Process Variation Robust Clock Tree Routing [*]

Wai-Ching Douglas Lam, Cheng-Kok Koh
ECE, Purdue University
W. Lafayette, IN 47907-1285
{douglam, chengkok}@ecn.purdue.edu

Abstract

As the minimum feature sizes of VLSI circuits get smaller while the clock frequency increases, the effects of process variations become significant. We propose a UST/DME based approach to perform simultaneous non-zero clock skew scheduling and clock tree routing, taking into consideration the effects of process variations on clock skews. Our approach ensures that the generated clock tree has a high tolerance to process variations while minimizing the total capacitance of the clock tree, which is proportional to the total wirelength and the total number of buffers. Monte Carlo simulations show that our approach generates clock trees that are highly tolerant to process variations.

1 Introduction

As the minimum feature sizes of VLSI circuits get smaller while the clock frequency increases, the effects of process variations become significant. These variations arise during the manufacturing process from the tolerances of the equipment used [14].

Process variation on clock trees may result in violations in clock skew constraints. In [12], it is shown that interconnect variations in clock trees can cause as much as 25% variation in clock skew. In buffered clock trees, more sources of variation are introduced as buffers are affected by process variations as well.

One approach to counter such variations would be to use Monte Carlo simulations to determine the worst case timing performance before constructing the clock tree. However, that requires a high computational cost. Although statistical timing analysis may reduce the run-time complexity [1, 3], we use a preventive approach towards synthesizing clock trees that are robust to process variations by considering the “worst-case” delays along all paths in a clock tree. This greatly simplifies the layout synthesis process, and experimental results are promising.

While non-tree designs [15] can benefit from a higher tolerance to skew variation, the increase in wire lengths present a larger load to the clock driver and leads to an increase in clock power consumption.

Increasing the common paths between sinks that are more sensitive to process variation can increase the tolerance to process variation [17]. However, in contrast to many existing clock tree construction algorithms [2, 11, 16], their approach does not take into consideration the locality of clock pins and no emphasis is placed on wirelength reduction.

Zero Skew Tree (ZST) construction algorithms [2, 4] inherently cannot handle process variations; zero skew constraints can never be satisfied in the presence of process variations. However, generalizations of zero skew tree construction to Bounded Skew Tree (BST) [11] construction or Useful Skew Tree (UST) [16] construction are naturally more robust to process variations. A straight-forward approach to handling process variations in these construction algorithms is to reduce the skew ranges by a safety margin (usually a predetermined constant), thereby preventing the algorithm from selecting skew values that are at the very extremes of the skew constraints. This translates to an increase in tolerance to process variations. However, since the sensitivity to process variation is dependent on the topology, the value of the safety margin may be inadequate or over-pessimistic. In particular, different parts of a clock tree should have different safety margins as they have different sensitivities to process variations.

In [13], the authors extend the Bounded Skew Tree/Deferred-Merge Embedding (BST/DME) approach to construct a process variation aware clock tree. During the construction of the clock tree, it considers the worst case variation limit due to process variations. With any BST/DME approaches, there is a limitation that the solution space available for clock skew scheduling may be very limited. To be exact, the skew bound is the intersection of all skew constraints between every pair of sinks, which may be an empty set.

To overcome the limitation of restricted skew bounds, in this work, we extend the Useful Skew Tree/Deferred-Merge Embedding (UST/DME) approach to improve on achieving a process variation robust (PVR) useful skew tree while minimizing wirelength. We refer to our algorithm as the PVR-UST/DME approach. During the intermediate steps of the bottom-up construction of the UST, we analyze and consider the worse case bound on wire delays and clock skews such that the resulting clock tree is robust to process variations. The primary objective is to minimize the total capacitance of the clock tree, which is proportional to the total wirelength.

In this work, we also consider buffer insertion under process variation. There are two issues related to buffer insertion. First, process variation affects logic devices (buffers in this work) as well. As a result, more variation is introduced to the problem. Second, the buffers inserted may have a great impact on topology construction, since we do not enforce that the number of buffers from the root to all sinks to be equal. This may result in an unbalanced tree which reduces the tolerance to process variations. We insert the minimal number of buffers to reduce the total capacitance of the clock tree.

The paper is organized as follows: In Section 2, we present some preliminary definitions and background information. The UST/DME algorithm is briefly reviewed in Section 3. We present the PVR-UST/DME algorithm that performs scheduling and construction of a useful skew clock tree without a prescribed topology in Section 4. We provide some experimental results in Section 6 and conclude the paper.

2 Effects of Process variation on Delay and Skew

In this paper, the Elmore delay model is used to compute the delays and skews. Each wire with length l and width w is modeled by a RC π-type circuit with parasitics $R = r_0 \times l/w$, $C = c_a \times l \times w/2 + c_f \times l/2$, where r_0 is the wire resistance per unit

[*]This work was supported in part by NSF under contract number CCR-9984553, CCR-0203362, and a DAC scholarship

square, c_a is the area capacitance per unit length and c_f is the fringing capacitance per unit length.

Without process variations, w can be treated as a constant and has a nominal width (specified by the designer), denoted by w_{nom}. In this paper, subscript "nom" represents the condition when process variations are absent. With the introduction of variation on w, the actual width w_e of the wire segment e in the manufactured circuit may vary between w_{max} (maximum width) and w_{min} (minimum width). In this paper, we set w_{max} to be $w_{nom}+3\sigma$ and w_{min} to be $w_{nom}-3\sigma$, where σ is the standard deviation of a given process corner. The downstream capacitance C_{T_e} seen from segment e may also vary. The above variations will cause the delay to deviate. Although the variation on wire height can be factored into this work as another cause of delay variation, we focus only on wire width variations in this paper.

In this work, the buffer model that we use is a switch-level RC model that has three parameters: input capacitance ($cbin$), output resistance ($rbuf$) and intrinsic delay (d_b). Process variation will also affect these buffer parameters. Therefore, for input capacitance, we have $cbin_{min}$, $cbin_{max}$ and $cbin_{nom}$. For output resistance, we have $rbuf_{min}$, $rbuf_{max}$ and $rbuf_{nom}$. For intrinsic delay, we have $d_{b_{min}}$, $d_{b_{max}}$ and $d_{b_{nom}}$.

The variations in the above mentioned parameters are all taken into consideration in our algorithm. We shall show how these variations affect the clock tree construction. For now, we simply assume that the combination of these variations leads to a maximum delay (t_{max}) and minimum delay (t_{min}) of any wire.

For a clock tree, we use $S=\{s_1,s_2,...,s_n\}$ to denote the set of clock sinks (clock pins of sequential elements) and the clock source as s_0. Consider a synchronous circuit using edge triggered flip-flops (FFs) under a single-phase clocking scheme and let s_j be the clock pin of FF_j. For two clock sinks s_j and s_k, if there are no process variations, the skew between these sinks is defined as $skew_{j,k_{nom}}=t_{j_{nom}}-t_{k_{nom}}$, where $t_{j_{nom}}$ and $t_{k_{nom}}$ are the nominal delays (clock arrival time) from s_0 to s_j and s_0 to s_k respectively.

A pair of FFs are *sequentially adjacent* when only some purely combinational logic exists between the two flip-flops. Let FF_j and FF_k are two sequentially adjacent flip-flops and let the nominal clock arrival times to FF_j and FF_k be $t_{j_{nom}}$ and $t_{k_{nom}}$ respectively. In order to prevent hold time violation:

$$t_{j_{nom}}-t_{k_{nom}} \;\geq\; t_{hold,\max}-t_{pFF,\min}-t_{logic,\min}+\delta_l\,. \tag{1}$$

Similarly, to prevent setup time violation:

$$t_{j_{nom}}-t_{k_{nom}} \;\leq\; T_{clk}-t_{pFF,\max}-t_{logic,\max}-t_{setup,\max}-\delta_u\,. \tag{2}$$

These two inequalities pose upper and lower bound constraints on $skew_{j,k_{nom}}$. $t_{logic,\max}$ and $t_{logic,\min}$ are the maximum and minimum propagation delays through the combinational logic block; $t_{pFF,\max}$ and $t_{pFF,\min}$ are the maximum and minimum propagation delays through the flip-flop; and T_{clk} is the clock period. t_{setup} and t_{hold} are the setup and hold times of the flip-flop. In [16], safety margins δ_l and δ_u, are both included in the inequalities to enhance the robustness of the circuits. For simplicity, we use $l_{j,k}\leq skew_{i_{nom},j_{nom}}=t_{j_{nom}}-t_{k_{nom}}\leq u_{j,k}$ to represent lower- and upper-bound skew constraints between FF_i and FF_j. We also denote an inequality $a\leq x\leq b$ as $x\in[a,b]$. Therefore, we can express Eqns. (1) and (2) as:

$$skew_{j_{nom},k_{nom}}\in[l_{j,k},u_{j,k}]\,. \tag{3}$$

With the presence of process variations, we have:

$$t_{j_{min}}-t_{k_{max}}\;\leq t_{j_{nom}}-t_{k_{nom}}\;\leq t_{j_{max}}-t_{k_{min}}\,. \tag{4}$$

Therefore, to ensure correct operation, we have:

$$t_{j_{min}}-t_{k_{max}} \;\geq\; t_{hold,\max}-t_{pFF,\min}-t_{logic,\min}, \tag{5}$$

and

$$t_{j_{max}}-t_{k_{min}} \;\leq\; T_{clk}-t_{pFF,\max}-t_{logic,\max}-t_{setup,\max}. \tag{6}$$

In this paper, we assume that t_{hold}, t_{setup}, t_{logic}, t_{pFF} takes on the maximum or minimum values under process variations. Moreover, we explicitly include clock skew variations in our formulation. Therefore, safety margins are not utilized in Eqns. (5) and (6).

In this work, the problem that we would like to solve can be stated as follows: *Given the skew constraints between the clock sinks, and the clock pin locations, construct a useful skew clock tree such that it is tolerant to variations in wire width (limited between w_{max} and w_{min}) and in device parameters ($cbin$, $rbuf$ and d_b). The primary objective is to minimize the total capacitance of the clock tree, which is proportional to the total wirelength and the total number of buffers.*

3 Review of the UST/DME Algorithm

In this work, we make use of the UST/DME approach to construct the clock tree. The process of tree construction follows the DME-based paradigm:

1. A bottom-up phase to construct a binary tree of merging regions or segments, and
2. A top-down phase to determine the exact locations of the internal nodes.

Merging segments are Manhattan arcs constructed temporarily during the bottom-up phase that represent the loci of possible embedding points of internal nodes of the final clock tree. Since a particular skew value is always constant on a merging segment under the Elmore delay model, the locations of merging segments determine the clock skew assignment of the final clock tree. When the skew can vary within a range bounded by skew constraints, the concept of merging segments can be generalized to that of merging regions.

As the top-down phase is identical to those presented in [4, 6], emphasis in this paper is placed on the bottom-up phase, which proceeds as follows: At the beginning of the algorithm, we have a forest F of singleton subtrees T_{s_i}'s, each containing a clock sink in S. While F has more than one subtree, two subtrees are selected , say T_u and T_w, for merging to form the new parent subtree T_v. This process continues until one subtree remains in F , which is the final clock tree.

In this work, we assume that the topology of the clock tree is not given. Hence, the order of subtree merging must be determined. This is the most crucial aspect in achieving minimal wirelength because it controls the topology of the constructed clock tree. The order is determined from the nearest neighbor graph (NNG) [10], which stores the promising merging pairs of subtrees based on the estimated merging cost (a measure of the wire length required for the merging operation) of all subtrees in F . In one iteration, $|F|/k$ *independent* nearest-neighbor pairs from the NNG are selected in a non-decreasing order of merging cost for merging, where k is a constant ranging from 2 to 4 [10]. The NNG is updated after each iteration.

3.1 Incremental Skew Scheduling in UST/DME

In the UST/DME algorithm, when two subtrees are merged, a new merging region that represents the skew range bounded by the skew constraints of all the sinks that are children to the two subtrees is created. While it means that commitment to a particular skew value within this skew range is allowed, the validity of this range must first be ensured (feasibility). Furthermore, committing to a particular skew causes a change in other skew ranges. Therefore, before we can further proceed with the subtree merging, we must ensure that the change does not cause infeasibility to occur. The process of committing to a skew and ensuring the validity of the skew range is known as skew scheduling.

Due to space constraints, the fundamental aspects of ***Incremental Skew Scheduling*** is cited here without proof. In incremental skew scheduling, a ***constraint graph*** $G_c(V,E)$ is used to capture all the skew constraints between every single pair of clock sinks/FFs that are adjacent to each other [7, 8]. G_c is a directed graph with vertices $v_j \in V$ and directed edges $e_{j,k} \in E$. v_j represents the sink nodes while the edge weight of edges in E records the skew constraints (Eqns. (1) and (2)) between every pair of adjacent clock sinks.

There are two important aspects in Incremental Skew Scheduling. One is the creation of the ***feasible skew range (FSR)*** between any pair of sinks, which is stored in a FSR matrix [16]. A key property of the FSR is that when a skew commitment is made within a FSR, it is guaranteed that a feasible clock skew schedule exists. The second aspect is that after we commit the skew between two sinks to a particular value in the corresponding FSR, an update in the FSR matrix is done to maintain the feasibility of the FSRs.

In the UST/DME algorithm, skew ranges are computed and merging regions are constructed without considering process variations. With the presence of process variations, we must ensure that the merging regions has skew ranges that satisfy Eqns. (5) and (6), which would ensure that they also satisfy Eqns. (1) and (2). We shall present our PVR-UST/DME approach and show how our approach handles process variations in the next section.

4 PVR-UST/DME Approach

4.1 PVR Merging Region Construction

Suppose we have a Manhattan merging segment L such that all points on this segment is equidistant (rectilinear) to a clock sink, say s_1. Under the Elmore delay model (ignoring process variations), the delay from any point on L to s_1 is a constant. We denote the delay to be $t(L \rightsquigarrow s_1)$. Now suppose that all points on L are also equidistant to another sink, s_2, and the delay from any point on L is $t(L \rightsquigarrow s_2)$. Therefore the skew that L represents is $t(L \rightsquigarrow s_1) - t(L \rightsquigarrow s_2)$, is also a constant. With such a property, we can translate the FSR between two clock sinks (obtained from the FSR matrix) to a region bounded by two merging segments. One merging segment corresponds to the skew value at the lower bound of the FSR while the other at the upper bound of the FSR. We call this region a merging region $mv(i)$.

Now suppose we are trying to merge two merging segments $L_u \subseteq mr(u)$ and $L_w \subseteq mr(w)$ to form a new parent merging region $mr(v)$. $mr(u)$ is a parent to clock sinks $S_u = \{s_{u_1}, s_{u_2}, ..., s_{u_p}\}$ and $mr(w)$ is a parent to clock sinks $S_w = \{s_{w_1}, s_{w_2}, ..., s_{w_q}\}$. Suppose that for a particular pair of sinks (s_{u_1}, s_{w_1}), $FSR_{s_{u_1}, s_{w_1}} = [-l_{u_1,w_1}, u_{u_1,w_1}]$. To represent this FSR in $mr(v)$, we can construct two merging segments in $mr(v)$. One single merging segment L_{v_1} representing the lower bound would be constructed such that all points on L_{v_1} satisfy:

$$t(L_{v_1} \rightsquigarrow s_{u_1}) - t(L_{v_1} \rightsquigarrow s_{w_1}) = -l_{u_1,w_1}\,, \tag{7}$$

where $t(L_v \rightsquigarrow u_1)$ denotes the delay from L_v to s_{u_1} and $t(L_v \rightsquigarrow w_1)$ denotes the delay from L_v to s_{w_1}. The other single merging segment L_{v_2} representing the upper bound would be constructed

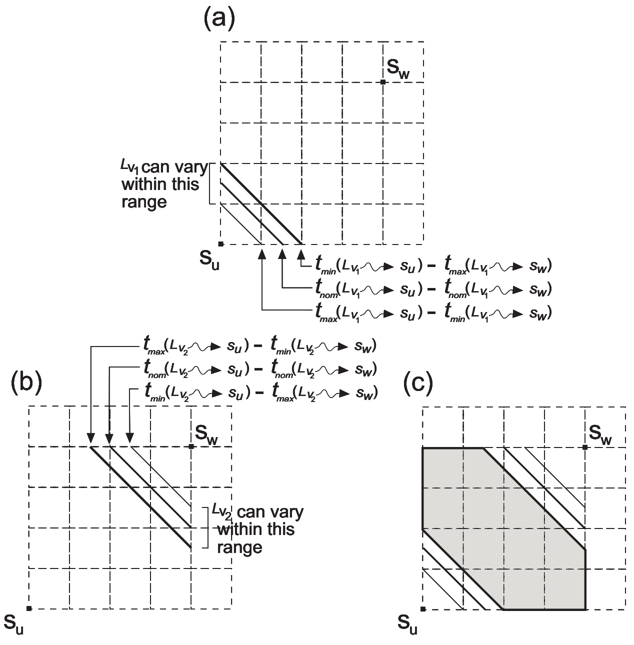
(a)
s_w
L_{v1} can vary within this range
s_u
$t_{min}(L_{v_1} \rightsquigarrow s_u) - t_{max}(L_{v_1} \rightsquigarrow s_w)$
$t_{nom}(L_{v_1} \rightsquigarrow s_u) - t_{nom}(L_{v_1} \rightsquigarrow s_w)$
$t_{max}(L_{v_1} \rightsquigarrow s_u) - t_{min}(L_{v_1} \rightsquigarrow s_w)$
$t_{max}(L_{v_2} \rightsquigarrow s_u) - t_{min}(L_{v_2} \rightsquigarrow s_w)$
$t_{nom}(L_{v_2} \rightsquigarrow s_u) - t_{nom}(L_{v_2} \rightsquigarrow s_w)$
$t_{min}(L_{v_2} \rightsquigarrow s_u) - t_{max}(L_{v_2} \rightsquigarrow s_w)$
(b)
s_w
L_{v2} can vary within this range
s_u
(c)
s_w
s_u

Figure 1: Construction of narrowed merging region for s_u and s_w in the presence of process variation.

such that all points on L_{v_2} satisfy:

$$t(L_{v_2} \rightsquigarrow s_{u_1}) - t(L_{v_2} \rightsquigarrow s_{w_1}) = u_{u_1,w_1}\,. \tag{8}$$

The feasible skew region in $mr(v)$ is the region bounded by L_{v_1} and L_{v_2}. Note that this is the merging region that satisfies only the FSR for s_{u_1} and s_{w_1}. Therefore, we have to repeat this process with every pair of sinks, one sink from each subtree. However, it turns out that if process variation is not present, the merging segments are identical for all pairs of sinks, therefore, only the FSR of one of these pairs of sinks is required to construct the merging region [16].

Consider the same example in the presence of process variation. Since the delays from the new L_v to sinks S_u and the delays from the new L_v to S_w can vary within a range between a maximum and a minimum value, we have to go through all pairs of sinks (one sink from each subtree) to construct the merging region.

Here, we take a closer look at what happens to the merging segments for the same pair of sinks (s_{u_1}, s_{w_1}) under process variations. To represent this FSR in $mr(v)$, we can construct two merging segments in $mr(v)$, L_{v_1} and L_{v_2}. But since,

$$\begin{aligned} t_{min}(L_v \rightsquigarrow s_{u_1}) - t_{max}(L_v \rightsquigarrow s_{w_1}) \le\ & t_{nom}(L_v \rightsquigarrow s_{u_1}) - t_{nom}(L_v \rightsquigarrow s_{w_1}) \\ \le\ & t_{max}(L_v \rightsquigarrow s_{u_1}) - t_{min}(L_v \rightsquigarrow s_{w_1})\,. \end{aligned} \tag{9}$$

Therefore, the location of L_{v_1} can vary between:

$$t_{min}(L_{v_1} \rightsquigarrow s_{u_1}) - t_{max}(L_{v_1} \rightsquigarrow s_{w_1}) = -l_{u_1,w_1}\,, \tag{10}$$

and

$$t_{max}(L_{v_1} \rightsquigarrow s_{u_1}) - t_{min}(L_{v_1} \rightsquigarrow s_{w_1}) = -l_{u_1,w_1}\,. \tag{11}$$

Likewise, the location of L_{v_2} can vary between:

$$t_{min}(L_{v_2} \rightsquigarrow s_{u_1}) - t_{max}(L_{v_2} \rightsquigarrow s_{w_1}) = u_{u_1,w_1}\,, \tag{12}$$

and

$$t_{max}(L_{v_2} \rightsquigarrow s_{u_1}) - t_{min}(L_{v_2} \rightsquigarrow s_{w_1}) = u_{u_1,w_1}\,. \tag{13}$$

Since by definition, the merging region is the region where the merging segments can satisfy the $FSR_{s_{u_1},s_{w_1}}$ (for this sink pair (s_{u_1}, s_{w_1})), we take the most narrow region bounded by L_{v_1} and L_{v_2} to bound the merging region of $mr(v)$. This guarantees that any segment selected within this narrowed merging region $nmr_{s_{u_1},s_{w_1}}$ still satisfies $FSR_{s_{u_1},s_{w_1}}$ even in the presence of process variation. In our algorithm, we only compute the most narrow L_{v_1} and L_{v_2}; the six merging segments in Fig. 1 is drawn to illustrate the effects of process variations.

Note that this $nmr(v)_{s_{u_1},s_{w_1}}$ satisfies only $FSR_{s_{u_1},s_{w_1}}$. Of course, the final $mr(v)$ must also have a feasible region that satisfies the remaining pairs of sinks. We can form these narrowed merging regions $nmr(v)_{s_{u_i},s_{w_j}}$ where $i = 1,...,p$ and $j = 1,...,q$.

Once the individual narrowed merging regions of every pair of sinks have been formed, we take the intersection of all these narrowed merging regions and form the final $mr(v)$:

$$mr(v) = \bigcap_{s_{u_i}\in S_u, s_{w_j}\in S_w} nmr(v)_{s_{u_i},s_{w_j}} \qquad (14)$$

The outline of the PVR-UST/DME algorithm is given in Figure 2. To construct the merging region $mr(v)$, we sample the merging region $mr(u)$ with h number of merging segments $L_u \subseteq mr(u)$. It is important to realize that each of these merging segments actually represents a particular skew value within the FSR. Therefore, when we consider a merging segment L_u from the merging region $mr(u)$ for merging (like in step 6), we are essentially committing to a skew value within the FSR. Therefore, an update of the FSR matrix is required. This means that the merging region for the other subtree $mr(w)$ has to be reconstructed with the new updated FSR. Once this merging region is reconstructed, we sample it with h merging segments $L_u \in mr(u)$. We construct a merging region for L_u and each L_w. We iterate this process for all merging segments from $mr(u)$, constructing h^2 merging regions in total. Among these merging regions, we select the least cost merging region as the new parent merging region $mr(v)$.

Although the clock trees constructed with our PVR-UST/DME algorithm have high tolerance to process variation, we cannot guarantee all trees are entirely tolerant. This is because as we construct the trees bottom up, we repeatedly commit to skews that leads to the tightening of the remaining feasible skew ranges. As we get higher up, the wires (from root to sink) become longer and therefore have greater variations in delays. It is certainly possible that the delay variations between two sinks may become larger than the corresponding feasible skew range, which means that the narrowed merging region for that particular pair of sinks would not have a proper bounded region. In such a case, we use the nominal merging regions (without considering process variations) for that pair of sinks and continue to construct the narrowed merging regions for the remaining pairs of sinks. This ensures that the FSR will still be valid and that the total wirelength will be minimized at a small sacrifice of process tolerance as shown in the experimental results. The use of wiresizing to expand the merging region or the use of topology perturbation [18] may be useful in overcoming the above limitation; we will consider such methods in a future extension of this work.

4.2 Delay Bounds and Skew

For the PVR merging region construction, it requires the values of t_{max} and t_{min}. Here, we describe how these values are obtained.

In a clock tree, the path from root to a sink may contain off-branch subtrees (subtrees that do not contain any segments along the path). At the nodes along the path, a downstream capacitance of these off-branch subtrees can be seen. Let the off-branch subtree rooted at node i be T_{off_i} and its nominal capacitance be $C_{T_{off_{i_{nom}}}}$. With the presence of process variations, each of these capacitances may vary between $C_{T_{off_{i_{max}}}}$ (all wires in T_{off_i} having the maximum wire width) and $C_{T_{off_{i_{min}}}}$ (all wires in T_{off_i} having the minimum wire width). In our bottom-up clock tree construction, we keep track of the minimum, maximum and nominal downstream capacitance of these subtrees.

Input: Clock pins S; local skew constraints C;
Output: A clock tree routing T satisfying constraints in C or no solution

```
1.  Construct constraint graph $G_C = (V,E)$ from $C$
2.  If $G_C$ has negative cycles, return no solution
3.  Build the $FSR$ matrix from $G_C$
4.  Determine the order of subtree merging from nearest
      neighbor graph (NNG) [9] based on merging cost
5.  for each merging of subtrees $T_u$ and $T_w$ to form $T_v$
      based on (NNG)
6.      Select $L_u \in mr(u)$ (skew commitment)
7.      Update $FSR$ matrix for skew commitment made in 6
8.      Reconstruct $mr(w)$ if $FSR$ matrix is updated
9.      Select $L_w \in mr(w)$ (skew commitment)
10.     Update $FSR$ matrix for skew commitment made in 9
11.     for all pairs of sinks between $s_i \in T_u$ and $s_j \in T_w$
12.       Get $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ for $s_i \in T_u$, $s_j \in T_w$
13.       Construct narrowed merging region $nmr(v)$
            with feasible skew $skew_{i,j} \in FSR_{i,j}$
14.     Form final $mr(v)$ from the intersection of all $nmr(v)$s
15. Reconstruct $mr$s caused by changes in $FSR$ matrix
16. Perform DME top-down Embedding (Embed each node
      $v$ on $L_v \in mr(v)$ in top-down order)
```

Figure 2: The PVR-UST/DME algorithm.

To find t_{min} for a path where *all the edge lengths are known*, we transform this into a minimum delay wire sizing problem [5]. The minimum delay wire sizing problem is to assign monotonously decreasing wire widths (within w_{max} and w_{min}) from root to sink to optimize t_{min}. For the off-branch subtrees along the path, we set their capacitance values to be $C_{T_{off_{i_{min}}}}$. We then utilize a quick iterative process from [5], that computes the corresponding wire widths in about $O(d)$ time, where d is the number of wire segments. Similarly, for t_{max}, the maximum delay can be found by assigning monotonously increasing wire widths from root to sink (within w_{max} and w_{min}). In this case, we set the capacitance of the off-branch subtrees along the path to be $C_{T_{off_{i_{max}}}}$ and perform the iterative wire width assignment process to determine the maximum delay achievable.

This provides a reasonable bound on the minimum (maximum) delay of any wire path due to varying wire widths, instead of assigning maximum (minimum) widths to all wire branches.

However, in order to construct the merging regions as shown in Section 4.1, we must compute the exact location of the boundary merging segments based on the minimum (maximum) delays and skews. In other words, we use Elmore delay to compute the wire lengths required from the child merging regions in order to satisfy the skew value required on the merging segment (i.e. not all edge lengths are known *a priori*). Using the wire sizing approach, this become slightly complicated because the computed length changes the upstream resistance and capacitance seen from the sinks, which in turn changes the maximum and minimum delay and skews.

Therefore, we start with a initial guess of the lengths (we assume nominal delays for this guess). We calculate the minimum and maximum delays from the merging segment to the sinks. The skew is then computed based on these delays. After that, we compute the difference between this computed skew and the desired skew and make changes to the lengths. This process is repeated until the length converges to the desired skew value. Convergence can be guaranteed because Elmore delay is proportional with lengths. Increasing the length on one side increases the delay with respect to the same sink.

4.3 Buffer Insertion

In this work, we have also included buffer insertion in our PVR-UST/DME algorithm to maintain an acceptable clock signal integrity with steep rising and falling edges.

When we build the trees of merging regions bottom-up, we keep track of the maximum delay from the roots of the trees to either sinks or buffers in a DC-connected component. If we detect that the 10% to 90% rise time (0.9RC for distributed interconnects) in any root of a subtree in F is comparable to a small fraction of the clock period (in this paper we set it to be 5%), we insert a buffer to that particular root.

To compute the maximum delay with the presence of buffers, we assign $cbin_{max}$, $rbuf_{max}$ and $d_{b_{max}}$ for the buffers that lie along the path and perform the iterative wire width assignment process. Likewise, we use $cbin_{min}$, $rbuf_{min}$ and $d_{b_{min}}$ when we compute the minimum delay. These delay bounds affect the merging region construction as mentioned in Section 4.1.

5 Complexity Analyses

Suppose we start with n sinks. In iteration i, $|F|/k$ subtrees are selected for merging, based on the NNG. Therefore n/k^i subtrees are present in iteration i. For simplicity, we assume that all subtrees are balanced binary trees.

Consider the merging of a pair of subtrees rooted at nodes u and w. The two subtrees have p sinks and q sinks, respectively. Therefore the maximum number of wire segments from root u and w to its respective sinks are $log\, p$ and $log\, q$, respectively. To update the minimum and maximum delays from root u to each of its sinks, $O(p\, log\, p)$ operations are required. Likewise, it requires $O(q\, log\, q)$ operations to update the minimum and maximum delays from root w to each of its sinks. Suppose we have h samples in merging region $mr(u)$. For one sample L_u in merging region $mr(u)$ (essentially a skew commitment), we require $O(n^2)$ operations to first perform a FSR computation [16]. To reconstruct $mr(w)$ (due to the skew commitment in $mr(u)$), it requires $O(q^2\, logq)$ operations. For this particular sample L_u from $mr(u)$, h narrow merging regions for the parent node are constructed. Since it takes $O(pq\, log\,(p+q))$ number of operations to construct a single narrow merging region, it requires a total of $O(n^2 + q^2\, log\, q + hpq\, log\,(p+q))$ operations to construct the narrow merging regions for the particular sample L_u from $mr(u)$. This process is repeated for h samples in $mr(u)$, which would require $O(h(n^2 + q^2\, log\, q + hpq\, log\,(p+q)))$ operations.

The highest cost is to update the NNG where we compute the merging cost between all neighboring merging regions. Here, we are essentially performing normal merging operations except that we are obtaining only the cost of merging but not actually merging the regions together (a test merge). In the worst case, we have to compare $(n/k^i)^2$ merging regions. Therefore, it requires $O((n/k^i)^2(p^2\, log\, p + h(n^2 + q^2\, log\, q + hpq\, log\,(p+q))))$ operations, where $p^2\, log\, p$ is the number of operations to reconstruct $mr(u)$ because the FSR may have changed by the previous merging steps in the same iteration.

Without loss of generality, we let $k = 2$. Therefore, after $log\, n$ iterations, the clock tree construction is complete. Suppose that in iteration i, both p and q are approximated by 2^i in the worst case and h is a small constant. Therefore, the overall complexity is:

$$\sum_{i=0}^{log\, n} n^2\left(\frac{n^2}{2^{2i}} + (3i+1)\log 2\right) \approx O(n^4) \tag{15}$$

Table 2: **Results with Buffer Insertion.**

Circuit	PVR-UST/DME Wirelength μm	Buffers inserted	% yield
s1423	95842	10	100.00
s5378	154334	21	93.18
s15850	402123	46	89.56

6 Experiments and results

The proposed clock tree routing algorithm has been implemented in C++ and tested on three ISCAS89 benchmark circuits on a Sun UltraSparc-II. The wires are assumed to have a resistance of 0.001Ω per unit square, an area capacitance of $1.2fF/\mu m$ and a fringing capacitance of $1.5fF/\mu m$. These numbers are typical for a $0.25\mu m$ process technology. The buffers used have a nominal input capacitance of $25fF$, a nominal output resistance of 75Ω and a nominal intrinsic delay of ps. The targeted clock speed is 1Ghz. The amount of variation of all the parameters that are subjected to process variation has been set to an arbitrary value of $\pm 20\%$. In other words, the minimum is set at 80% of the nominal value and the maximum is set at 120% of the nominal value for any interconnect or device parameter.

The results are tabulated in Tables (1-2). In Table 1, we see that the PVR-UST/DME has a better performance in terms of wirelength than the Zero Skew Tree routing (ZST) approach. When compared to the UST/DME algorithm, the wirelengths attained from PVR-UST/DME are longer for all three benchmarks. It is easy to understand the cause of the increase. The results obtained from the UST/DME algorithm [16] are based on the fact that process variation is not considered; therefore the safety margins δ_l and δ_u are set to zero. This results in larger merging regions and they become closer to each other. Therefore a shorter wirelength can be obtained. In Table (2), results with buffer insertion is presented.

6.1 Monte Carlo Simulations

To truly evaluate the quality of our solution and to justify the necessity to consider process variations, we performed Monte Carlo HSPICE simulations on the constructed clock trees. In these experiments, the constructed tree (maintaing the tree topology) is fed into the simulator and wire widths are assigned to all the tree branches. The assignment of wire widths follows a normal distribution that has a mean of w_{nom} and a standard deviation that corresponds to w_{max} and w_{min} at the $\pm 3\sigma$ point. The skews between all pairs of sinks are evaluated based on this width assignment using HSPICE and are compared with the given skew constraints. If the skew between any pair of sinks skew violates its corresponding skew constraint, the tree is considered to be faulty. For each clock tree, 10000 simulations are performed with a new set of width assigment in each run to determine the yield. The yield is defined as the percentage of clock trees tested in the simulations that meet the skew constraints under the presence of process variations.

To compare with the UST/DME approach, three clock trees are constructed with safety margins (Eqns. (1) and (2) of $0ps$, $150ps$ and $300ps$. We assume an equal safety margin for δ_l and δ_u. Any value higher than $300ps$ would result in infeasible skew

Table 1: **Comparison of Wirelengths.**

Circuit	Number of clock pins	ZST Wire-length μm	*UST/DME Wire-length μm	PVR-UST/DME Wire-length μm	% Reduction over ZST	% Increase over UST/DME	CPU Time (min:sec)
`s1423`	74	107277	89189	95844	10.6	7.5	0:15
`s5378`	179	176517	149391	156338	12.5	4.7	1:44
`s15850`	597	448599	355561	402153	10.4	13.1	15:17

*Please refer to Section 6 for more details

Table 3: **Results of Monte Carlo Simulations.**

Circuit	% Yield using UST/DME $\delta_u = \delta_l = 0ps$	% Yield using UST/DME $\delta_u = \delta_l = 150ps$	% Yield using UST/DME $\delta_u = \delta_l = 300ps$	% Yield using PVR-UST /DME
`s1423`	100.00	100.00	100.00	100.00
`s5378`	8.90	19.25	38.51	97.48
`s15850`	0.00	0.00	0.00	94.32

schedules i.e., G_c with negative cycles. The results are tabulated in Table 3.

From Table 3, we see that process variations can have a great impact on yield. While smaller sized clock trees are more tolerant to process variations, process variations can seriously affect large sized clock trees. In the UST/DME approach, if safety margins are not set (set to $0ps$), the algorithm takes full advantage of the entire FSR of every merging region and tends to select the extreme limits of skew bounds to achieve shorter wirelengths. Therefore, it may be very sensitive to process variations. While increasing the safety margins seems to increase the tolerance to process variations, if it is improperly specified, it may be inadequate in reducing the effects of process variations. This is because different tree topologies requires different safety margins. In cases where the topology is not given, it is virtually impossible to determine an appropriate safety margin. With buffers inserted, the yield decreases due to the addition of device variation (Table 2). Nevertheless, the yield of the proposed PVR-UST/DME approach is still significantly higher than the UST/DME approach.

References

[1] A. Agarwal, Blaauw. D., V. Zolotov, and S. Vrudhula. Computation and refinement of statistical bounds on circuit delay. In *Proc. Design Automation Conf*, pages 348–353, 2003.

[2] K. D. Boese and A. B. Kahng. Zero-skew clock routing trees with minimum wirelength. In *Proc. IEEE Int. ASIC Conf.*, pages 1.1.1–1.1.5, September 1992.

[3] H. Chang and S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single pert-like traversal. In *Proc. Design Automation Conf*, 2003.

[4] T.-H. Chao, Y.-C. Hsu Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng. Zero skew clock routing with minimum wirelength. *IEEE Trans. on Circuits and Systems*, 39(11):799–814, November 1992.

[5] Chung-Ping Chen and D. F. Wong. Optimal wire-sizing function with fringing capacitance consideration. In *Proc. Design Automation Conf*, pages 604–407, 1997.

[6] 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 Electronics Systems*, 3(3):341–388, 1998.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter 25.5, pages 539–543. The MIT Press, 1990.

[8] R. B. Deokar and S. S. Sapatnekar. A graph-theoretic approach to clock skew optimization. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 407–410, 1994.

[9] M. Edahiro. Minimum path-length equi-distant routing. In *Proc. IEEE Asia-Pacific Conf. on Circuits and Systems*, pages 41–46, December 1992.

[10] M. Edahiro. A clustering-based optimization algorithm in zero-skew routing. In *Proc. Design Automation Conf*, pages 612–616, June 1993.

[11] A. B. Kahng and C.-W. A. Tsao. Practical bounded-skew clock routing. *Journal of VLSI Signal Processing (Special issue on High Performance Clock Distribution Networks)*, 16(2-3):199–215, June-July 1997.

[12] Y. Liu, S. Nassif, L. Pileggi, and A. Strojwas. Impact of interconnect variations on the clock skew of a gigahertz microprocessor. In *Proc. Design Automation Conf*, 2000.

[13] B. Lu, J Hu, G. Ellis, and H. Su. Process variation aware clock tree routing. In *International Symposium on Physical Design*, pages 174–181, 2003.

[14] V. Mehrotra, S. Sam, D. Boning, A. Chandrakasan, R. Vallishayee, and S. Nassif. A methodology for modeling the effects of systematic within-die interconnect and device variation on circuit performance. In *Proc. Design Automation Conf*, pages 172–175, 2000.

[15] H. Su and S. Sapatnekar. Hybrid structured clock network construction. In *Proc. Int. Conf. on Computer Aided Design*, pages 333 – 336, 2001.

[16] C.-W. A. Tsao and C.-K. Koh. UST/DME: A clock tree router for general skew constraints. *ACM (TODAES)*, 7:359–379, 2002.

[17] D. Velenis, E. Friedman, and M. Papaefthymiou. A clock tree topology extraction algorithm for improving the tolerance of clock distribution networks to delay uncertainty. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 4.422–4.425, May 2001.

[18] J. G. Xi and W. W.-M. Dai. Useful-skew clock routing with gate sizing for low power design. In *Proc. Design Automation Conf*, pages 383–388, 1996.