Skew Scheduling and Clock Routing for Improved Tolerance to Process Variations*

Ganesh Venkataraman, C. N. Sze and Jiang Hu
Dept. of Electrical Engineering, Texas A&M University, College Station, TX 77843,USA
{ganesh, cnsze,jianghu}@ee.tamu.edu

Abstract *The synthesis of clock network in the presence of process variation is becoming a vital design issue towards the performance of digital circuits. In this paper, we propose a clock tree design algorithm which is driven by the tolerance towards process variations. We consider tolerance to process variation in various stages of clock tree synthesis which include clock skew scheduling, abstract tree generation and layout embedding. The primary objective of this work is to minimize the maximum skew violation and a layout embedding technique specifically targeting this objective is detailed. Experimental results indicate the our proposed procedure leads to significant reduction in maximum skew violation due to process variation with negligible change in wire length.*

Keywords: *skew scheduling, clock routing, process variation, layout embedding, reliability.*

I. Introduction

As the VLSI feature size keeps shrinking, clock skew, which is the difference between the clock signal delay to registers, becomes more sensitive to manufacturing process variations [1,2]. It is reported in [2] that interconnect variation alone may cause 25% change on clock skew. Therefore, it is very important to consider the impact of process variations in clock tree synthesis.

The synthesis of clock trees can be divided into two stages: skew scheduling and clock routing. Clock skew scheduling [3, 4, 5, 6] determines the relative clock signal delay target of each clock sink (register), and the objective is usually to minimize the clock period. Clock routing [7, 8, 9, 10] aims at producing a routing tree which fulfills the skew constraints with other objectives such as wire length minimization. Clock tree routing can usually be viewed as recursively merging a set of subtrees (initially a set of sinks) in a bottom-up manner. The subtrees are merged in pairs to create a new subtree while the merging node is the new root. The clock tree routing is accomplished when there is only one tree left. There are two major decisions to be made in the bottom-up process: (1) *abstract tree generation* that tells which subtrees should be merged together; (2) *layout embedding* that decides the locations of the merging points. These two decisions can be made in an integrated manner [11] or separately [8].

Clock skew scheduling can be applied to improve skew tolerance to process variations [3, 4, 5, 6, 12, 13] by allocating skew **safety margin** which is the maximum skew change allowed without affecting circuit functionality. The allocation of the safety margin depends on skew permissible ranges [13]. The work in [13] schedules the skew of each clock sink pair targeting at the middle of the skew permissible range by quadratic programming, so as to increase the safety margin in general cases. But, the work does not consider the location information of the sinks.

An uncertainty driven abstract tree construction algorithm is proposed in [14]. However, its delay model employed is very primitive and physical locations of clock sinks are not considered. For clock tree layout embedding, one of the most influential works is the deferred-merge embedding (DME) algorithm [8]. DME aims at the zero skew clock routing with wire length minimization while it can also be applied to obtain non-zero skew specification. A process variation aware layout embedding algorithm is suggested in [15]. In the work of UST/DME [10], the clock routing is integrated with incremental skew scheduling such that further wire length reduction is obtained compared with DME. However, process variations are not considered in UST/DME.

In order to avoid weakness of the previous works on handling the process variations, we propose new algorithms such that the process variations are considered in both skew scheduling and clock routing. In skew scheduling, an estimation of variations based on clock sink locations is employed to guide skew safety margin towards sink pairs which are more vulnerable to variations. This is in contrast to the work of [13] which attempts to allocate the safety margins evenly regardless of the clock sink locations. In clock routing, an embedding technique is developed to minimize the maximum skew violation among all sink pairs optimally. This is a remarkable improvement with respect to the heuristic variation aware embedding in [15]. Monte Carlo simulation results on benchmark circuits show that our techniques can reduce the variation induced skew violations significantly compared with previous work.

II. Preliminaries

A clock net consists a clock source and a set of clock sinks $S = \{s_1, s_2, ...s_n\}$ each of which corresponds to a register. Two registers (clock sinks) are sequentially adjacent [12] if there is only combinational logic between them. For a pair of sequentially adjacent registers i and j, the clock signal arrival time to them t_i and t_j has to satisfy [3]:

$$t_{hold} - D_{min} \leq t_i - t_j \leq T - t_{setup} - D_{max} \quad (1)$$

where T is the clock cycle time, and D_{max} and D_{min} are the maximum and the minimum combinational logic path delay from i to j, respectively. Therefore, the **clock skew** $t_{ij} = t_i - t_j$ is bounded by $t^l(i,j) = t_{hold} - D_{min}$ and $t^u(i,j) = T - t_{setup} - D_{max}$ which form the **skew permissible range**.

When process variations are considered, the value of t_{ij} is no longer deterministic in an actual clock routing tree. In this

*This work is supported in part by Semiconductor Research Corporation under contract number 2003-TJ-1205 and DAC graduate scholarship.

paper, we focus on the variation of wire width w and each sink load C_i. Other variations can be handled by our work in a similar way. As in [15], we assume the wire width and sink load variations follow normal distributions and the variation is approximately bounded by the 3σ value represented by $W_l \leq w \leq W_u$ and $C_i^l \leq C_i \leq C_i^u$. The **skew violation** due to variations is defined as:

$$t^{vio}(i,j) = \max(t^l(i,j) - t_{ij}, t_{ij} - t^u(i,j)) \qquad (2)$$

The problem we will solve is formulated as follows.

Skew scheduling and clock routing for improved tolerance to process variations Given a set of clock sinks $S = \{s_1, s_2, ...s_n\}$, and a set of skew constraints as shown in equation (1), find the clock arrival time assignment t_i for each clock element s_i and construct a clock tree such that the maximum skew violation among all pairs of clock sinks is minimized while wire width varies between W_u and W_l and load capacitance C_i of each sink s_i varies between C_i^l and C_i^u.

As in the previous works [9, 10, 15], we employ the Elmore delay model. Although the Elmore delay is sometimes inaccurate, it has a high fidelity [16]. Moreover, the Elmore delay based solutions can at least serve as basis for further tuning with more accurate models.

III. The Algorithm

A. Algorithm Overview

Our algorithm consists of two major parts: (1) variation aware skew scheduling and (2) variation aware clock routing. The algorithm overview is also provided in Figure (1).

In the variation aware skew scheduling, an estimated skew variation between each pair of sequentially adjacent registers is computed based on their locations. Then, a linear programming based skew scheduling algorithm is performed to maximize the relative skew safety margin according to the estimated skew variations. The delay target to each register is obtained after the scheduling.

The variation aware clock routing is a procedure of interleaving abstract tree generation with layout embedding. The abstract tree generation or the merging scheme is similar to [17] and consists of two steps: (1) finding a sink node v_r (which is a part of the subtree $T(v_i)$) with the maximum delay target and (2) finding the other sink node v_l (which is a part of subtree $T(v_j)$) to be merged with $T(v_i)$ such that the merging cost is minimized. The merging cost will be defined later. The layout embedding is based on DME with consideration on process variations. The merging segment between subtrees $T(v_i)$ and $T(v_j)$ is found such that the maximum skew violation between sinks of two subtrees is minimized.

B. Process Variation Aware Skew Scheduling

In skew scheduling, delay target t_i to each register i is determined while the permissible range of (1) is satisfied. In order to improve tolerance to process variations, the skew safety margin $\min(t^u(i,j) - t_{ij}, t_{ij} - t^l(i,j))$ needs to be maximized for each pair of sequentially adjacent registers. Since the distance between each pair of registers may be different from each other, the skew variation for each pair is usually different from each other as well. Therefore, different amount of safety margin should be allocated for different register pairs. The objective of our skew scheduling scheme is to maximize skew safety margin with consideration of skew variation between

```
Algorithm Overview
Clock Scheduling and Routing
Input : clock sinks, initial skew permissible range
Output : clock tree routing
begin
1.      Variation aware skew scheduling
1.1         Estimate skew guarding band for
            each pair of sequentially adjacent registers
1.2         Find delay target to each register
            such that variation aware safety
            margin is maximized
2.      Variation aware clock routing
        Subtree set $B = \{s_1, s_2, ..., s_n\}$
        While ( $|B| > 1$ )
2.1         Abstract tree generation
2.1.1           Select sink $v_r$ (part of subtree $T(v_i)$)
                with max delay target
2.1.2           Select another sink $v_l$ (part of subtree $T(v_j)$)
                to be merged such that the merging
                wire length is minimized
2.2         Variation aware embedding
            Find merging segment for $T(v_i)$ and $T(v_j)$
            such that $t^{vio}(i,j)$ between each pair
            $m \in T(v_i)$and $n \in T(v_j)$ is minimized
2.3         $B \leftarrow B - (T_l, T_r) +$ Newly formed subtree
        end while
```

Fig. 1.: Scheduling and routing algorithm

registers. If the estimated skew variation between register i and j is δ_{ij}, we formulate the scheduling problem as:

$$\text{Maximize} \qquad M \qquad (3)$$
$$\forall \text{ seq adj reg } i \text{ and } j$$
$$M \leq t^u(i,j) - \delta_{ij} - (t_i - t_j)$$
$$M \leq (t_i - t_j) - t^l(i,j) - \delta_{ij}$$

where M is the shared safety margin for each pair. This formulation may allocate safety margins according to anticipated variation effect instead of an uniform allocation in previous works [3, 6, 13]. This linear programming problem can be solved by the binary search method introduced in [4].

The estimated skew variation is obtained based on distance between registers since the clock routing information is not available in the scheduling stage. For two registers i and j, we assume they are merged at the middle point between them and use the skew variation from this merging as an estimation. Let wire resistance and capacitance coefficient be r and c, respectively. Then a wire segment of length l driving a load of C_l has delay of $\frac{1}{2}rcl^2 + \frac{rl}{w}C_L$. If wire width w varies around nominal value w_0 with bound of $w_0 \pm dw$ and sink load varies as $C_{Li} = C_{0Li} \pm dC_{Li}$, the skew variation between i and j is estimated to be:

$$\delta_{ij} = \frac{D_{ij} r}{2w_0}(dC_{Li} + \frac{C_{Li} \cdot dw}{w_0} + dC_{Lj} + \frac{C_{Lj} \cdot dw}{w_0})$$

where D_{ij} is the Manhattan distance between register i and j. Even though this estimation is an approximation, it reflects the trend that variability may grow with distance.

C. Process Variation Aware Layout Embedding

In this section, we detail a layout embedding scheme that aims to minimize the maximum violation in skew due to pro-

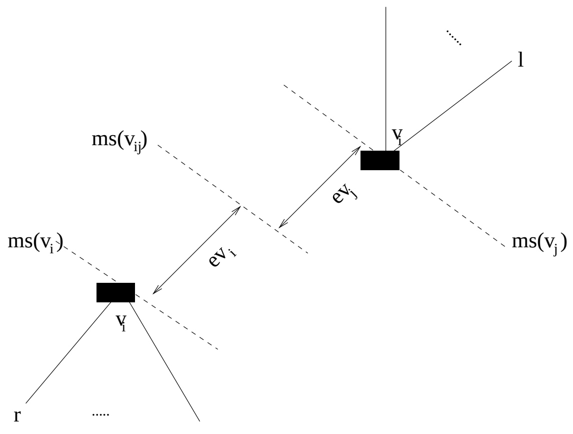

Fig. 2.: **Merging and Embedding of subtrees**

cess variation. Process variation aware embedding was handled in [15], where a heuristic was used to identify a single critical pair and the layout embedding focussed on this pair alone. In our work, we propose a formulation that considers the effect due to variation in all pairs.

Our embedding follows the DME [8] framework. DME consists of a bottom-up phase where the locus of possible merging points is identified to match the required skew constraints. The top-down phase identifies the actual location of the merging points. When two subtrees are merged to achieve zero skew, the set of possible points of the new parent node has the property that the distances between the parent and the root of the individual subtrees are fixed values. Hence the locus corresponds to a Manhattan arc commonly referred to as the **merging segment** [8]. We introduce some basic notation that will be followed in the paper. Figure (2) illustrates merging/embedding of two nodes v_i and v_j. Here $ms(v_i)$ (in dashed lines) refers to the merging segment at node v_i of the clock tree. $T(v_i)$ refers to the entire subtree rooted at v_i. v_{ij} is the parent node formed by merging $T(v_i)$ and $T(v_j)$. e_{v_i}, e_{v_j} and D_{ij} denote the Manhattan distances between $ms(v_i)$ and $ms(v_{ij})$, $ms(v_i)$ and $ms(v_{ij})$, $ms(v_j)$ and $ms(v_j)$ respectively.

Consider two subtrees $T(v_i)$ and $T(v_j)$ to be merged. In our approach, we shall show that there exists a locus of points that has fixed distances from $ms(v_i)$ and $ms(v_j)$ such that it minimizes the maximum skew violation. Maximum skew violation is defined as the maximum deviation of the skew from its allowed bounds among all sequentially adjacent register pairs. Let T^{vio}_{max} denote the maximum skew violation. Then:

$$T^{vio}_{max} = \max_{(i,j)}(t^l(i,j) - t_{ij}, t_{ij} - t^u(i,j)) \tag{4}$$

Ideally, we would like T^{vio}_{max} to be negative implying that there is no violation. We first formally state the problem that is addressed in this section.

Process Variation Aware Layout Embedding
Given two subtrees $T(v_i)$ and $T(v_j)$ to be merged, find a parent merging segment $ms(v_{ij})$ such that the maximum skew violation among all sequentially adjacent pairs (s_r, s_l) where $s_r \in T(v_i)$ and $s_l \in T(v_j)$ is minimized.

The above problem can also be formulated mathematically as described below. For $s_r \in T(v_i)$ and $s_l \in T(v_j)$, let $\overline{t_{rl}}(e_{v_i}, e_{v_j})$ ($\underline{t_{rl}}(e_{v_i}, e_{v_j})$) denote the maximum (minimum) skew between s_r and s_l due to process variation. Let t_{rl} denote the nominal skew value. We intend to minimize the maximum skew violation, or equivalently maximize the minimum difference between the skew bounds and the worst case skew among all pairs. This is similar to the scheduling described in section 3.2 and can be written as:

$$\begin{aligned} &\text{Maximize} \quad S_{ij} \\ &\forall\ r \in T(v_i) \text{ and } l \in T(v_j) \\ &S_{ij} \le t^u(r,l) - \overline{t_{rl}}(e_{v_i}, e_{v_j}) \\ &S_{ij} \le \underline{t_{rl}}(e_{v_i}, e_{v_j}) - t^l(r,l) \end{aligned} \tag{5}$$

where S_{ij} is a variable similar to the safety margin in skew scheduling. Maximizing S_{ij} is equivalent to minimizing skew violation.

The procedure to evaluate $\overline{t_{rl}}(e_{v_i}, e_{v_j})$ and $\underline{t_{rl}}(e_{v_i}, e_{v_j})$ under variation in wire-width and load capacitances was detailed in [15]. We shall state it in brief and show how formulation (5) can be transformed into a mathematical programming problem with quadratic constraints. We also outline a technique which can be used to solve this problem. Though we concentrate on wire-width and load capacitance variations, the formulation given in (5) is generic and can be easily modified to take care of other variations as well.

$(\underline{t_r}, t_r, \overline{t_r})$ denote the minimum, nominal and maximum delays from s_r (which is part of the subtree $T(v_i)$) to the node v_i. (C^l_i, C_i, C^u_i) represent the minimum, nominal and maximum downstream capacitances at node v_i respectively. Let $\alpha^l_i(\alpha^u_i) = r\frac{C^l_i}{W_u}(r\frac{C^u_i}{W_l})$ and $\phi = \frac{1}{2}rc$. Then:

$$\overline{t_{rl}}(e_{v_i}, e_{v_j}) = (\overline{t_r} + \phi e^2_{v_i} + \alpha^u_i e_{v_i}) - (\underline{t_l} + \phi e^2_{v_j} + \alpha^l_j e_{v_j}) \tag{6}$$

W_u and W_l denote the bounds on the wire width. Interested reader may refer to [15] for detailed derivation of the these equations.

Similarly $\underline{t_{rl}}(e_{v_i}, e_{v_j})$ may be written as:

$$\underline{t_{rl}}(e_{v_i}, e_{v_j}) = (\underline{t_r} + \phi e^2_{v_i} + \alpha^l_i e_{v_i}) - (\overline{t_l} + \phi e^2_{v_j} + \alpha^u_j e_{v_j}) \tag{7}$$

We can transform the formulation (5) into a mathematical programming problem using equations (6) and (7).

$$\begin{aligned} &\text{Maximize } S_{ij} \\ &\forall\ r \in T(v_i) \text{ and } l \in T(v_j) \\ &S_{ij} \le (t^u(r,l) - \overline{t_r} + \underline{t_l}) - \phi(e^2_{v_i} - e^2_{v_j}) - (\alpha^u_i e_{v_i} - \alpha^l_j e_{v_j}) \\ &S_{ij} \le (-t^l(r,l) - \overline{t_l} + \underline{t_r}) + \phi(e^2_{v_i} - e^2_{v_j}) + (\alpha^l_i e_{v_i} - \alpha^u_j e_{v_j}) \\ &e_{v_i}, e_{v_j} \ge 0 \quad e_{v_i} + e_{v_j} \ge D_{ij} \end{aligned} \tag{8}$$

The above formulation is quite different from the work in [15] where a single pair of critical sinks were identified. After this, embedding was done in such a way that the center of the worst case skew matches with the center of the permissible range. However, this is a heuristic and there could be other pairs (other than the critical one) which could face a significant deviation in skew due to process variations. Formulation (5) attempts to overcome this difficulty. The parent merging segment cannot exist if $(e_{v_i} + e_{v_j} < D_{ij})$ and the final constraint takes care of that. The formulation appears to be a difficult one to solve because of the presence of quadratic constraints. But, we can consider two separate cases: case 1 does not require any wire snaking and case 2 requires wire snaking. In each of these cases, the formulation can be transformed in to another one with linear constraints. The method to identify the need for wire snaking has been presented in [15] and will not be repeated here due to space constraints.

Case 1: No wire snaking
In this case, $e_{v_i} + e_{v_j} = D_{ij}$. Define the following parameters:

$$m_1(r,l) = (t^u(r,l) - \overline{t_r} + \underline{t_l}) + \phi D_{ij}^2 + \alpha_j^l D_{ij} \tag{9}$$
$$m_2(r,l) = (-t^l(r,l) - \overline{t_l} + \underline{t_r}) - \phi D_{ij}^2 - \alpha_j^u D_{ij}$$
$$k_1 = 2\phi D_{ij} + \alpha_i^u + \alpha_j^l, \quad k_2 = 2\phi D_{ij} + \alpha_j^u + \alpha_i^l$$

With no snaking formulation(8) gets modified in to the following linear programming problem with just one variable:

$$\text{Maximize } S_{ij} \tag{10}$$
$$\forall\ r \in T(v_i) \text{ and } l \in T(v_j), \quad S_{ij} \le m_1(r,l) - k_1 e_{v_i}$$
$$S_{ij} \le m_2(r,l) + k_2 e_{v_i}, \quad e_{v_i} \ge 0$$

Notice that the constraints in (10) represent straight lines with a positive slope of k_2 or the negative slope of $-k_1$. For the same pair of subtrees $T(v_i)$ and $T(v_j)$, k_1 and k_2 are the same for all (r,l). We have two sets of lines with lines in the same set parallel to one another. Figure (3) gives a plot of the lines when there are two such pairs. The upper half of the figure gives 4 straight lines that represent the 4 inequality constraints (in (10)) corresponding to the 2 pairs. The lower half plots the values of S_{ij} as e_{v_i} is increased. Let $m_1(r_1,l_1) = \min_{r\in T(v_i),l\in T(v_j)} m_1(r,l)$ and $m_2(r_2,l_2) = \min_{r\in T(v_i),l\in T(v_j)} m_2(r,l)$.

Lemma 1
If no snaking is required, that is $0 \le e_{v_i}, e_{v_j} \le D_{ij}$, then the optimal solution to (10) exists at a point

$$e_{v_i} = \frac{m_1(r_1,l_1) - m_2(r_2,l_2)}{k_1 + k_2} \tag{11}$$

The proof is Lemma 1 is straightforward and Figure (3) gives the geometric intuition behind the proof.

Case 2: With wire snaking
Wire snaking could mean either $(e_{v_i} = 0, e_{v_j} > D_{ij})$ or $(e_{v_i} = 0, e_{v_j} > D_{ij})$. We discuss the latter and the equations can be modified to fit the former. Let $m_1^{'}(r,l) = (t^u(r,l) - \overline{t_r} + \underline{t_l})$ and $m_2^{'}(r,l) = (-t^l(r,l) - \overline{t_l} + \underline{t_r})$. When we substitute $e_{v_i} = 0$ in formulation (8) we get:

$$\text{Maximize } S_{ij} \tag{12}$$
$$\forall\ r \in T(v_i) \text{ and } l \in T(v_j)$$
$$S_{ij} \le m_1^{'}(r,l) + \phi e_{v_j}^2 + \alpha_j^l e_{v_j}$$
$$S_{ij} \le m_2^{'}(r,l) - \phi e_{v_j}^2 - \alpha_j^u e_{v_j}, \quad e_{v_j} \ge 0$$

The optimal solution for (12) is obtained in a manner similar to that discussed for case 1. Here, we have non-linear terms. But still the coefficients of e_{v_j} and $e_{v_j}^2$ are the same for all (r,l). Let $m_1{}'(r_1,l_1) = \min_{r\in T(v_i),l\in T(v_j)} m_1^{'}(r,l)$ and $m_2{}'(r_2,l_2) = \min_{r\in T(v_i),l\in T(v_j)} m_2^{'}(r,l)$.

Lemma 2
If wire snaking is required with $e_{v_i} = 0$ and $e_{v_j} > D_{ij}$, then the optimal solution to (12) exists at a point e_{v_j} that satisfies the following equation:

$$m_1^{'}(r_1,l_1) + \phi e_{v_j}^2 + \alpha_j^l e_{v_j} = m_2^{'}(r_2,l_2) - \phi e_{v_j}^2 - \alpha_j^u e_{v_j} \tag{13}$$

D. Alternative Layout Embedding

An alternative embedding procedure was also performed to make a comparison with our proposed scheme. This method closely follows [15] with a key difference. In [15], one critical pair was selected using a weighted function that considers both the physical distance as well as the permissible range.

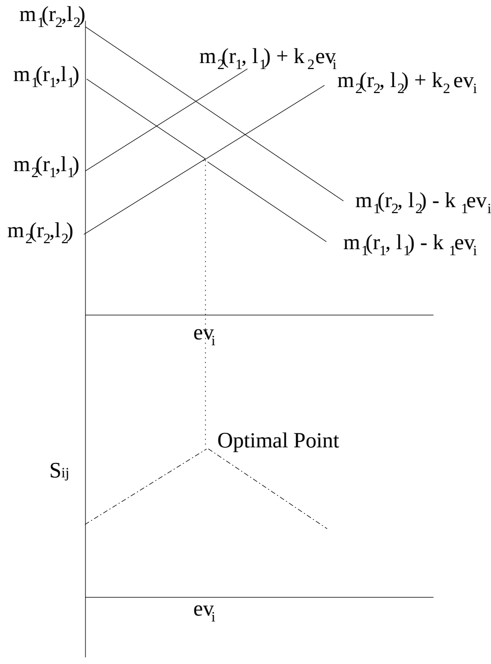

Fig. 3.: **Plot of S_{ij} with two pairs**

Here we consider the feasible skew range (FSR) [10] instead of the permissible range. The FSR matrix is updated after every merging. After such an updation, it is possible that the feasible skew range of two sinks (say s_r and s_l) gets reduced. Interested reader may refer to [10] for a detailed description of the FSR matrix and the process used to update it after every merge. In such a scenario, we reduce the range by an additional factor (δ_{rl}) whose value equal the difference between the maximum and the nominal skew multiplied by a scaling factor. This way, we have an incremental scheduling scheme that reflects the effect due to process variation.

E. Abstract Tree Generation

We first select a sink node (s_m in a subtree rooted at v_i) which has the maximum delay target. The second node is selected based on basis of expected wire length added. Consider a sink node s_n in a subtree rooted at (v_j). We compute the values of e_{v_i} and e_{v_j} by the following procedure. To compute the actual values of e_{v_i} and e_{v_j}, we first need to find two sink pairs (r_1,l_1) and (r_2,l_2) such that $m_1(r_1,l_1) = \min_{r\in T(v_i),l\in T(v_j)} m_1(r,l)$ and $m_2(r_2,l_2) = \min_{r\in T(v_i),l\in T(v_j)} m_2(r,l)$. This could be prohibitingly expensive for all pairs. So we use the heuristic where we assign $(r_1,l_1) = (r_2,l_2) = (m,n)$. Then by using the procedure detailed earlier, we identify the need for snaking and use equation (11) or (13) to compute e_{v_i} and e_{v_j}. This serves as the merging cost. We claim that such a heuristic performs better and have experimental results to support the claim. This procedure will be referred to as *embedding aware abstract topology construction*.

F. Algorithm Complexity

The complexity of the process variation aware scheduling is $O(nm)$ where n is the number of sinks and m is the number of constraints for the linear programming formulation. The complexity of each embedding is $O(n)$ and the total runtime for $O(n)$ embeddings is $O(n^2)$. Even though $m = O(n^2)$ in the-

Case	#Sinks	BASE CASE			
		WireLen	#Vio	Max Vio	Run time
r1	267	1971347	0.206	3.434	00:01
r2	598	3849918	18.60	69.91	00:06
r3	862	4591019	33.37	88.52	00:13
r4	1903	9531821	553.389	573.64	00:51
r5	3101	14391465	1523.3	1392.39	02:31

TABLE I: **Results for the base case**

ory, in practice $m = O(n)$. Therefore, the overall complexity of our algorithm is $O(n^2)$.

IV. Experimental Results

The proposed procedure was tested on five benchmark circuits [7]. The values of the technology parameters used were as follows: nominal wire width = 0.18 μm, per unit resistance = 0.0042 Ω and per unit capacitance = 3.18-6pF/μm^2. The benchmark circuits do not contain the timing information (permissible range for each sink pair). So, we generated the permissible ranges randomly such that for every (s_r, s_l), $t^l(r,l) < 0 < t^u(r,l)$. The permissible ranges were not symmetric but chosen such that $||t^u(r,l)| - |t^l(r,l)|| < 100ps$, $t^l(r,l) \in (-600ps, -100ps)$ and $t^u(r,l) \in (100ps, 600ps)$. The code was written in C and run on a Sun Solaris Ultra Sparc machine with 2 GB RAM.

We ran our experminents for five methods (based on scheduling, topology and embedding) denoted as:

- BASE CASE: No scheduling, topology based on distance, embedding follows [15]
- SP-DISTANCE: Process variation aware scheduling, topology based on distance, embedding follows [15]
- MP-DISTANCE: Process variation aware scheduling, topology based on distance, the new embedding scheme
- SP-eAWARE: Process variation aware scheduling and topology, embedding follows [15]
- MP-eAWARE: Process variation aware scheduling and topology, the new embedding scheme

In the above mentioned methods, BASE CASE is almost similar to [15] and MP-eAWARE represents our complete solution proposed. Expermiments were run on the remaining three methods to show the effect of each individual technique, thereby providing a better insight. Monte Carlo simulations (1000 runs) were done for all five methods. In each run, the wire width and load capacitance were chosen at random from a uniform distribution with mean being the nominal value and standard deviation set such that 3σ corresponds to 10% deviation from the nominal value. The results for all variants different methods are tabulated in Tables I, II and III. Table I shows the results for BASE CASE, Table II for SP-DISTANCE and MP-DISTANCE, Table III for SP-eAWARE and MP-eAWARE. The sub columns represent wire length (WireLen), Number of violations (#Vio), average maximum violation in ps (Max Vio) Run time (in min:sec) and Improvement (of maximum violation in percentage compared to the base).

As it is evident from the Table III there is a significant difference in the maximum skew violation in our proposed scheme and the base case. Moreover (MP-DISTANCE, MP-eAWARE) perform better than (SP-DISTANCE, SP-eAWARE) highlighting the effectiveness of our embedding technique. Our primary focus is to reduce the maximum skew violation and not number of violations since the former has a greater impact on the choice of the clock frequency. For example consider two scenarios: case (a) 50 violations with a maximum violation of 25 ps and case (b) 1 violation of 50 ps. Case (a) could be handled by slowing the clock by 25 ps whereas case (b) would require a reduction of 50 ps in clock speed. Nevertheless, proposed technique results in a significant gain in the number of violations as well.

V. Conclusion

In this paper, both skew scheduling and clock routing algorithms are proposed to improve skew tolerance to process variations. In the proposed skew scheduling algorithm, process variations are considered directly in skew safety margin allocations so that a larger safety margin can be obtained for registers far apart. In the clock routing algorithm, a new layout embedding technique is developed to optimally minimize the maximum skew violation due to process variations. The effectiveness of the proposed algorithms is validated through Monte Carlo simulations on benchmark circuits.

References

[1] S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena, and C. Guardiani, “Analysis of the impact of process variations on clock skew,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 13, no. 4, pp. 401–407, 2000.

[2] Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas, “Impact of interconnect variations on the clock skew of a gigahertz microprocessor,” in *DAC 2000*, pp. 168–171.

[3] J. P. Fishburn, “Clock skew optimization,” *IEEE Transactions on Computers*, vol. 39, no. 7, pp. 945–950, 1990.

[4] R. B. Deokar and S. S. Sapatnekar, “A graph-theoretic approach to clock skew optimization,” in *ISCAS 1994*, pp. 407–410.

[5] A. Takahashi and Y. Kajitani, “Performance and reliability driven clock scheduling of sequential logic circuits,” in *ASP-DAC 1997* pp. 37–42.

[6] C. Albrecht, B. Korte, J. Schietke, and J. Vygen, “Cycle time and slack optimization for VLSI-chips,” in *ICCAD 1999* pp. 232–238.

[7] R. S. Tsay, “An exact zero-skew clock routing algorithm,,” *IEEE TCAD 1993*, vol. 12, pp. 242–249.

[8] T.-H. Chao, Y.-C. Hsu, J.-M.Ho, K. D. Boese and A. B. Kahng, “Zero Skew Clock Routing with Minimum Wirelength,” in *IEEE Transactions on Circuits and Systems-II*, vol. 39, no. 39, pp. 799–814, 1992.

[9] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, “Bounded-skew clock and Steiner routing,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 3, no. 3, pp. 341–388, 1998.

Case	SP-DISTANCE					MP-DISTANCE				
	WireLen	#Vio	Max Vio	Run time	Improv.(%)	WireLen	#Vio	Max Vio	Run time	Improv.(%)
r1	1866923	0.007	0.0653	00:03	98.11	1831421	0.001	0.0047	00:01	99.85
r2	3844554	6.983	42.57	00:16	39.11	3800512	2.93	17.80	00:06	74.54
r3	4385608	27.55	93.87	00:40	-6.04	4367335	15.95	58.73	00:14	33.65
r4	9248079	516.73	544.63	02:40	5.07	9328236	521.49	488.19	00:56	14.90
r5	14368114	1491.21	1383.35	05:09	0.65	14441731	1474.21	1251.36	02:20	10.13

TABLE II: **Single pair and multiple pair embedding schemes -when abstract topology is distance based**

Case	SP-eAWARE					MP-eAWARE				
	WireLen	#Vio	Max Vio	Run time	Improv.(%)	WireLen	#Vio	Max Vio	Run time	Improv.(%)
r1	1810806	0.015	0.156	00:03	95.46	1810911	0.0	0.0	00:01	100.00
r2	3740438	2.013	20.139	00:14	71.19	3718947	0.747	7.37	00:06	89.46
r3	4375701	30.993	85.721	00:39	3.16	4377841	16.27	69.19	00:14	21.84
r4	9246696	496.53	522.21	02:31	8.97	9213758	457.41	483.28	00:55	15.75
r5	13963649	1422.69	1228.44	04:57	11.77	14165250	1337.42	1128.81	02:21	18.93

TABLE III: **Single pair and multiple pair embedding schemes -when abstract topology is embedding aware**

[10] C.-W. A. Tsao and C.-K. Koh, "UST/DME: a clock tree router for general skew constraints," *ACM Transactions on Design Automation of Electronic Systems.*, vol. 7, no. 3, pp. 359–379, 2002.

[11] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," in *DAC 1993*, pp. 612–616.

[12] J. L. Neves and E. G. Friedman, "Optimal clock skew scheduling tolerant to process variations," in *DAC 1996*, pp. 623–628.

[13] I. S. Kourtev and E. G. Friedman, "Clock skew scheduling for improved reliability via quadratic programming," in *ICCAD 1999*, pp. 239–243.

[14] D. Velenis, M. C. Papaefthymiou, and E. G. Friedman, "Reduced delay uncertainty in high performance clock distribution networks," in *DATE 2003* ,pp. 68–73.

[15] B. Lu, J. Hu, G. Ellis, and H. Su, "Process variation aware clock tree routing," in *ISPD 2003* pp. 174–181.

[16] K. D. Boese, A. K. Kahng, B. A. McCoy, and G. Robins, "Near-Optimal Critical Sink Routing Tree Constructions," in *IEEE TCAD 1995*, vol. 14, no. 12, pp. 1417–1436.

[17] R. Chaturvedi and J. Hu, "A simple yet effective merging scheme for prescribed-skew clock routing," *ICCD 2003*, pp. 282-287.