Practical Techniques for Minimizing Skew and Its Variation in Buffered Clock Networks

ABSTRACT

Clock skew is becoming increasingly difficult to control due to many variational effects. Link based non-tree clock distribution is a cost-effective technique for reducing clock skew variations. However, previous works based on this technique were limited to unbuffered clock networks and neglected spatial correlations in the variation models. In this work, we overcome these shortcomings and make the link based non-tree approach feasible for realistic designs. The short circuit risk and multi-driver delay issues in buffered non-tree clock networks are investigated. A new clock skew tuning technique under accurate gate/wire delay models is proposed to enhance the effect of link insertion. Our approach is validated with SPICE based Monte Carlo simulations, considering spatial correlations among variations. The experimental results show that our approach can reduce the maximal skew by 47%, improve the skew yield from 15% to 73% on average with a decrease on the total wire and buffer capacitance.

1. INTRODUCTION

With the advent of nanometer VLSI technology, clock distribution networks are often haunted by increasingly significant variational effects such as process variations [1] and power supply noise [2]. Non-tree clock distribution network [3–7] has been recognized as a promising approach for reducing clock skew variations, since multiple signal paths can compensate each other's variations. Among the non-tree techniques, the clock mesh [3, 5] is perhaps the most effective and most well-known one, but it usually entails a large wire/power overhead and is therefore affordable only in high performance products such as microprocessors. In contrast, the link based non-tree method [6–8] is much more cost effective and will be the focus of study in this paper.

In [7], an analysis is performed to show that a link resistor inserted between two nodes in an RC network always reduces the skew or the skew variation between them. The impact of the link on the other node pairs is also analyzed [7] and found to be much less tractable. Based on these analyses, the work of [7] proposes algorithms of constructing a non-tree clock network by inserting cross links in an existing clock tree. Later, some algorithm improvements were suggested in [8]. These algorithms follow an important rule that links are inserted only between nodes with equal nominal delays.

The link based non-tree method has several advantages among which the most important one is its cost-effectiveness. Compared to clock trees, cross link insertion can reduce the maximum skew variation by over 30% with less than 2% increase on wirelength [7]. A clock mesh can reduce the maximum skew variation by 90% but with over 60% increase on wirelength [7]. Therefore, a link based non-tree approach is an appealing choice for many ASIC designs which have relatively stringent cost/power constraints. Since such non-trees are built upon existing clock trees, this method can be easily incorporated with traditional tree based design methodology. Moreover, it can be easily extended to achieve useful non-zero clock skews. The relatively difficult non-tree delay computation is circumvented in the methods of [7]. The link insertion in [6] is a special case which handles only H-trees.

Despite the advantages, previous works on link based non-tree clock network [7,8] have a few shortcomings which hamper their applicability. The major weakness is that these works [7, 8] are limited to unbuffered clock networks. In reality, most of clock networks are buffered due to the requirements on signal slew rate and minimal path delay. More importantly, buffer variations [1, 2] are usually the major contributors to clock skew variations. The effect of link insertion in buffered networks is more difficult to control than that in unbuffered cases due to the nonlinear behavior of buffer delays and the appearance of multi-driver nets. In addition to this methodological weakness, the experiment setup of [7, 8] neglected spatial correlations [9, 10] in the variation models. It has been recognized that many variations such as intra-die process variations [10] and power supply fluctuations are spatially correlated. Moreover, the testcases in [7, 8] do not contain sequential adjacency information. A pair of clock sinks, which are usually flip-flops or latches, are sequentially adjacent [11] if there is a path between them consisting of only combinational logic. The clock skew constraints are essential only between sequentially adjacent clock sinks [12]. Therefore, evaluating skew variations between all pairs of clock sinks as in [7, 8] does not necessarily convey the most relevant results.

The major goal of this work is to overcome these shortcomings and make the link based non-tree technique applicable in practice. The main contributions of this work are summarized as follows.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$5.00.

- Link insertion in a buffered network may result in multiple drivers for a subnet. We suggest a design criterion for avoiding short circuit risk in a multi-driver net. Some analysis results obtained in [7] for single driver nets are extended for multi-driver nets.

- We propose a new skew tuning technique for buffered clock tree under accurate gate and wire delay models. The effect of link insertion depends on a well-designed buffered clock tree which is not easy to obtain under gate and wire delay models. The proposed technique can decrease nominal clock skew considerably and therefore enhances the effectiveness of link insertion.

- A complete methodology of link based buffered clock network under accurate gate and wire delay models is proposed. This methodology utilizes the buffered clock tree construction techniques which are friendly to link insertion.

- The proposed method is validated with SPICE based Monte Carlo simulations considering spatially correlated power supply variations, buffer and wire process variations. Our experiments are performed on testcases with sequential adjacency information. The maximal skew, standard deviation of skew variations and skew yield [13], which is the probability of satisfying a certain skew bound, are evaluated in the Monte Carlo simulations.

The rest of this paper is organized as follows. The major conclusions of previous work [7] are summarized in Section 2. Several issues involving multi-driver nets are discussed in Section 3. A new clock skew tuning technique is introduced in Section 4. The proposed link based buffered clock network methodology and algorithms are described in Section 5. The experimental results are provided in Section 6. Finally, we conclude in Section 7.

2. LINK INSERTION REVIEW

For completeness, we summarize a few important conclusions from previous link insertion research [7]. The basic idea behind the link based non-tree clock network method is to obtain a non-tree by inserting cross links between nodes in an existing clock tree. A link can be modeled as a link resistor with a pair of link capacitors at the two ends. Adding only link capacitances to a clock tree may change the skew but does not change the tree topology. The original skew can be restored by tuning the tree as in conventional clock tree construction methods.

If a link resistor is inserted between a pair of nodes with equal nominal delay (or zero nominal skew), there is no change on nominal delay at any node in the clock network. If there is skew variation between the two end nodes of the link resistor, the magnitude of the variation is *always* scaled down by the link resistance. The effect of the scaling is strong when the link resistance is small or the nearest common ancestor node of the two end nodes is close to the root. If one end of the link is in subtree T_l and the other end is in a disjoint subtree T_r, the link resistance can reduce skew variation between any pair of nodes of T_l and T_r. However, the link resistance may worsen skew variability between nodes in some other circumstances (see [7]).

The major guidelines for link insertions include:

- Links are always inserted between nodes with zero nominal skew.

- Links are preferentially inserted between node pairs which are close to each other and their nearest common ancestor node is close to the root in the abstract topology.

- Links need to be distributed evenly in the clock network so that their skew worsening effects can cancel each other.

3. MULTI-DRIVER NETS

If cross links are inserted in a buffered clock network, it is likely that a sub-net is driven by multiple buffers or drivers. This fact causes two issues which do not exist in link insertion for unbuffered clock networks. One is the risk of short circuit between the outputs of different buffers. The other is whether or not the analysis on delay and skew in [7] still valid in the multi-driver nets.

3.1 Short Circuit Avoidance

If the signal arrival times at the inputs of two buffers driving the same sub-net are significantly different, there is a risk of short circuit power consumption. This arrival time difference can be caused by either nominal delay difference or delay variations. Consider the example in Figure 1 where the outputs of the two buffers are initially low and then switch to high with time difference of Δ. There is an time interval Δ during which the output of upper buffer is high while the output of the lower buffer is low. Therefore, there could be a short circuit current flowing from the power supply to the ground through the upper buffer and then the lower buffer as indicated by the dashed line in Figure 1.

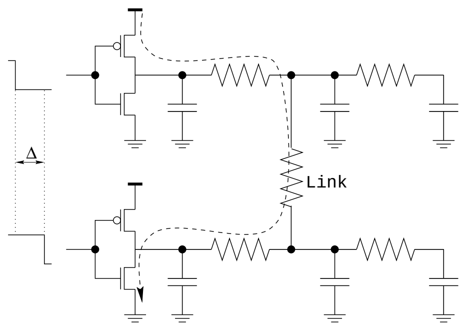

Figure 1: If there is significant difference Δ between signal arrival time to the two drivers, there is risk of short circuit indicated by the dashed line.

However, there is a delay for a signal (or a current) to propagate from one buffer to the other. If this delay is greater than Δ, then there is not enough time to establish the short circuit current. In other words, the output of the lower buffer may switch to high before the signal of the upper buffer propagates to it. Based on this observation, a design criterion for avoiding short circuit current between two buffers is derived as follows.

Denote the two buffers as B_i and B_j. Let the upper bound of the difference between signal arrival time to B_i and B_j be $\Delta_{ij,max}$ considering variations. The lower bound $\tau_{i\leadsto j}$

of signal propagation delay from B_i to B_j can be obtained through the method of [14].

$$\tau_{i\rightsquigarrow j} = \frac{\sum_{(u,v)\in path(B_i\rightsquigarrow B_j)} R_{uv}^2 C_v}{\sum_{(u,v)\in path(B_i\rightsquigarrow B_j)} R_{uv}} \qquad (1)$$

where (u, v) indicates two end nodes of an edge, R_{uv} is the edge resistance and C_v is the total capacitance downstream of node v. The low bound $\tau_{j\rightsquigarrow i}$ of signal propagation delay from B_j to B_i can be obtained similarly. Then, the criterion for avoiding short circuit between B_i and B_j is:

$$\min(\tau_{i\rightsquigarrow j}, \tau_{j\rightsquigarrow i}) > \alpha\Delta_{ij,max} \qquad (2)$$

where $\alpha > 1$ is a constant used for added safety margin.

3.2 Multi-driver Delay Analysis

In [7], it was shown that a link resistor inserted between two nodes in an RC network always reduces the skew or the skew variation between them. However, this conclusion is for the single driver case. We will show that a multi-driver net can be converted to an equivalent single driver net and therefore the conclusion in [7] still holds for multi-driver nets.

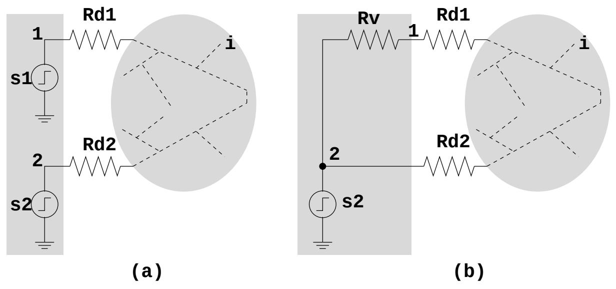

Figure 2: The dual driver net in (a) can be converted to the single driver net in (b) when signal departure time t_1 at node 1 is no less than the signal departure time t_2 at node 2.

If there are two drivers for a net as in Figure 2(a), the term of signal delay is not well defined since the signal departure time from the two drivers may be different. More precisely, we need to find the signal arrival time t_i to a node i in the multi-driver net given the signal departure time t_1 and t_2 from from node 1 and node 2 in Figure 2(a). Without loss of generality, it can be assumed that $t_1 \geq t_2$, i.e., $t_1 = t_2 + \Delta$ and $\Delta \geq 0$. Now let us consider inserting a virtual resistance R_v between the signal source s_1 and node 1. If the signal delay across the virtual resistance equals Δ and the signal departure time from s_1 is t_2, the signal departure time t_1 at node 1 is not changed after this virtual resistance insertion. If the signal departure times at both s_1 and s_2 are t_2 with this insertion, we can merge s_1 with s_2 into a single source as shown in Figure 2(b). Please note that this merging does not affect the gate driving capability as the driving capability is decided only by R_{d1} and R_{d2} in this model. If there are more than two drivers, we can do the same to obtain an equivalent single driver net. Thus, the conclusions in [7] still hold for multi-driver nets due to this transformation.

In the above transformation, we also need to find the value of the virtual resistance R_v such that the delay across it is equal to Δ. Since the net in Figure 2(b) has a single driver, the signal delay t_1 at node 1 is well defined by letting the signal departure time $t_2 = 0$. Evidently, t_1 is a function $t_1(R_v)$ depending on R_v and the value of R_v can be obtained by solving the equation $t_1(R_v) = \Delta$. Even though the Elmore delay at nodes in a non-tree RC network as in Figure 2(b) for any fixed value of R_v can be computed by the tree partitioning method [15], obtaining an analytical expression for function $t_1(R_v)$ is not trivial. Fortunately, the non-tree here is a special case where the links are inserted only between nodes with the same nominal delay [7]. It has been shown in [7] that inserting such link resistors does not change delay to any node. If the Elmore delay to a node i is initially t_i, then its delay becomes $\hat{t}_i$ after a link resistance R_l is inserted between two arbitrary nodes u and w. According to [15], the Elmore delay $\hat{t}_i$ can be computed as:

$$\hat{t}_i = t_i - \frac{t_u - t_w}{R_l + r_u - r_w} r_i \qquad (3)$$

where t_u and t_w are the Elmore delay at node u and w before inserting the link resistance, and the value of r_i is equal to the Elmore delay at i when node capacitance $C_u = 1$, $C_w = -1$ and the other node capacitances are zero. Obviously, when the link resistance is inserted between two nodes u and w with $t_u = t_w$, there is no change on delay at any node. Therefore, $t_1(R_v)$ can be obtained by ripping up all of the link resistance from the non-tree network and finding the Elmore delay in the resulting tree. In other words, $R_v = \frac{\Delta}{C_1}$ where C_1 is the total downstream capacitance at node 1 for the tree.

4. ACCURATE AND LOCALIZED SKEW TUNING

In this section, we will introduce a new skew tuning technique which can be applied to general buffered clock network synthesis and is especially friendly to link based clock networks.

According to [7], link resistors need to be inserted between a pair of nodes with zero nominal skew so that nominal delay at each node is not affected. In other words, we need to tune the skew between a selected pair of nodes to be zero (or minimal) after the link capacitance is added. The skew tuning in buffered clock networks has to be performed under accurate gate model such as SPICE or lookup table model, since the RC switch gate model employed in [7, 8] is inadequate on handling the nonlinear behavior of buffer delays.

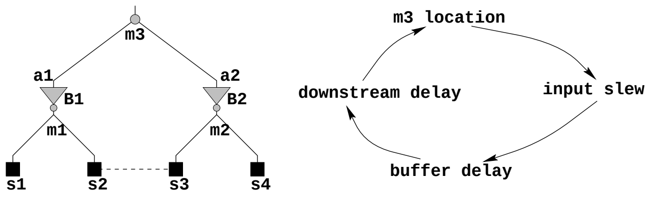

Figure 3: Tuning the location of merging node m3 in a buffered clock tree falls into a cyclic dependency.

However, skew tuning in a buffered clock network under accurate gate and wire delay model is much more difficult than Elmore delay based tuning [16]. We will illustrate this difficulty through an example of zero skew clock tree construction in Figure 3. If zero skew has already been obtained for the buffered subtrees rooted at a1 and a2, we attempt to tune the location of the merging node m3 such that the delay from m3 to each sink (s1, s2, s3 or s4) is the same. Let *downstream delay* at a node k, or the delay from k to sinks, be denoted as d_k. The location of m3 is decided based on downstream delay $d_{a1} = delay(B1) + d_{m1}$ and $d_{a2} = delay(B2) + d_{m2}$. The buffer delays $delay(B1)$ and $delay(B2)$ depend on their input slew rates. However, the input slew rates depend on the location of merging node m3. We thus get into a *vicious cycle* that makes it very difficult to accurately find a merging node location that gives zero skew. This cycle is depicted at the right part of Figure 3.

The weakest link in this cycle is the dependence of merging node location on the downstream delay d_{a1} and d_{a2}. We propose a tunable clock buffer technique to break the weakest link as well as the vicious cycle. If buffer delay $delay(B1)$ and $delay(B2)$ can be tuned without affecting other delay or slew in the buffered tree, we can decide the location of m3 regardless downstream delay d_{a1} and d_{a2} and then obtain zero skew at sinks by tuning the buffer delays.

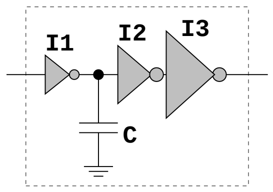

Figure 4: Tunable clock buffer.

Figure 4 shows an example of a tunable buffer containing three cascaded inverters even though different number of inverters can be employed. There is a tunable dummy capacitor C between inverter I1 and inverter I2. For a given input slew and a given output load, the delay of the buffer can be tuned by sizing the dummy capacitor. Since the dummy capacitor is sandwiched between inverters in the buffer, changing its size does not affect any other delay or slew in the buffered tree but the buffer delay itself. This is quite different from previous works [17,18] where dummy capacitors are directly exposed to a subtree and a change on a capacitor may affect everywhere in the subtree. Since this tuning technique is performed locally, it runs faster than global tuning methods such as [19].

For the example in Figure 3, we can simply choose a location for m3 such that the delay from m3 to a1 is the same as the delay from m3 to a2. Then, we can make $d_{a1} = d_{a2}$ by sizing the dummy capacitors in tunable buffer B1 and B2 so that clock skew among the sinks is zero. Since the delay change with respect to the size of a dummy capacitor is monotone, the size of a dummy capacitor can be decided through a binary search guided by circuit simulations.

5. LINK BASED BUFFERED CLOCK NETWORK CONSTRUCTION

5.1 Buffered Clock Tree Construction

The link based non-tree clock network construction starts with an initial buffered clock tree. There are many previous works on clock tree routing [20,21] and buffered clock tree construction [22–25] and various techniques are included in these different works. We integrate the best of them, those friendly to link insertion and our own tuning technique (Section 4) into a buffered clock tree construction method which can facilitate better link insertion effect.

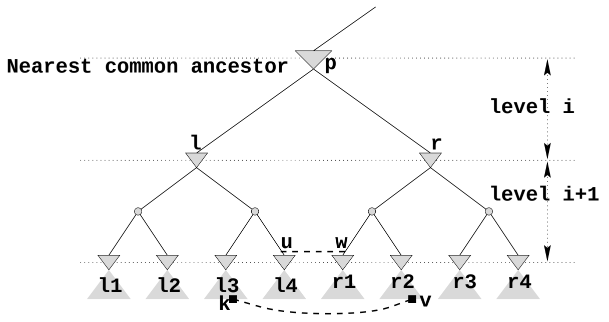

Figure 5: Link insertion in buffered clock tree. Dashed lines indicate links.

Similar as previous works [22–24], we try to build a balanced tree with roughly equal number of buffers along every source-sink path. This balanced buffered clock tree scheme is illustrated in Figure 5. Such balanced structure itself has certain tolerance to inter-die variations [22,23]. Further, we will show later that it is friendly to link insertion.

The buffered clock tree is constructed through a bottom-up merging procedure like many traditional clock routings [20, 21,25]. The merging order is based on the nearest neighbor method [21] which selects a pair of subtrees closest to each other for merging. In order to maintain the tree balance, we impose an extra restriction that only subtrees with fewer levels are merged first. The location of each merging node is decided by the DME (Deferred Merge Embedding) technique [20] based on the Elmore delay model. Buffers are inserted at every internal node at the same level as in Figure 5 such that the maximum load of each buffer/driver is limited. This is an indirect way to ensure proper signal slew rate [25].

In addition to the structural balance, we perform **delay balancing** [22] for subtrees at each level. After the buffer insertions, the entire tree can be partitioned into subtrees at different levels indicated by the dotted lines in Figure 5. In delay balancing, we make the delay of subtrees at the same level identical. For example, $delay(l \rightsquigarrow u) = delay(r \rightsquigarrow w)$ for Figure 5. The delay balancing can be achieved by using the tunable buffer introduced in Section 4 and sizing the dummy capacitors.

There are two main reasons for the delay balancing. First, delay balancing results in almost equal signal arrival time at each buffer of the same level. Hence, the risk of short circuit is greatly reduced if link insertion is restricted among subtrees of the same level according to the discussions in Section 3.1. The other reason is for the convenience of multilevel link insertion. Every subtree rooted at a buffer/driver becomes a zero skew subtree after delay balancing. Then, cross links can be easily inserted between subtrees at higher levels like the link between u and w in Figure 5.

After the buffered clock tree is constructed, a SPICE simulation is performed to obtain a precise estimation on clock skew. Usually, the skew within a subtree rooted at a buffer/driver is negligible. Since there is no buffers within such a subtree, the Elmore model provides a fairly good fidelity which is verified by SPICE simulations in [8]. However, there could be a significant delay difference between different subtrees at the same level. Thus, we perform a post-processing of delay balancing through tuning the clock buffers based on the SPICE base skew information. Compared to previous skew tuning work [19] using SPICE model, our method is much easier.

5.2 Link Insertion

The algorithm for link insertion in buffered clock networks has some significant differences from the unbuffered case [7, 8], even though they share the same top-level framework below.

1. Select the node pairs for link insertion.
2. Add link capacitance to the selected nodes and perform skew tuning to restore the original skew. The skew tuning includes two steps. First, the locations of merging nodes in each subtree rooted at a buffer/driver are tuned to restore zero skew for the subtree. Next, SPICE simulation is performed to obtain a precise inter-subtree skew estimation. And the inter-subtree skew is minimized by sizing the dummy capacitors in the tunable clock buffers. This step is the same as the post-processing in the initial buffered clock tree construction.
3. Insert link resistance into the selected node pairs. Since we always select node pairs with zero nominal skew and restore such zero skew in skew tuning of previous step, the link resistances do not affect nominal skew.

```
Procedure: $NodePairsBetweenTrees(T_l, T_r, m)$
Input: Two subtree $T_l$ and $T_r$,
       size indicator $m$ for each sink/buffer level
Output: A set $P$ of node pairs
1. $P \leftarrow \emptyset$
2. For each sink/buffer level deeper than $l$ and $r$
3.    Decompose $T_l$ to sub-subtrees $S_l = \{l_1, l_2 ... l_m\}$
4.    Decompose $T_r$ to sub-subtrees $S_r = \{r_1, r_2 ... r_m\}$
5.    Construct bipartite graph $G_{l,r}$ between $S_l$ and $S_r$
6.    $G_p \leftarrow$ MST of $G_{l,r}$
7.    For each edge $(l_i, r_j)$ in $G_p$
8.       If link between $l_i$ and $r_j$ has short circuit risk or
9.       no sequentially adjacent sinks between $l_i$ and $r_j$
10.          Remove $(l_i, r_j)$ from $G_p$
11.      Else if $degree(l_i) > 1$ and $degree(r_j) > 1$
12.      and $weight(l_i, r_j) >$ threshold
13.          Remove $(l_i, r_j)$ from $G_p$
14.   $P \leftarrow P \cup$ edges in $G_p$
15. Return $P$
```

Figure 6: Algorithm of selecting node pairs between two subtrees.

The most important step is node pair selection. Similar as [7,8], the selection proceeds from the root node toward the leaf nodes recursively. In Figure 5, when an internal node p is visited during this tree traversal, node pairs between its left subtree T_l rooted at node l and right subtree T_r rooted at r are selected. Every node pair between T_l and T_r shares the same nearest common ancestor node p. Then, the same procedure is performed for nodes l, r and their child nodes. The algorithm of selection between T_l and T_r is shown in Figure 6. The input of this algorithm contains a user-specified parameter m which roughly indicates the number of pairs to be selected. A larger m implies that more node pairs are selected.

Our selection algorithm has a significant difference from previous works [7,8] which restrict link insertion among only sink nodes. We also consider link insertion between input nodes of buffers at the same level. For example, a link can be inserted between node u and w in Figure 5. Because the delay balancing introduced in Section 5.1 is performed, the nominal skew between input nodes of buffers at the same level should be close to zero. Link insertion between these input nodes can further reduce their skew variation as well as the risk of short circuit. Moreover, such multilevel link insertion can enhance the effect of skew variability reduction when handling with severe buffer variations. Therefore, node pair selection is performed at each sink/buffer level deeper than l and r as indicated by the loop starting from line 2 of Figure 6.

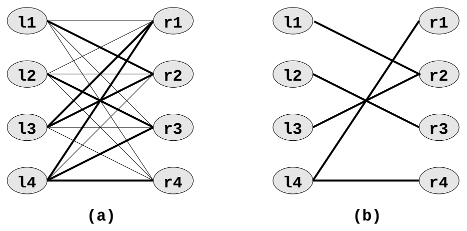

Figure 7: Find MST (thickened edges) on the bipartite graph in (a). After performing iterative deletion, the selected node pairs are indicated by the edges in (b).

The node pair selection procedure at each sink/buffer level is derived from the MST (Minimum Spanning Tree) based algorithm in [8]. Both the left subtree T_l and the right subtree T_r are decomposed into m sub-subtrees. In Figure 5, each subtree is decomposed into 4 sub-subtrees. Then, a bipartite graph is generated with each vertex corresponding to a sub-subtree as shown in Figure 7(a). The weight of an edge is equal to the minimum sink/buffer distance between the two sub-subtrees it is incident to. For example, if the nearest sink pair between sub-subtree l_3 and sub-subtree r_2 is k and v in Figure 5, the weight of edge (l_3, r_2) is equal to the distance between sink k and sink v. Next, an MST (Minimum Spanning Tree) on this bipartite graph is obtained as indicated by the thickened edges in Figure 7(a).

The edges in the MST correspond to candidate node pairs for the selection. These candidate edges are further pruned through a rule based iterative deletion as in line 7-13 of Figure 6. In addition to the algorithm in [8], our algorithm considers two more criteria for edge deletion. One is to ensure that there is no short circuit risk when a link is inserted for a node pair. If the inequality (2) is violated, the corresponding edge is removed. The other one is for improving the efficiency of link insertion by considering sequential adjacency information. As the skew constraints are essential only between sequential adjacent sinks [11], an edge between

Case	#Sinks	Non-balancing, WO-link						Balancing, WO-link				
		Delay	Skew	WL	#Buf	Total cap	CPU	Delay	Skew	WL	Total cap	CPU
s9234	135	468	97	41498	8	7.24	0.3	367	0.5	37043	6.44	31
s5378	164	612	71	46218	20	8.09	1.0	379	2.9	42522	3.00	51
s13207	500	663	60	133650	80	23.15	2.0	662	11.7	129203	23.01	203
Ave	266	581	76	73789	36	12.8	1.1	469	5.0	69589	10.8	95

Table 1: Nominal results for buffered clock trees. The maximal source-sink delay and the maximal skew are in ps. The wirelength (WL) results are in μm. The total capacitance is in pF. CPU time is in seconds.

sub-subtree l_i and sub-subtree r_j is removed if there is no sequentially adjacent sink pair between them. The rule considered in line 12 of Figure 6 is similar to that in [7,8]. This rule tries to remove edges which may cause physically long links, as long links increase wirelength and disturbance to the original clock tree. Before enforcing this rule, the degree of each end vertex of the edge is checked. If one of the end vertices has degree less than 2, the edge deletion may result in no link for the corresponding sub-subtree and is therefore skipped.

Since the bipartite graph has m vertices, the MST has $2m - 1$ edges. Hence, the number of node pairs selected for T_l and T_r at a sink/buffer level is on the order of m.

6. EXPERIMENT

6.1 Experiment Setup

The ISCAS89 benchmark suite is employed for the experiment as it includes relatively complete circuits with both combinational and sequential elements so that the sequential adjacency information for clock sinks (flip-flops) are available. In contrast, the well known $r1 - r5$ clock net benchmarks [16] do not have combinational logic or the sequential adjacency information. Since the clock skew constraints are essential only for sequentially adjacent sink pairs [12], the ISCAS89 benchmark circuits are more realistic than the $r1 - r5$ benchmark circuits.

First, synthesis is performed on the ISCAS89 benchmark circuits by using SIS [26] to obtain a mapped netlist. Then, an academic placement tool *mPL* [27] is employed to get circuit placement. Assuming a 180nm technology, wire capacitance values are obtained by using the SPACE 3D extraction tool [28]. The other wire parameters such as ILD (inter-layer dielectric) dimensions and sheet resistances are taken from [29]. Gate/buffer models and timing simulations are based on HSPICE with 180nm technology parameters from [30].

Our techniques are implemented in C++ and run on a Sun Solaris Ultra Sparc machine with 2GB RAM. Clock skew variations are evaluated through Monte Carlo simulations. Variations on buffer channel length, power supply level, wire width and sink load capacitance are considered. These variations are assumed to follow Gaussian distribution with standard deviations equal to 5% of their nominal values. Spatial correlations among the variations are handled by the PCA (Principle Component Analysis) method as in [10].

6.2 Experiment Design

The experiments are designed to test the effect of the proposed techniques:

- **Balancing vs. non-balancing.** The tunable buffer (Section 4) based delay balancing (Section 5.1) is a main technique for both tuning nominal skew and facilitating link insertion. Thus, both non-balancing and balancing cases are tested in the experiments.

Case	Balancing + link				
	Delay	Skew	WL	Total cap	CPU
s9234	370	0.8	39866	6.96	32
s5378	380	0.7	43097	7.62	53
s13207	674	11.8	130074	23.13	413
Ave	475	4.4	71012	12.6	166

Table 2: Nominal results for link based buffered clock networks. The maximal source-sink delay and the maximal skew are in ps. The wirelength (WL) results are in μm. The total capacitance is in pF. CPU time is in seconds.

- **Link vs. WO-link.** Link insertion is the core technique of the proposed approach. Results with link insertion are compared with those without link insertion (WO-link).

The experimental results are organized in two parts:

- **Nominal results.** In this part, the maximal source-sink delay (ps) and the maximal skew (ps) are reported. In addition, we show the major resource usages including wirelength (**WL**) in μm, number of buffers and the CPU time (in seconds) of running our algorithms. We also report total wire and buffer capacitance which includes the tunable dummy capacitances so that there is an overall estimation on wire and buffer cost on a normalized basis.

- **Variation results.** These results are obtained from 1000 iterations of Monte Carlo simulations for each case. The maximum skew (**MS**) and standard deviation (**SD**) of skew among all iterations are recorded. In addition, the skew yield (**SY**) [13], which is the probability of satisfying a certain skew bound (**SB**), is reported.

6.3 Experimental Results

The nominal results without (with) link insertions are listed in Table 1 (Table 2). These data show that our delay balancing technique can reduce both source-sink delay and the nominal skew significantly. The link insertion does not change the nominal delay and skew very much compared to the results from delay balanced trees. This fact is in accordance with our expectation since the same delay balancing technique is applied for the tree construction and the skew tuning after inserting the link capacitors. Since the delay balancing is based on sizing the dummy capacitors within tunable buffers, we need to observe increase of the buffer capacitance including the dummy capacitance. The

	Non-balancing, WO-link				Balancing, WO-link			Balancing + link		
Case	MS	SD	SB	SY	MS	SD	SY	MS	SD	SY
s9234	175	23	50ps	0%	112	16	42%	64	10	95%
s5378	217	32	50ps	5%	95	13	38%	94	14	63%
s13207	247	33	100ps	41%	243	33	48%	180	27	61%
Norm ave	1	1		1	0.70	0.72	2.87	0.53	0.59	4.87

Table 3: Variation results. The maximal skew (MS), standard deviation (SD) and skew bound (SB) are in ps. The last row shows the normalized average values.

capacitance data indicate that the total buffer capacitance is much smaller than the total wire capacitance. Therefore, the buffer capacitance increase is often smaller than the wire capacitance reduction due to the delay balancing. The CPU time for delay balancing and the skew tuning for link insertion tends to be high as implied by the rightmost column of Table 1 and Table 2. This is mostly due to the SPICE runtime when the sizing the dummy capacitors. Since the number of clock networks on a chip is usually not large, this runtime is still reasonable for practical applications.

The variation results are shown in Table 3. One can see that the delay balancing technique can actually reduce skew variation in term of the improvement on the maximal skew (MS) and the skew yield (SY). Obviously, the link insertion can improve the standard deviation (SD) in addition and can improve the maximal skew and the skew yield more significantly. The average skew yield is improved from 15% to 73% by our method.

7. CONCLUSION

In order to cope with the increasingly significant skew variations, we propose a new skew tuning and link insertion technique for buffered clock networks. The induced multi-driver issues such as short circuit risk and multi-driver delay estimation are studied. The skew tuning technique is accurate and friendly to the link insertion methodology. Experimental results from SPICE based Monte Carlo simulations confirm the effectiveness of our approach.

8. 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*, 13(4):401–407, November 2000.

[2] R. Saleh, S. Z. Hussain, S. Rochel, and D. Overhauser. Clock skew verification in the presence of IR-drop in the power distribution network. *IEEE Transactions on Computer-Aided Design*, 19(6):635–644, June 2000.

[3] M. P. Desai, R. Cvijetic, and J. Jensen. Sizing of clock distribution networks for high performance CPU chips. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 389–394, 1996.

[4] N. A. Kurd, J. S. Barkatullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland. A multigigahertz clocking scheme for the Pentium 4 microprocessor. *IEEE Journal of Solid-State Circuits*, 36(11):1647–1653, November 2001.

[5] P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. H. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie. A clock distribution network for microprocessors. *IEEE Journal of Solid-State Circuits*, 36(5):792–799, May 2001.

[6] N. Bindal, T. Kelly, N. Velastegui, and K. L. Wong. Scalable sub-10ps skew global clock distribution for a 90nm multi-GHz IA microprocessor. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 346–355, 2003.

[7] A. Rajaram, J. Hu, and R. Mahapatra. Reducing clock skew variability via cross links. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 18–23, 2004.

[8] A. Rajaram, D. Pan, and J. Hu. Improved algorithms for link based non-tree clock network for skew variability reduction. In *Proceedings of the ACM International Symposium on Physical Design*, pages 55–62, 2005.

[9] C. Visweswariah. Death, taxes and failing chips. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 343–347, 2003.

[10] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 621–625, 2003.

[11] J. L. Neves and E. G. Friedman. Design methodology for synthesizing clock distribution networks exploiting nonzero localized clock skew. *IEEE Transactions on VLSI Systems*, 4(2):286–291, June 1996.

[12] J. P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, C-39:945–951, July 1990.

[13] X. Jiang and S. Horiguchi. Statistical skew modeling for general clock distribution networks in presence of process variations. *IEEE Transactions on VLSI Systems*, 9(5):704–717, October 2001.

[14] J. Rubinstein, P. Penfield, and M. A. Horowitz. Signal delay in RC tree networks. *IEEE Transactions on Computer-Aided Design*, CAD-2(3):202–211, July 1983.

[15] P. K. Chan and K. Karplus. Computing signal delay in general RC networks by tree/link partitioning. *IEEE Transactions on Computer-Aided Design*, 9(8):898–902, August 1990.

[16] R.-S. Tsay. An exact zero-skew clock routing algorithm. *IEEE Transactions on Computer-Aided Design*, 12(2):242–249, February 1993.

[17] Y.-M. Lee, H. Y. Lai, and C. C.-P. Chen. Optimal spacing and capacitance padding for general clock structures. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 115–119, 2001.

[18] A. Kapoor, N. Jayakumar, and S. P. Khatri. A novel

clock distribution and dynamic de-skewing methodology. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 626–631, 2004.

[19] K. Wang and M. Marek-Sadowska. Clock network sizing via sequential linear programming with time-domain analysis. In *Proceedings of the ACM International Symposium on Physical Design*, pages 182–189, 2004.

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

[21] M. Edahiro. A clustering-based optimization algorithm in zero-skew routings. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 612–616, 1993.

[22] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage. Skew and delay optimization for reliable buffered clock trees. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 556–562, 1993.

[23] J. G. Xi and W. W.-M. Dai. Buffer insertion and sizing under process variations for low power clock distribution. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 491–496, 1995.

[24] X. Zeng, D. Zhou, and W. Li. Buffer insertion for clock delay and skew minimization. In *Proceedings of the ACM International Symposium on Physical Design*, pages 36–41, 1999.

[25] R. Chaturvedi and J. Hu. Buffered clock tree for high quality IC design. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 381–386, 2004.

[26] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. SIS: a system for sequential circuit synthesis. Memorandum no. M92/41, ERL, University of California, Berkeley, May 1992.

[27] CPMO-constrained placement by multilevel optimization. http://ballade.cs.ucla.edu/cpmo/. Computer Science Department, UCLA.

[28] SPACE: VLSI physical design modeling and verification. http://space.tudelft.nl. Delft University of Technology in the Netherlands.

[29] S. P. Khatri, A. Mehrotra, R. K. Brayton, A. Sangiovanni-Vincentelli, and R. H. J. M. Otten. A novel VLSI layout fabric for deep sub-micron applications. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 491–496, 1999.

[30] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 201–204, 2000.