times, and let \bar{D}_s^{ij} and \bar{d}_s^{ij} be, respectively, the largest and smallest nominal delays through the combinational logic block s between the source register with clock phase i and sink register with clock phase j . The minimum acceptable clock period T can be found using the following LP [3]:

$$\begin{aligned} & \text{minimize: } T \\ & \text{Subject to: } \forall s \in S, \forall i, j \in 1, \dots, m \end{aligned} \quad (3)$$

$$\begin{aligned} \bar{D}_s^{ij} + \bar{r}_i - \bar{r}_j + t_{setup}^j &\leq T \\ \bar{d}_s^{ij} + \bar{r}_i - \bar{r}_j - t_{hold}^j &\geq 0 \end{aligned}$$

To allow this general formulation, the convention is adopted that, if logic block s does not contain a combinational path between registers controlled by clock phases i and j , then $\bar{D}_s^{ij} = -\infty$ and $\bar{d}_s^{ij} = +\infty$. For a required clock period of T_c , if $T \leq T_c$ then the clock network which achieves the values of $\bar{r}_q, 1 \leq q \leq m$ is generated, placed, and routed. However, achieving a clock network that can produce such arrival times is more complex than achieving a minimal skew tree, and clock skew optimization remains an active research area [6]. In any case our method can be used irrespective of whether clock skew optimization is used to generate the clock network or not.

3.2 Verification for Variability

Using the layout of the circuit resulting from either the standard or skew optimization *nominal point* analyses as a starting point, along with characterized cell and interconnect process sensitivities, variational path delays of the circuit can now be extracted. For a register with nominal clock phase arrival time \bar{r}_q , its actual r_q under variability is a hyperplane, given by:

$$r_q = \bar{r}_q + \sum_{l=1}^p \lambda_l^{(q)} X_l \quad (4)$$

The dependence of combinational path delays and clock phase arrival times on process parameters may cause some timing constraints to fail for some process settings. In order for the circuit to meet the timing constraints under variability, then, for every logic block s , the following inequalities should be satisfied over the whole process parameter space for all clock phases i and j :

$$\begin{aligned} D_s^{ij} + r_i - r_j + t_{setup}^j &\leq T_c \\ d_s^{ij} + r_i - r_j - t_{hold}^j &\geq 0 \end{aligned} \quad (5)$$

which is equivalent to saying that:

$$\begin{aligned} \max_X (D_s^{ij} + r_i - r_j) + t_{setup}^j &\leq T_c \\ \min_X (d_s^{ij} + r_i - r_j) - t_{hold}^j &\geq 0 \end{aligned} \quad (6)$$

Note that here we make the simplifying assumption that t_{setup}^j and t_{hold}^j are constant, rather than dependent on process parameters. Therefore, these are not included in the max and min expressions. However this assumption is by no means necessary, and doesn't affect the applicability of our approach. Performing the verification in (6) can be done as follows. Let us first consider the case where bounding hyperplane approximations are used for D_s^{ij} and d_s^{ij} , as discussed above, so that D_s^{ij} and d_s^{ij} are hyperplanes, given by:

$$D_s^{ij} = \bar{D} + \sum_{l=1}^p \Phi_l X_l, \text{ and } d_s^{ij} = \bar{d} + \sum_{l=1}^p \phi_l X_l \quad (7)$$

Writing r_i and r_j as in (4), we can now write (6) as:

$$\begin{aligned} \max_X \left(\bar{D} + \bar{r}_i - \bar{r}_j + \sum_{l=1}^p (\Phi_l + \lambda_l^{(i)} - \lambda_l^{(j)}) X_l \right) + t_{setup}^j &\leq T_c \\ \min_X \left(\bar{d} + \bar{r}_i - \bar{r}_j + \sum_{l=1}^p (\phi_l + \lambda_l^{(i)} - \lambda_l^{(j)}) X_l \right) - t_{hold}^j &\geq 0 \end{aligned} \quad (8)$$

Note that the expressions inside the ‘‘max’’ and the ‘‘min’’ operations are hyperplanes. Finding the maximum or minimum value of a hyperplane over the process space is a straight-forward operation that is performed by computing the value of hyperplane at a carefully chosen corner, as described earlier in (2). For the constraints in (8), we use the term ‘‘worst-case’’ corner to refer to the process corner used in the verification of the constraint. Thus for a setup constraint the ‘‘worst-case’’ corner is the process corner used to maximize its hyperplane, whereas for a hold-time constraint this term refers to the process corner that minimizes its hyperplane.

Now consider the case when D_s^{ij} and d_s^{ij} are written as piecewise planar functions. For example, suppose that D_s^{ij} is the piecewise planar surface formed by taking the ‘‘max’’ of the following hyperplanes:

$$\begin{aligned} D^{(1)} &= \bar{D}^{(1)} + \sum_{l=1}^p \Phi_l^{(1)} X_l \\ D^{(2)} &= \bar{D}^{(2)} + \sum_{l=1}^p \Phi_l^{(2)} X_l \\ &\vdots \\ D^{(u)} &= \bar{D}^{(u)} + \sum_{l=1}^p \Phi_l^{(u)} X_l \end{aligned} \quad (9)$$

Then, in order to verify the setup constraint of D_s^{ij} we have to verify the following constraints:

$$\begin{aligned} \max_X \left(\bar{D}^{(1)} + \bar{r}_i - \bar{r}_j + \sum_{l=1}^p (\Phi_l^{(1)} + \lambda_l^{(i)} - \lambda_l^{(j)}) X_l \right) + t_{setup}^j &\leq T_c \\ \max_X \left(\bar{D}^{(2)} + \bar{r}_i - \bar{r}_j + \sum_{l=1}^p (\Phi_l^{(2)} + \lambda_l^{(i)} - \lambda_l^{(j)}) X_l \right) + t_{setup}^j &\leq T_c \\ &\vdots \\ \max_X \left(\bar{D}^{(u)} + \bar{r}_i - \bar{r}_j + \sum_{l=1}^p (\Phi_l^{(u)} + \lambda_l^{(i)} - \lambda_l^{(j)}) X_l \right) + t_{setup}^j &\leq T_c \end{aligned} \quad (10)$$

These constraints are verified by writing each at its ‘‘worst-case’’ corner as is done for the setup constraint of (8). However, note that ‘‘worst-case’’ corners for these constraints can be different. Let T_v be the smallest clock period for which all the setup constraints of a circuit are satisfied under process variability. This value can be found by evaluating the left-hand side expressions of all setup constraints, such as (10), and choosing the largest value. We call $M_v = T_v - T_c$ the required timing margin; this is the margin that has to be added to T_c so that the circuit can operate correctly in the presence of process variability. The same approach is taken for d_s^{ij} , where its hold-time constraint is also written for each of the hyperplanes that form its piecewise planar surface as in (10) and may be verified in a similar manner.

4. OPTIMIZATION

If the verification process described above reveals that the circuit does not meet its timing constraints for a required T_c under *all* process settings, then an optimization problem can be formulated to modify the clock network so that the required timing margin is minimized, thus possibly allowing the timing constraints to be satisfied at all process corners, as follows.

A set of physical parameters of the clock tree, such as wire lengths and/or widths, are chosen as *variables* to be optimized. By varying these optimization variables, the different clock phase delays can be modified to minimize the required timing margin. However, before we explain how this is done, we will first examine how varying the physical parameters of an interconnect *RC*-structure of a clock tree affects its clock phase arrival times.

4.1 Physical Parameter to Clock Phase Delay

Consider Fig. 2, which shows a generic segment of the path between a clock source and a register, consisting of the interconnect *RC*-structure con_k , its ‘‘input buffer’’ buf_k , and one of its ‘‘output buffers’’ buf_{k+1} . We are interested in the effect that varying a physical parameter of con_k would have on the clock phase arrival time r seen in Fig. 2. We will use variation in wire width as the physical parameter of interest in our discussion, however all of the arguments that follow can be made for a variation in wire length as well. Let w_k be the wire width of con_k , and Δw_k be a change in w_k . A variation in w_k would have an immediate effect on the delays of the three ‘‘local’’ circuit elements, con_k , buf_k , and buf_{k+1} . First, the introduction of Δw_k would vary the delay of con_k by Δd_{con_k} . Second, the effective load capacitance seen by buf_k would vary as a result of Δw_k , thus leading to a variation Δd_{buf_k} in its delay, d_{buf_k} . In addition to the variation in the delay of buf_k , the change in its effective load capacitance also leads to a variation in its output signal slew. This in turn varies the delay of buf_{k+1} by $\Delta d_{buf_{k+1}}$. One could argue that this variation in output signal slew of buf_k would also lead to a similar effect in buf_{k+1} , which would then lead to the same effect in buffers further downstream. This would extend the effect of Δw_k to ‘‘non-local’’ circuit elements. However, the impact on the delays of these elements is small in practice, and it can be safely ignored. Thus, the delay variations in the three ‘‘local’’ circuit elements seen in Fig. 2 would account for most of the variation seen in r . This variation, Δr , can now be written as:

$$\Delta r = \Delta d_{buf_k} + \Delta d_{con_k} + \Delta d_{buf_{k+1}} \quad (11)$$

In our approach, a linear model is used to approximate the dependence of delay variations of circuit elements on Δw_k . Thus, a first order Taylor series approximation of d_{buf_k} , d_{con_k} , and $d_{buf_{k+1}}$, around the original value of w_k is used to write:

$$\begin{aligned} \Delta d_{buf_k} &\approx \frac{\partial d_{buf_k}}{\partial w_k} \Delta w_k, \quad \Delta d_{con_k} \approx \frac{\partial d_{con_k}}{\partial w_k} \Delta w_k \\ \Delta d_{buf_{k+1}} &\approx \frac{\partial d_{buf_{k+1}}}{\partial w_k} \Delta w_k \end{aligned} \quad (12)$$

and leads to a linear dependence of Δr on Δw_k :

$$\Delta r \approx \left(\frac{\partial d_{buf_k}}{\partial w_k} + \frac{\partial d_{con_k}}{\partial w_k} + \frac{\partial d_{buf_{k+1}}}{\partial w_k} \right) \Delta w_k = \frac{\partial r}{\partial w_k} \Delta w_k \quad (13)$$

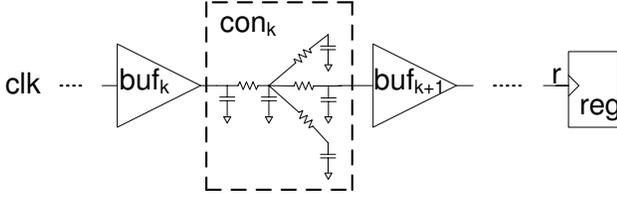


Figure 2: Section of a clock tree path

However, r , d_{buf_k} , d_{con_k} , and $d_{\text{buf}_{k+1}}$ are also hyperplanes in the set of process parameters X . Let r be written as seen in (4), and d_{buf_k} , d_{con_k} , and $d_{\text{buf}_{k+1}}$ as follows:

$$d_{\text{buf}_k} = \alpha_0 + \sum_{l=1}^p \alpha_l X_l, \quad d_{\text{con}_k} = \beta_0 + \sum_{l=1}^p \beta_l X_l \quad (14)$$

$$d_{\text{buf}_{k+1}} = \gamma_0 + \sum_{l=1}^p \gamma_l X_l$$

Then, using (4), we write:

$$\frac{\partial r}{\partial w_k} = \frac{\partial \bar{r}}{\partial w_k} + \sum_{l=1}^p \frac{\partial \lambda_l}{\partial w_k} X_l \quad (15)$$

and, using (14) and (13), we deduce that:

$$\frac{\partial \bar{r}}{\partial w_k} = \frac{\partial \alpha_0}{\partial w_k} + \frac{\partial \beta_0}{\partial w_k} + \frac{\partial \gamma_0}{\partial w_k}, \quad \text{and} \quad \frac{\partial \lambda_l}{\partial w_k} = \frac{\partial \alpha_l}{\partial w_k} + \frac{\partial \beta_l}{\partial w_k} + \frac{\partial \gamma_l}{\partial w_k} \quad (16)$$

where $1 \leq l \leq p$.

4.2 Modeling Multiple Variable Parameters

So far, we have presented an analysis of the effect of varying a single physical parameter on clock phase arrival time. We now describe how one can capture the effect of varying multiple physical parameters on clock phase arrival times. Without loss of generality, we choose to vary only the widths of the various interconnect structures of the clock network. Let W represent the vector of interconnect widths of the clock tree, and let ΔW be the vector of variations in these widths. Let the number of interconnect segments in the clock tree be n , and we write $\Delta W = (\Delta w_1, \Delta w_2, \dots, \Delta w_n)$, where each component represents the variation in an interconnect segment in the clock tree. For an arbitrary clock phase arrival time r , the introduction of ΔW leads to a change Δr :

$$\Delta r \triangleq r(W + \Delta W) - r(W) \quad (17)$$

Because Δr is the aggregate result of n “local” delay variations in the clock tree, the result of (13) is generalized to write:

$$\Delta r \approx \sum_{k=1}^n \frac{\partial r}{\partial w_k} \Delta w_k \quad (18)$$

Using (18) and (15) we can write:

$$\Delta r \approx \sum_{k=1}^n \frac{\partial \bar{r}}{\partial w_k} \Delta w_k + \sum_{l=1}^p \left(\sum_{k=1}^n \frac{\partial \lambda_l}{\partial w_k} \Delta w_k \right) X_l \quad (19)$$

4.3 Formulation of the Optimization Problem

The aim of the optimization step is to determine the required changes in interconnect wire widths, $\Delta W = (\Delta w_1, \Delta w_2, \dots, \Delta w_n)$, given allowable bounds $\Delta W_{\min} \leq \Delta W \leq \Delta W_{\max}$, so that the required timing margin, M , is minimized, given a required clock period T_c . One possible way to achieve this is to extend the work in [3] by finding an optimal assignment ΔW^* such that it minimizes $M = T - T_c$, where T is the clock period for which the

timing constraints are met under all process corners. This can be expressed as follows:

$$\text{minimize: } T - T_c \quad (20)$$

$$\text{Subject to: } \Delta W_{\min} \leq \Delta W \leq \Delta W_{\max}$$

$$\forall s \in S, \forall i, j \in 1, \dots, m$$

$$\max_X (D_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) + t_{\text{setup}}^j \leq T$$

$$\min_X (d_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) - t_{\text{hold}}^j \geq 0$$

Note that r_i , r_j , D_s^{ij} , and d_s^{ij} are extracted affine functions that depend solely on the process parameters X_l , $1 \leq l \leq p$, whereas the variations in clock phase arrival times, Δr_i and Δr_j , depend on both Δw_k and X_l , as shown in (19), and can be written as:

$$\Delta r_i = \sum_{k=1}^n \frac{\partial \bar{r}_i}{\partial w_k} \Delta w_k + \sum_{l=1}^p \left(\sum_{k=1}^n \frac{\partial \lambda_l^{(i)}}{\partial w_k} \Delta w_k \right) X_l$$

$$\triangleq \sum_{k=1}^n g_k^{(0)} \Delta w_k + \sum_{l=1}^p \left(\sum_{k=1}^n g_k^{(l)} \Delta w_k \right) X_l \quad (21)$$

$$\Delta r_j = \sum_{k=1}^n \frac{\partial \bar{r}_j}{\partial w_k} \Delta w_k + \sum_{l=1}^p \left(\sum_{k=1}^n \frac{\partial \lambda_l^{(j)}}{\partial w_k} \Delta w_k \right) X_l$$

$$\triangleq \sum_{k=1}^n h_k^{(0)} \Delta w_k + \sum_{l=1}^p \left(\sum_{k=1}^n h_k^{(l)} \Delta w_k \right) X_l \quad (22)$$

If the minimum value, M^* , of $T - T_c$ is such that $M^* \leq 0$, then the the timing constraints of the circuit can be met in the presence of process variability. Otherwise, the constraints simply cannot be satisfied given both the required clock T_c and the allowable ranges on ΔW . In that case, achieving sign-off will require different changes in the design, not only *tweaking* the interconnect wire widths, as we propose here. In any case, M^* corresponds to the smallest required timing margin achieved for $\Delta W^* \in [\Delta W_{\min}, \Delta W_{\max}]$, in order to meet timing constraints at all process corners.

In the interest of efficiency, we transform the problem (20) into a linear program (LP), by writing each of its constraints at a specific process corner, as follows. Consider the following constraint from (20):

$$\max_X (D_s^{ij} + r_i - r_j + \Delta r_i - \Delta r_j) + t_{\text{setup}}^j \leq T \quad (23)$$

Let us start with the case where D_s^{ij} is approximated by a single bounding hyperplane. In this case we can write:

$$D_s^{ij} + r_i - r_j \triangleq t = t_0 + \sum_{l=1}^p t_l X_l \quad (24)$$

Let $\hat{X}_s^{ij} = (\hat{X}_1, \dots, \hat{X}_p)$ be the process corner that maximizes the expression in (24). Note that this is the *pre-optimization* “worst-case” corner of (23), i.e., when $\Delta r_i = 0$ and $\Delta r_j = 0$. This corner is used to write (23) as:

$$(D_s^{ij} + r_i - r_j + \Delta r_i - \Delta r_j) \Big|_{\hat{X}_s^{ij}} + t_{\text{setup}}^j \leq T \quad (25)$$

where writing $D_s^{ij} + r_i - r_j$ at this corner gives a constant value K :

$$(D_s^{ij} + r_i - r_j) \Big|_{\hat{X}_s^{ij}} = t_0 + \sum_{l=1}^p t_l \hat{X}_l \triangleq K \quad (26)$$

and using (21), writing Δr_i at \hat{X}_s^{ij} gives:

$$\Delta r_i \Big|_{\hat{X}_s^{ij}} = \sum_{k=1}^n \left(g_k^{(0)} + g_k^{(1)} \hat{X}_1 + \dots + g_k^{(p)} \hat{X}_p \right) \Delta w_k$$

$$\triangleq \sum_{k=1}^n G_k \Delta w_k \quad (27)$$

and similarly, Δr_j can be written as:

$$\begin{aligned} \Delta r_j \Big|_{\hat{X}_s^{ij}} &= \sum_{k=1}^n \left(h_k^{(0)} + h_k^{(1)} \hat{X}_1 + \dots + h_k^{(p)} \hat{X}_p \right) \Delta w_k \\ &\triangleq \sum_{k=1}^n H_k \Delta w_k \end{aligned} \quad (28)$$

Thus writing (23) at \hat{X}_s^{ij} gives the following constraint:

$$K + \sum_{k=1}^n (G_k - H_k) \Delta w_k + t_{setup}^j \leq T \quad (29)$$

Note that this constraint is now linear in the optimization variables Δw_k , $1 \leq k \leq n$, and no longer has a “max” expression over the process parameters. The same transformation can be done for the rest of the “max” and “min” operations in all of the setup and hold time constraints, respectively, to give the following optimization problem:

$$\text{minimize: } T - T_c \quad (30)$$

$$\begin{aligned} \text{Subject to: } &\Delta W_{\min} \leq \Delta W \leq \Delta W_{\max} \\ &\forall s \in S, \forall i, j \in 1, \dots, m \end{aligned}$$

$$\begin{aligned} (D_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) \Big|_{\hat{X}_s^{ij}} + t_{setup}^j &\leq T \\ (d_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) \Big|_{\hat{x}_s^{ij}} - t_{hold}^j &\geq 0 \end{aligned}$$

All the constraints in (30) are now linear in the optimization variables, so that (30) is an LP which can be solved efficiently using commercial LP solvers.

Next, consider the case when D_s^{ij} and d_s^{ij} are expressed as piecewise planar functions. Let D_s^{ij} be the “max” of the set of hyperplanes in (9), then a constraint similar to (29) is written for each of $D^{(1)}, \dots, D^{(u)}$. This would result in an LP that is very similar to (30), but which has a larger number of constraints. However, since actual path delays are now used in the optimization rather than conservative estimates, this would result in a less pessimistic and more accurate minimum value of the required margin $T - T_c$.

In its present form, (30) performs an optimization where each constraint is written at its own pre-optimization “worst-case” corner, which was found during the verification step. For a particular constraint, say (23), transforming it to (25) and using the latter in (30) is meant to ensure that the post-optimization circuit would satisfy (23) for those process settings X_v such that:

$$(D_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) \Big|_{X_v} \leq (D_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) \Big|_{\hat{X}_s^{ij}} \quad (31)$$

However, there is no guarantee that \hat{X}_s^{ij} would be the “worst-case” corner for the post-optimization expression $(D_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j)$. The reason for this is that the introduction of Δr_i and Δr_j alters the process parameter sensitivities of the expression inside the “max” operation in (23), as can be seen from (21) and (22). Thus, in order for (30) to ensure that the timing margin found allows the circuit to pass timing at *all* process corners, we must make sure that possible changes in “worst-case” corners are accounted for in (30). In the next section, we propose a method that deals with this possibility, in a conservative fashion, while preserving the linearity of the optimization problem.

5. COVERING ALL CORNERS

In this section, we present a method that can be used to determine whether the post-optimization “worst-case” corner of a timing constraint in (20) could possibly be different from its pre-optimization “worst-case” corner. After that, we present our proposed steps to modify (30) if a constraint is found to be as such. The aim of this modification is to ensure that the solution of (30) guarantees that the circuit would pass its timing constraints for all process corners.

5.1 Possible Changes in Worst-Case Corners

Recall from Section 3.2 that the process corner that maximizes or minimizes an affine expression in the space of process parameters can be found by looking at the signs of its sensitivities to each of these parameters. Therefore, in order to determine whether the “worst-case” corner of an expression might change post-optimization, we need to find which are the sensitivities whose signs might change as a result of the optimization performed. One way of doing this is as follows. Consider the setup constraint in (23) and let $t = D_s^{ij} + r_i - r_j$ as in (24) and let t' be as follows:

$$t' = D_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j \quad (32)$$

For the given bounds on ΔW , ΔW_{\min} and ΔW_{\max} , we want to determine whether the sign of the sensitivity to a process parameter X_l , $1 \leq l \leq p$ in t' could have a sign opposite to its sensitivity in t . Using (21) and (22) we can write the sensitivity to X_l in t' , denoted by t'_l , as follows:

$$t'_l = t_l + \sum_{k=1}^n (g_k^{(l)} - h_k^{(l)}) \Delta w_k \quad (33)$$

where t_l is the sensitivity of t to parameter X_l , which is known from (26). In addition to determining whether t_l and t'_l are guaranteed to have the same sign or not, we also want to determine the largest absolute value that t'_l can assume, if it is possible for it to have a sign different than that of t_l . This is not necessarily the largest magnitude that t'_l can have when it has a sign opposite to that of t_l , but rather the largest absolute value it can have, irrespective of sign. Assume, without loss of generality, that t_l is negative. Determining whether t'_l can be positive is done as follows. For $1 \leq k \leq n$, if $(g_k^{(l)} - h_k^{(l)}) < 0$, then we set $\Delta w_k = w_k^{\min}$. On the other hand, if $(g_k^{(l)} - h_k^{(l)}) \geq 0$, then we set $\Delta w_k = w_k^{\max}$. The value of t'_l is then computed using these assigned values. If the computed t'_l is positive then it is “at risk” of a sign change, and this value is the largest positive value it can assume. In this case, the smallest negative value of t'_l is also computed in a similar way and is compared to its largest positive value to find its maximum absolute value which we call t_l^{\max} .

5.2 Accounting For Changing Corners

For a timing constraint in (20), once all the sensitivities that are “at risk” of having different post- and pre-optimization signs have been determined, we can proceed to modify (30), as follows. Let the constraint under consideration be the setup constraint shown in (23), and let t and t' be as shown in (24) and (33), respectively. Assume, without loss of generality, that for process parameters X_1, \dots, X_c , their sensitivities in t' were found to always have the same signs as their sensitivities in t , while the signs of the sensitivities of process parameters X_{c+1}, \dots, X_p were found to be possibly different in t' than in t . The largest absolute values $t_{c+1}^{\max}, \dots, t_p^{\max}$ found for the sensitivities t'_{c+1}, \dots, t'_p are now summed to find a bound B_s :

$$B_s = \sum_{k=c+1}^p t_k^{\max} \quad (34)$$

Let the pre-optimization “worst-case” corner of (23) be $\hat{X}_s^{ij} = (\hat{X}_1, \dots, \hat{X}_p)$. The process setting $\hat{X}^* = (\hat{X}_1, \dots, \hat{X}_c, 0, \dots, 0)$, and B_s are used to write the following constraint in (30):

$$(D_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) \Big|_{\hat{X}^*} + t_{setup}^j \leq T - B_s \quad (35)$$

Note that writing the constraint in (35) is more conservative than the constraint written at \hat{X}_s^{ij} and hence the latter can be removed from (30). For a hold-time constraint, a bound B_h is also computed with the difference that the bound is added to the right-hand side of the inequality and not subtracted from it to write:

$$(d_s^{ij} + r_i + \Delta r_i - r_j - \Delta r_j) \Big|_{\hat{x}^*} - t_{hold}^j \geq B_h \quad (36)$$

Table 1: Exact vs Conservative Approach

ISCAS-89 Circuit	Num Variables m	Pre-optim M_v	Post-optim M_{exact}^*	Percent Improvement	Post-optim M_{cons}^*	Percent Improvement
s400	23	9.43%	4.37%	54%	4.69%	50%
s1432	86	11.13%	5.83%	48%	7.60%	31%
s5378	327	13.31%	2.29%	83%	2.89%	78%
s9234	281	10.37%	1.39%	87%	2.58%	75%
s13207	571	8.10%	1.46%	82%	2.28%	72%
s15850	959	10.69%	2.44%	77%	3.12%	71%
s38584	2732	5.12%	-1.88%	136%	-1.00%	120%

Such constraints would ensure that all “potentially worst-case” corners are covered by using the bounds B_s and B_h to tighten the timing constraints of (30). Of course, this comes at the price of introducing some pessimism in the LP.

6. RESULTS

In order to test our approach, we have selected a set of circuits from the ISCAS-89 benchmark suite. These circuits were synthesized and mapped to a 90nm CMOS library, and then placed and routed using available commercial tools. The HSPICE netlists for the logic blocks and clock trees of these circuits were then extracted. Nominal delays and slews were characterized for all cells in the library, and a set of 10 parameters X_1, \dots, X_{10} was selected to model process variations. The ranges of those parameters were chosen such that their combined effect on the delay or slew of a cell or an interconnect resistance or capacitance value is 15%. The sensitivities of gate delays and interconnect RC -trees to these process parameters were randomly generated, in order to provide difficult test-cases. With the above variational models, the HSPICE netlists of the different test circuits were fed into our STA timing engine, which was implemented in C++ and extended to handle parameterized static timing analysis (PSTA).

Two PSTA flows were implemented, and consequently two sets of experiments were compared. The first PSTA flow, based on [9], provides an exact analysis as it propagates delays in the circuit using piecewise planar delay models, where all “potentially critical path” delays are preserved at every node. Hence, D_s^{ij} and d_s^{ij} are “lists” of hyperplanes each corresponding to a path delay. The second flow, based on [11], propagates conservative hyperplane bounds to D_s^{ij} and d_s^{ij} . A nominal STA run, without consideration to process variability, is also performed on each of the circuits and the nominal clock period found is considered to be the required clock period T_c for the test circuits.

Using the propagated delays from the exact PSTA, we start by computing the pre-optimization required timing margin, M_v , which guarantees that all constraints are met at their “worst-case” corners. M_v can be easily computed using our verification method in Section 3.2 by maximizing the left-hand side of every setup constraint written as in (10), and recording the largest value achieved over all constraints to find the smallest clock period T_v that will allow correct operation. The required clock period T_c is then subtracted from T_v to find M_v . We then ran our LP formulation of (30) to find the minimum achievable margin M^* . In our LP, the optimization variables are chosen to be variations in interconnect wire widths of the clock tree Δw_k , $1 \leq k \leq n$ where n is the number of interconnect structures in the clock tree. The bounds on variations in wire widths were set such that $0 \leq \Delta w_k \leq w_k$, where w_k is the pre-optimization wire width. In other words, wire widths are allowed, at most, to *double*, as a result of the optimization.

The minimum post-optimization timing margins, M_{exact}^* and M_{cons}^* , achieved based on exact and conservative PSTA flows, respectively, are shown in Table 1, in addition to the pre-optimization margin M_v . Given the above bounds on the allowable wire changes Δw_k , we observed improvements ranging from 50% – 136%. This shows a considerable reduction in the required margin which allows the timing constraints of the circuit to be satisfied for all process corners. Our results also show that although using piecewise planar values guarantees a better post-optimization margin, for most circuits, the difference between the improvements achieved in the two approaches is small. We also recorded and compared the runtimes of the optimization for both the exact and

Table 2: Run-time Comparison

ISCAS-89 Circuit	Exact Approach	Conservative Approach	Speed-up (\times)
s400	0.06 s	0.05 s	1.16
s1432	2.45 s	0.98 s	2.5
s5378	0.93 s	0.72 s	1.29
s9234	1.31 s	1.25 s	1.05
s13207	3.56 s	2.3 s	1.55
s15850	13.27 s	9.83 s	1.35
s38584	20.78 s	13.71 s	1.52

conservative delay approaches. The results in Table 2 show that although the hyperplane approach is more conservative in terms of post-optimization margin, it typically produces a speed-up between $1.16\times$ and $2.5\times$. Thus, we see that one is faced with a runtime-accuracy tradeoff, where a good speed-up is achieved at the expense of some pessimism that is introduced to the analysis.

7. CONCLUSION

In this work, we presented a technique that modifies clock network wire widths and/or lengths so that a circuit meets its timing constraints at all process corners. Our method considers delay variability in both the clock network and the functional blocks, and is applicable near the end of the design flow. We showed that the problem of finding the required changes in line widths and/or lengths can be formulated as a Linear Program, which can be solved efficiently. Using our clock skew tuning approach, designers can considerably reduce the effect of process variations on circuit performance, as shown in our results.

8. REFERENCES

- [1] R. S. Tsay. Exact zero skew. In *Int. Conf. on Computer Aided Design (ICCAD)*, pages 336–339, 1991.
- [2] T. H. Chao, Y. C. Hsu, and J. M. Ho. Zero skew clock net routing. In *DAC*, pages 518–523, June 1992.
- [3] J. Fishburn. Clock skew optimization. *IEEE Trans. on Computer-Aided Design*, 39(7):945–951, July 1990.
- [4] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun. Analysis and design of latch-controlled synchronous digital circuits. *IEEE TCAD*, 11(3):322–333, March 1992.
- [5] J. L. Neves and E. G. Friedman. Buffered clock tree synthesis with non-zero clock skew scheduling for increased tolerance to process parameter variations. *Journal of VLSI Signal Processing*, 16:149–161, 1996.
- [6] I.M. Liu, T.L. Chou, A. Aziz, and DF Wong. Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion. In *Proceedings of the 2000 international symposium on Physical design*, pages 33–38. ACM New York, NY, USA, 2000.
- [7] S. Pulella, N. Menezes, and L. T. Pillage. Reliable non-zero skew clock trees using wire width optimization. In *Design Automation Conference*, pages 165–170, June 1993.
- [8] S. Pulella, N. Menezes, and L. T. Pileggi. Post-processing of clock trees via wiresizing and buffering for robust design. *IEEE TCAD*, 15(6):691–701, 1996.
- [9] K. R. Heloue, S. Onaissi, and F. N. Najm. Efficient block-based parameterized timing analysis covering all potentially critical paths. In *ICCAD*, pages 173–180, November 2008.
- [10] S. V. Kumar, C. V. Kashyap, and S. S. Sapatnekar. A framework for block-based timing sensitivity analysis. In *Design Automation Conference*, pages 688–693, 2008.
- [11] S. Onaissi and F.N. Najm. A linear-time approach for static timing analysis covering all process corners. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1291–1304, 2008.