ABSTRACT
The existence of non-uniform thermal gradients on the substrate in high performance IC’s can significantly impact the performance of global on-chip interconnects. This issue is further exacerbated by the aggressive scaling and other factors such as dynamic power management schemes and non-uniform gate level switching activity.

In high-performance systems, one of the most important problems is clock skew minimization since it has a direct impact on the maximum operating frequency of the system. Since clocks are routed across the entire chip, the presence of thermal gradients can significantly alter their characteristics because wire resistance increases linearly as the temperature increases. This often results in failure to meet original timing constraints thereby rendering the original topology unusable. Therefore it is necessary to perform a temperature aware re-embedding of the original topology to meet timing under these temperature effects. This work primarily explores these issues by proposing two algorithms that re-structure an existing clock tree topology to compensate for such temperature effects and as a result also meet timing constraints.

1. Introduction
In the last years, the power and integration densities in nanometer CMOS devices have dramatically increased, resulting in a huge increase in power density. For example, power density in microprocessors has doubled every three years [1]. Decreasing the power consumption of a chip is the most intuitive way to reduce power density; unfortunately, typical techniques to reduce power consumption tend to produce temperature gradients across the chip. (e.g., dynamic power management, clock-gating) tend to increase the variance of power dissipation over the chip area, which generally translates into many areas with different power densities. Such zones may thus generate temperature gradients in the substrate and, as a consequence, non-uniform substrate temperature. As reported in [16], temperature gradients of 50°C may exist across the substrate in high-performance ICs. The non-uniformity in the substrate temperature has an impact on the global interconnect. This phenomenon, for nanometer scale high performance ICs, may become critical from the performance and reliability perspective [3]. In fact, a temperature rise increases the interconnect delay degrading the circuit performance. This is mainly due to the linear dependency that exists between temperature and resistivity of metal interconnect. Moreover, a high temperature in the metal layers degrades the IC reliability. This is primarily due to electro-migration (EM) and the self-heating phenomena [2]. For these reasons, the effects of non-uniform interconnect thermal profile, caused by non-uniform temperature gradients over the substrate on the signal performance, i.e. signal delay and clock skew, cannot be yet neglected, mostly during performance optimization phases. As clock nets are the most active nets in a circuit, the effects of thermal gradients are even more prominent. Hence, it is not realistic to generate clock tree structures without taking into account temperature variations.

In [15] a technique for the design of a temperature-aware clock distribution network clock-tree was proposed that addresses the drawbacks of traditional clock-tree routing that assume a uniform thermal profile. This work, although quite effective, lacks in generality as it deals only with clock-trees with zero-skew (i.e., all paths from root to the leaves with same delay). In practice, however, circuits still operate correctly within some non-zero skew bound; this extra degree of freedom can be in fact exploited to obtain clock trees with smaller total wirelength.

In this paper we propose a new approach that, moving from the analysis of [15], generalizes it in two ways. First, our approach is more general in that it allows non-zero skew (while [15] supports zero-skew only). Second, our scheme is minimally intrusive because it is based on a restructuring of a given clock tree; such a post-processing allows localized re-design of the clock tree, without modifications of “safe” portions of the clock tree. The latter issue is also beneficial from the point of view of the compatibility with standard P&R tools: Working as a post-processing tool, our approach can, in principle, be applied on any type of clock network.

We present two algorithms with different speed/efficiency tradeoff that allowed to reduce, on a set of standard benchmarks, the worst case skew by 56% on average with a worst case increase in wirelength of less than 1%.

2. Previous Work
The design of the clock network has been the subject of intensive research in the past. Early approaches put the algorithmic basis for solving the basic clock routing problem, i.e., determining a clock topology (in almost all approaches, a tree) with zero skew, while minimizing total wire-length ([4, 5]). These solutions to “basic” clock routing has been eventually extended to deal with non-zero skew ([6, 7]), possibly combined with wire sizing and/or with buffer insertion ([8, 9, 10]). All these approaches have wire-length (as a metric for delay) as optimization criterion, with skew being a constraint.

The increased impact of interconnect on the power budget of a design has spun another set of approaches that rely on algorithms similar to the previous ones, yet targeting a different cost function, namely, the power dissipation of the clock network [11, 12]. Dealing with selective clock throttling mechanisms typical of low-power designs (e.g., clock gating) requires instead specialized algorithms [13, 14].
All these delay- or power-driven approaches are based on the assumption that temperature is constant along the clock network. The work by Ajami et al. [3] was the first one to address the problem of how temperature gradients affect clock skew. In their work, they provide a characterization of the problem and some experimental data on how different gradient affect clock skew.

The only work explicitly dealing with temperature-related effects while designing the clock network has been proposed recently by Cho et al. [15]. In that work, they propose an adaptation of the basic DME algorithm [5] in such a way that it can deal with skew variations due to temperature gradients along a clock trunk. Our work is similar to [15] in scope, yet it generalizes that approach in that (i) it can deal with a non-zero skew bound, and (ii) it builds a clock tree by performing a local post-processing optimization on a given clock-tree, therefore minimizing the amount of modifications to original tree structure.

3. Background

3.1 Design of Clock Distribution Network

The problem of the design of the clock distribution network consists of (i) defining a topology of the clock network, and (ii) routing the clock net to sinks (corresponding to storage elements, e.g., flip-flops), starting from a clock source, while minimizing a given cost (total wire-length, power dissipation) under some constraints on the maximum skew. The skew of a clock distribution network is defined as the maximum delay difference between any two (source,sink) pairs. In this work, as in most clock routing algorithms, we focus on binary tree topologies. Once the topology is defined, the routing the clock net to the sinks can be accomplished using various approaches; the most relevant one is the Deferred-Merge Embedding (DME) algorithm [5], which we will briefly review because it is the basis of our approach. The DME algorithm embeds an existing topology in the Manhattan plane by means of a two-step process. In the first step, the set of points feasible for placement of the internal nodes of the clock tree are generated bottom-up: These are called merging segments, and consist of a tree of line segments (for a zero-skew tree). The second step determines, in a top-down fashion, the exact position of the internal nodes on the tree of merging segments, by choosing the point that minimizes the desired cost function. DME can be easily adapted to non-zero skew clock trees [9]. In this case, merging segments become merging regions, so that more solutions are now available.

3.2 Thermal Effects on Clock Interconnect

Variations in temperature along long global wires affects their performance because wire resistance is temperature-dependent (whereas wire capacitance can be considered essentially temperature-independent), and precisely linearly dependent on temperature that is, \( R = R_0(1 + \beta T(x)) \), where \( R_0 \) is the resistance per unit-length at a reference temperature, \( \beta \) is the temperature coefficient of resistance, and \( T(x) \) is the temperature profile along the wire. Resistance increases thus in wire sections that exhibit higher temperatures; under an Elmore delay model, this increase translates into an increase of the delay of the wire. In the context of clock routing, the temperature-dependent delay impacts clock skew because the position of clock insertion points under a non-uniform thermal profile do not guarantee the desired skew bound under a different, non-uniform thermal profile. Figure 1 summarizes the principle that drives the placement of clock insertion points [3] while designing a temperature-aware clock tree.

4. Thermal-Aware Clock Tree Generation

4.1 Problem Formulation

We define the problem we are solving as follows: Given an existing clock tree \( T \) with skew bound \( B_0 \) and wirelength \( WL \), which under a non-uniform thermal profile \( T \) has a skew of \( B_T \geq B_0 \), modify \( T \) into a new tree \( T' \) so that the clock skew of this new tree is minimized for both uniform and thermal profiles with minimum wirelength overhead.

It is important to emphasize that our approach post-processes an existing clock tree with minimal impact on it, and does not design the tree from scratch. This preserves the quality of various optimization objectives which might have gone into design of the original clock tree. We propose two algorithms with different tradeoffs between execution time and optimization potential. Before describing the details, we will first introduce some terminology and notation.

4.2 Terminology and Notation

Table 1 lists the notations used. Quantities related to a non-uniform thermal profile \( T \) (and thus to new tree \( T' \)) are denoted by symbols with the ”T” subscript. \( ED_T \) denotes the temperature dependent Elmore delay, as defined by Equation 1 defined in [3].

\[
ED_T = ED + (c_0 \cdot L + C_L) \rho_0 \beta \int_0^L T(x)dx - c_0 \rho_0 \beta \int_0^L x \cdot T(x)dx
\]

where \( ED = \frac{\rho_0 c_0 L^2}{2} + \rho_0 L C_L \) is the Elmore Delay, \( \beta \) is the
Table 1: Basic Definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR</td>
<td>Clock tree under uniform profile $T=I_0$</td>
</tr>
<tr>
<td>TR$_T$</td>
<td>Clock tree under thermal profile $T(X)$</td>
</tr>
<tr>
<td>MAX(p)</td>
<td>Maximum delay of the sub-tree rooted at $p$</td>
</tr>
<tr>
<td>MIN(p)</td>
<td>Minimum delay of the sub-tree rooted at $p$</td>
</tr>
<tr>
<td>L(p), R(p)</td>
<td>Left/Right children of a node $p$</td>
</tr>
<tr>
<td>B</td>
<td>Skew bound of original clock tree $TR$</td>
</tr>
<tr>
<td>$B_T$</td>
<td>Modified skew for tree $TR_T$</td>
</tr>
<tr>
<td>WL</td>
<td>Wirelength of original clock tree</td>
</tr>
<tr>
<td>$WL_T$</td>
<td>Modified wirelength for tree $TR_T$</td>
</tr>
<tr>
<td>$ED_T$</td>
<td>Elmore delay at reference temperature</td>
</tr>
<tr>
<td>$ED_T$</td>
<td>Temperature-dependent Elmore delay</td>
</tr>
<tr>
<td>$c_0$, $p_0$</td>
<td>Capacitance/Resistance per unit length</td>
</tr>
<tr>
<td>$L_e$</td>
<td>Load capacitance as seen by the wire of length $L$</td>
</tr>
<tr>
<td>$SKEW_T(p)$</td>
<td>Original skew for a node $p$ at uniform temperature profile</td>
</tr>
<tr>
<td>$SKEW_T(p)_{up}$</td>
<td>Modified skew for a node $p$ under up temperature profile $T(X)$</td>
</tr>
<tr>
<td>$SKEW_T(p)_{down}$</td>
<td>Modified skew for a node $p$ under down temperature profile $T(X)$</td>
</tr>
<tr>
<td>$e_{u,v}$</td>
<td>Edge connecting a node $u$ to node $v$</td>
</tr>
<tr>
<td>$d_{u,v}$</td>
<td>Manhattan distance between $u$ and $v$</td>
</tr>
</tbody>
</table>

The skew $SKEW_T(p)$ for a generic node $p$ of the tree with children $u$ and $v$ is defined recursively as:

$$SKEW_T(p) = \max \{ \max[ED_T(e_{u,p}) + MAX(u), ED_T(e_{v,p}) + MAX(v)] - \min[ED_T(e_{u,p}) + MIN(u), ED_T(e_{v,p}) + MIN(v)] \}$$

4.3 Temperature Profiles

The two proposed algorithms are fairly general in nature, and work for any thermal profile. We ran our experiments assuming three different thermal profiles. The first one has a uniformly increasing temperature along the width of the chip. The other two profiles were generated by superimposing some of the thermal maps found in existing microprocessors. The first one has higher number of total clustered regions of particular temperature whereas, the second profile has fewer but bigger cluster of temperature profile. The latter two profiles are depicted in Figure 2.

Figure 2: Second and Third Temperature Profiles.

4.4 Solution Region Based Algorithm (SRB)

The SRB algorithm works as in BST/DME by constructing Solution Regions (SR) first in a bottom-up phase and then embedding the best solution for each point in the subsequent top-down phase. This ensures that the modified tree has minimum wirelength penalty since it works in the vicinity of the optimal embedding determined by the BST/DME. Our Solution Regions (SR) refer to a set of solutions that reduce temperature dependent skew $SKEW_T(p)$ for the point under consideration.

The algorithm takes as input the clock tree $TR$ generated by BST/DME with skew bound $B$, and temperature profile $T(X)$, and returns the modified clock tree $TR_T$, with a resulting skew $SKEW_T$.

Algorithm 1 SRB_TreeGen$(TR, T_X)$

1: $TR_T = \text{Build\_Bottom\_Up}(TR, T(X))$
2: $TR_T = \text{Embed\_Top\_Down}(TR_T)$
3: $TR_T$ = Update\_Tree\_Characteristics$(TR_T)$
4: $SKEW_T$ = Recalculate\_Skew$(TR_T, T(X))$

We start bottom-up by updating $SKEW_T(p)$ for each Steiner point by calculating the $ED_T$’s given by Equation 1 from that point to its left and right children. This skew is determined to evaluate solution feasibility.

We will now explain the two major stages of the algorithm in detail, namely, the Build\_Bottom\_Up and Embed\_Top\_Down operations.

4.4.1 Bottom-Up Phase

The procedure Build\_Bottom\_Up (Algorithm 2) takes as inputs a clock tree $TR$ and a temperature profile $T(X)$, and returns a set of solution regions, each one associated to a node $p$ of $TR$. The algorithm recursively builds a SR for a node $p$ SR based on the SR of its children $u$ and $v$ by invoking the function BuildSR, which visits the tree in a post-order fashion. If the node is a sink, the SR of the node is the embedding (i.e., the position) of the node itself. For a generic node $p$, we recur on its two children ($u$ and $v$), and use their SR’s to build the SR for the current node. This is done by exhaustively evaluating the feasibility of the move of the points $p, u,$ and $v$ within their SR; $SR(p)$ initially consists of the set of points within a distance $Rad$ from $p$ (Figure 3).

We then evaluate $ED_T$ for any two pairs of Steiner points corresponding to the $SR$ of $p, u,$ and $v$. If moving the $p$ to another position will yield a skew lesser than the $SKEW_T(p)$ for that Steiner point in the original tree, it qualifies as a feasible solution (DetermineSolutionFeasibility). A solution consists of information regarding its position and its children’s position, and its minimum/maximum delay to any sink of the subtree rooted at this node. We then prune the solution (UpdateSolutionRegion) according to the following criteria:

Algorithm 2 Build\_Bottom\_Up$(TR, T(X))$

1: $r =$ root of $TR$
2: $Rad =$ search radius
3: $SR(r) =$ BuildSR$(r, Rad)$
4: BuildSR$(p, Rad)$
5: if $p$ is a sink then
6: $SR(p) =$ $e(p)$
7: else
8: $u =$ $L(p); v =$ $R(p)$
9: $SR(u) =$ BuildSR$(u, Rad)$
10: $SR(v) =$ BuildSR$(v, Rad)$
11: foreach (point $r$ with distance $\leq Rad$ from $p$)
12: $SR(r)$ = UpdateSolutionRegion$(SR(p))$
13: endfor
14: ComputeThermalDelay$(r, srl, srr)$
15: DetermineSolutionFeasibility$(r)$
16: $SR(p) =$ UpdateSolutionRegion$(SR(p))$
17: endfor
18: endfor
19: end if
• Skew, i.e., best solution has minimum $SKEW_T(p)$.

• Wirelength, i.e., best solution has minimum wirelength $WL$ but not always minimum $SKEW_T(p)$.

Figure 3: Basic Operations of the BuildSR algorithm.

The worst-case complexity of the algorithm is cubic in the magnitude of the radius $Rad$. In practice, the pruning of the solution aims at reducing execution time by allowing only a selected number of solutions within a $SR$.

4.4.2 Top-down Embedding

In this phase the final modified position for each candidate point is determined from their respective $SR$. This phase translates works exactly as in the top-down phase of DME; the tree is visited in pre-order, and for a generic node determine its embedding position based on the position of its parent. More precisely, given the embedding $e(p)$ of a node, we determine the location of its children $e(u)$ and $e(v)$ by picking the point in $SR(u)$ ($SR(v)$) that minimizes thermal distance between that point and $e(p)$. While computing the embedding, we update other critical characteristics of the tree which include wirelength information and capacitance for each point, which we use to recalculate the skew under the given thermal profile.

4.5 Critical Path Based Algorithm (CPB)

We observe that the definition of skew as given by Equation (4.2) is dependent on the sinks which have very high or very low delays from the source of the clock tree. In this section, we use this idea and present an algorithm that modifies only these extreme paths with minimal impact on the resulting clock tree. There are two practical cases where such an algorithm is advantageous: (i) If the thermal map shows a specific small region which has high thermal density, only the sinks which lie in that region should ideally be modified; (ii) For big benchmarks, the computation time for Algorithm 1 is comparable to the time for clock tree generation itself. Hence, an alternative faster algorithm is required.

4.5.1 Preliminaries

• We define Delay Till Root as the delay, associated with a sink, from the root of the clock tree to the sink. Under a uniform temperature profile this delay is denoted by $DTR$, and $DTR_T$ if a non-uniform temperature profile is given.

• We define Skew Bound Window (SBW) as a time window of width $B$ which can be placed over a sorted list of $DTR$s.

Placing this window at a sink with $DTR$ value of $D_i$ asserts that all the all sinks whose $DTR > D_i - B$ and $DTR < D_i$ have non-skew-violating delays with respect to this position of SBW. See Figure 4 for illustration.

• For a particular position of SBW, all sinks with $DTR > D_i$ belong to the Violating on High (VOL) set. Similarly the sink with $DTR < D_i - B$ belong to the Violating on Low (VOH) set. A position of SBW is optimum when $|VOH| + |VOL|$ is minimized.

• For each Steiner node, High Violation Below (HVBl) and Low Violation Below (LVB) fields denote the number of High and Low violations in the fanin cone of that Steiner node respectively.

Figure 4: $VOH$ and $VOL$ sinks for a particular position of SBW at Sink with $DTR$ $D_i$

4.5.2 Optimize Critical Path

In this section, we describe our second algorithm. The pseudocode of the procedure, Optimize_Critical_Path, is detailed in Algorithm 3.

Algorithm 3 Optimize_Critical_Path($TR$)

1: $S$$\leftarrow$ Sinks of $TR$
2: Compute and sort $DTR_T$ for all $S$
3: Compute Optimal SBW position and $VOH$ and $VOL$ sets
4: Apply Minimize_Cluster($TR$, $VOH$, $VOL$)
5: Recalculate $DTR_T$. Sort. Find $VOH$ and $VOL$.
6: Propagate Violations.
7: for all sink n in $VOH$ and $VOL$ do
8: FixThisNode(n, true)
9: end for

This procedure receives as input the clock tree topology $TR$. The $DTR_T$ are computed and sorted in decreasing order followed by calculation of the optimal position of SBW. For this position, the set $VOH$ and $VOL$ are calculated. Then, to reduce the delay of sinks clustered together, the procedure Minimize_Cluster described in Algorithm 4 is performed. With the new resultant tree, we recompute the $DTR_T$ for each sink and sort them in decreasing order. The optimal position of the SBW is re-generated and the set of $VOH$ and $VOL$ are updated. For each node which is in the set $VOH$, we call the procedure FixThisNode with the flag as true. With flag as true, Algorithm 5 optimizes the delay of the current node by recursively applying optimization to its parent until it reaches the root of the clock tree.

4.5.3 Clustered VOH (VOL) Minimization

As mentioned above, we apply the procedure Minimize_Cluster described in Algorithm 4 to gain reduction of delay of many VOH (VOL) sinks clustered together with very minimal changes in the clock network. For every VOH (VOL) sink n, we climb up the clock tree towards the root node and increment the HVB (LVB) fields
for all Steiner nodes in the path from sink \( s \) to root of clock tree. This pair of numbers, HVB and LVB, form the violation signature of a Steiner node. A higher number for a signature means there are violations in the fan-in cone of the node under observation which are violating the skew bound. This is followed by a pre-order traversal (\texttt{FixNode}) from the root of the clock tree looking for a Steiner node which has a non-conflicting violation signature. A post-order visit guarantees a change is made on higher levels of the clock tree structure which are common to cluster of \( VOH \) (\( VOL \)) below this Steiner node. A non-conflicting signature means that for this Steiner node, only one of HVB and LVB are non-zero. To this node, we apply the \texttt{FixNode} procedure (Algorithm 5) with flag false. All Steiner nodes in the fanin cone of this particular Steiner point are marked as optimized.

### 4.5.4 Node Delay Reduction

The procedure \texttt{FixNode}, described in Algorithm 5, is used for reducing the delay of a node \( n \) from the root by repositioning \( n \)'s parent. We predict the quadrant and a radius (defined as distance of \( P' \) from \( P \) in Figure 5) to move in this quadrant. Let this region be called \( S \). We follow this up by estimation of elmore delay under thermal variation for all possible locations of \( P' \) lying in \( S \). The location which minimizes the delay from \( U \) to \( P \) is picked up. In case this algorithm is called with flag true, the function recurses calling itself on the parent \( P \) of the current node \( n \).

#### Quadrant Estimation:

For each parent \( P \) in Figure 5, the region around it can be broken up into four Quadrants as shown in Figure 5. Let us assume the position of \( U \) and \( V \) be as shown in Figure 5. Also, assume that the node \( U \) lies on one of the \( VOL \) sink’s path. In this case, we target \textit{increasing} delay of segment \( UP \) and \textit{reducing} delay of segment \( VP \). The first condition warrants choosing Quadrant I, II and IV whereas the second condition requires choosing Quadrant IV for possible movement of Steiner node \( P \). We take intersection of these requirements which means the Steiner node \( P \) should move in Quadrant IV only as shown by the shaded region.

#### Radius Estimation:

Once a quadrant is identified, the radius which Steiner node \( P \) would move is estimated using Equation (1) by estimating difference in delay by increasing/decreasing a segment of length \( L \) by an amount \( x \). We move our Steiner node a few points less and more than the estimated radius. By using above two estimates, it is possible to prune out most of the unfeasible possible locations of \( P' \) from consideration which allows exhaustive searching in the region of interest shown by the concentric arcs in Figure 5.

---

**Algorithm 5 FixThisNode**

1: \( n \) = Node to fix 
2: if \( n \) is root of TR, return 
3: \( U \leftarrow n, V \leftarrow \text{Sibling of } n, P \leftarrow \text{Parent of } n \) 
4: \( PP \leftarrow \text{Parent of } P \) 
5: \( S \leftarrow \text{FindTargetQuadrant}(P) \) 
6: \( R \leftarrow \text{InitialRadiusDoes}(P) \) 
7: for all \( P' \in \text{Quadrant } S \) at radius \( R \) do 
8: Pick best \( P' \) such that \( DB(U - P') + DB(P' - PP) \) is minimized 
9: end for 
10: if flag then 
11: \texttt{FixThisNode}(P, \text{flag}); 
12: end if 

---

**Figure 5: Selection of quadrant for Steiner node \( P \)**

\textbf{Feasible locations of \( P' \):} We select the location \((x, y)\) in the pruned search space for \( P' \) as is shown in Figure 5 which minimizes the delay from \( PP \) to \( U \) through \( P' \). Let \( DB(U - P') \), be the Delay Between segment \( U \) and \( P' \) and \( DD(U) \) be the Delay as seen Downwards from the Steiner node \( U \). then we choose a location which minimizes \( DB(U - P') + DB(P' - PP) \) subject to

\[
DB(U - P') + DD(U) \geq DB(V - P') + DD(V)
\]

This condition guarantees that all the possible location of Steiner node \( P' \), the Steiner node \( V \) does not become more critical than \( U \) as it would lead to a situation at least as difficult as what we are trying to solve.

### 5. Experimental Results

We have implemented in C++ the two above described algorithm around the framework provided by the Bounded Skew Tree/DME code from [17]. We have benchmarked the algorithms for the r1-r5 and p1-p2 sink location files [4], standard benchmarks for evaluating clock-tree routing algorithms. For all our experiments, we measured the wirelength, skew for both uniform and non-uniform thermal profile, and minimum and the maximum \( DTR \) before and after clock tree optimization. For all the benchmarks, the initial value of skew bound given to BST was 100ps. The actual bound was calculated by applying Elmore delay model under thermal effects modeled in [3] and using integration by summation.
Table 2 summarizes the results. The table is split in three parts, each one corresponding to one of the three thermal profiles. For all the experiments, an initial skew bound of 100ps was used. Column *Orig* reports the skew value under the given thermal profile before optimization: This is the value to which the initial skew bound of 100ps rises due to the non-uniform temperature profile. Columns *SRB* and *CPB* reports the skew after optimization with our algorithms, as well as the relative savings, computed as $(\text{Opt}[\text{ps}] - \text{Orig}[\text{ps}]) / \text{Orig}[\text{ps}]$ where *Opt* is the skew value after using either *SRB* or *CPB*.

Results show that the SRB algorithm yields about 56% savings with over the skew value for the clock tree generated by BST in the presence of thermal profiles, while the CPB algorithm yields about 51% savings. This difference in optimization potential comes at the price of an average 1:4 ratio in execution time, in favor of the CPB algorithm.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th><em>Orig (ps)</em></th>
<th><em>SRB (ps)</em></th>
<th>%</th>
<th><em>CPB (ps)</em></th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>121</td>
<td>107</td>
<td>66.6</td>
<td>109</td>
<td>57.1</td>
</tr>
<tr>
<td>P2</td>
<td>171</td>
<td>138</td>
<td>46.5</td>
<td>141</td>
<td>42.2</td>
</tr>
<tr>
<td>R1</td>
<td>181</td>
<td>149</td>
<td>39.5</td>
<td>154</td>
<td>33.3</td>
</tr>
<tr>
<td>R2</td>
<td>365</td>
<td>184</td>
<td>68.3</td>
<td>194</td>
<td>64.5</td>
</tr>
<tr>
<td>R3</td>
<td>271</td>
<td>149</td>
<td>71.3</td>
<td>154</td>
<td>68.4</td>
</tr>
<tr>
<td>R4</td>
<td>663</td>
<td>318</td>
<td>61.3</td>
<td>340</td>
<td>57.3</td>
</tr>
<tr>
<td>R5</td>
<td>1268</td>
<td>723</td>
<td>46.7</td>
<td>784</td>
<td>41.4</td>
</tr>
</tbody>
</table>

Average: 57.2 52.1

<table>
<thead>
<tr>
<th>Benchmark</th>
<th><em>Orig (ps)</em></th>
<th><em>SRB (ps)</em></th>
<th>%</th>
<th><em>CPB (ps)</em></th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>168</td>
<td>121</td>
<td>64.9</td>
<td>127</td>
<td>60.3</td>
</tr>
<tr>
<td>P2</td>
<td>386</td>
<td>203</td>
<td>64.0</td>
<td>214</td>
<td>60.1</td>
</tr>
<tr>
<td>R1</td>
<td>246</td>
<td>165</td>
<td>65.5</td>
<td>174</td>
<td>49.3</td>
</tr>
<tr>
<td>R2</td>
<td>324</td>
<td>181</td>
<td>63.8</td>
<td>189</td>
<td>60.2</td>
</tr>
<tr>
<td>R3</td>
<td>891</td>
<td>463</td>
<td>54.1</td>
<td>487</td>
<td>51.0</td>
</tr>
<tr>
<td>R4</td>
<td>1494</td>
<td>794</td>
<td>50.2</td>
<td>823</td>
<td>48.1</td>
</tr>
<tr>
<td>R5</td>
<td>2785</td>
<td>1578</td>
<td>44.9</td>
<td>1622</td>
<td>43.3</td>
</tr>
</tbody>
</table>

Average: 55.4 56.2

6. Conclusions

Temperature variations across the die affect power consumption, performance, reliability, and manufacturability of ultra-deep submicron chips. The clock network, in particular, is very susceptible to such temperature gradients because it spans the entire area of the die. Clearly, any deviation of the clock signals from the expected behavior directly translates in timing errors; this makes thermal analysis essential to guarantee correct operations of a design. In this work, we have presented a technique for the temperature-aware design of the clock distribution network, which leverages conventional, non-temperature-based algorithms to restructure a given the clock tree so as to minimize the thermally-induced skew with marginal increase of the wirelength. Two algorithms that span different points in the speed/efficiency tradeoff have been presented, that allowed to reduce the worst case skew by 56% on average.

7. Acknowledgements

The authors thank Prof. David Z. Pan at University of Texas, Austin for providing us with a pre-print copy of [15].

8. REFERENCES

[17] A. B. Kahng et al., vlsicad.ucsd.edu/GRSC/bookshelf/Slots/BST.