Clock Buffer Polarity Assignment for Power Noise Reduction

Rupak Samanta, Ganesh Venkataraman, and Jiang Hu

***Abstract*—Power/ground noise is a major source of VLSI circuit timing variations. This work aims to reduce clock network induced power noise by assigning different signal polarities (opposite switchings) to clock buffers in an existing buffered clock tree. Three assignment algorithms are proposed: 1) partitioning; 2) 2-coloring on minimum spanning tree; and 3) recursive min-matching. A post-processing of clock buffer sizing is performed to achieve desired clock skew. SPICE based experimental results indicate that our techniques could reduce the average peak current and average delay variations by 50% and 51%, respectively.**

***Index Terms*—Clock distribution, clock skew, polarity, power/ground noise.**

I. Introduction

When the supply voltage decreases with VLSI technology scaling, circuit performance becomes increasingly vulnerable to power/ground noise [1]–[3]. This problem is exacerbated by the increase in frequency and and large gate count in the scaled technologies. Based on an estimation in [4], a 0.1-V power noise may cause 80% inverter delay variation at 45-nm technology. A main culprit of power noise is clock network which keeps drawing huge current frequently from the power supply network [5]–[9]. Power/ground noise is acute at the beginning of clock cycle when FFs and the gates are switching simultaneously. In order to reduce the clock-induced power noise, few works [5]–[7], [9] attempt to avoid simultaneous flip-flop switchings through clock skew scheduling. The common approach is to spread the computation across the entire clock period so that the peak of the power/ground noise occurring at the beginning of the clock cycle is distributed across the entire clock period. In [5], [6], the flip-flops are grouped into buckets that are switched at different times. However, such an approach suffers from the limitation that the flip-flops within the same bucket still switch at the same instant of time. Moreover, the approaches [5], [6], do not consider the effect of clock skew scheduling on current profiles of combinational logic. The work [7], uses a graph based clock scheduling approach to minimize the peak current, hence the power supply noise. In [9], a circuit optimization technique called skew spreading is used to schedule the clock arrival time at each registers such that peak current is reduced. In addition to performing the clock skew scheduling, the approaches [7], [9] also consider the effect of clock scheduling on the current profiles of combinational logic. The skew scheduling approaches [5]–[7], [9] are restricted by timing constraint of the combinational logic. Since, the effectiveness of these approaches is dependent upon the available slack in the application, the power/ground noise result is expected to vary across different applications.

Manuscript received May 22, 2007; revised October 29, 2007 and February 15, 2008. First published April 21, 2009; current version published May 20, 2009. This work was supported in part by Semiconductor Research Corporation under Contract 2004-TJ-1205.

R. Samanta is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 78743 USA (e-mail: rupak9@tamu.edu).

G. Venkataraman is with Magma Design Automation, San Jose, CA 95110 USA (e-mail: ganeshv@magma-da.com).

J. Hu is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 78743 USA (e-mail: jianghu@ece.tamu.edu).

Digital Object Identifier 10.1109/TVLSI.2008.2009187

The use of on-chip decoupling capacitor to suppress the power/ground noise has been discussed in [10], [11]. The main idea is to use the charge stored in the decoupling capacitor to supply the switching transients. In another similar approach [12], the authors use a stub to suppress the power/ground noise. They attach a quarter length stub to the power supply line of the LSI chip. It acts as a band eliminate filter and suppress the power supply noise for a designed frequency.

Besides the flip-flops, the switchings of clock buffers also contribute greatly to the clock-induced noise. In a clock network of an industrial application-specific integrated circuit (ASIC) design, there could be several thousands of clock buffers [13]. A recent work [8] proposes to use different signal polarities on clock buffers so that the roughly simultaneous same-direction switchings are replaced by a mixture of opposite-direction switchings. Signal polarity refers to whether or not a signal switches in the same direction as the clock source. The main idea of [8] is illustrated in Fig. 1. In Fig. 1(a), all buffers have the same signal polarity and therefore they have either simultaneous rising switches, which draw large current from power (V_{dd}) network, or simultaneous falling switches which draw large current from ground (V_{ss}) network. In contrast, the application of opposite polarities as in Fig. 1(b) decreases current withdraw since only a half of the buffers draw current from V_{dd} while the others draw from V_{ss} at the same time. Please note that polarity assignment to a buffer is different from selecting between inverting or non-inverting type for the buffer, although these two are related. By using different types of flip-flops, positive-edge or negative-edge triggered, both signal polarities can be accommodated at flip-flops with hardly any impact to the original circuit design. For the example in Fig. 2, when the polarity of clock signal $t2$ is reversed from Fig. 2(a) to (b), the circuit timing is not affected if flip-flop FF2 is changed from positive-edge triggered to negative-edge triggered.

When assigning polarities, the work of [8] partitions the clock sinks (flip-flops) into two subsets, one for positive polarity and the other for negative polarity. Then, two subtrees are constructed separately for the two subsets, i.e., one subtree has

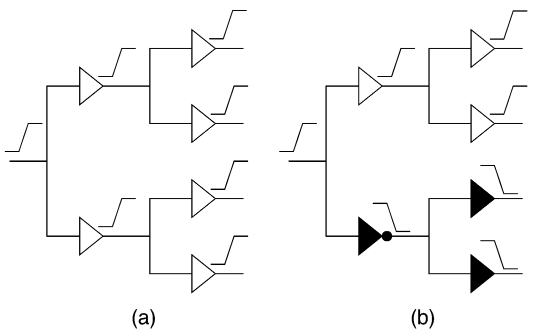

Fig. 1. All buffers in (a) have positive signal polarity and switch in the same direction. The dark buffers in (b) are assigned with negative polarity and switch in the direction opposite to the buffers with positive polarity.

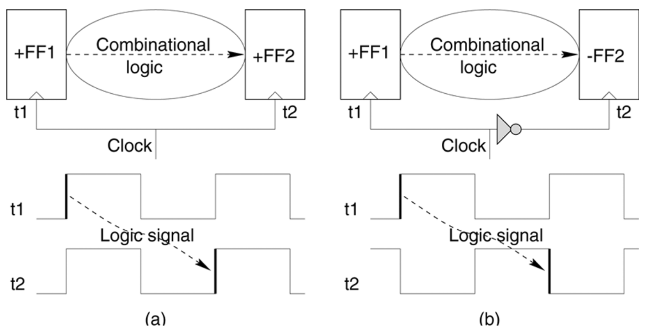

Fig. 2. (a) Positive-edge triggered FFs with original polarity. (b) FF2 is changed to negative-triggered for the reversed polarity. The circuit timing is not affected if the skew t1–t2 is maintained.

only positive polarity and the other subtree has only negative polarity. However, this approach faces a dilemma considering the following two typical scenarios.

- If the two subsets are spatially separated from each other like in Fig. 3(a), the two subtrees (one in solid lines and the other in dashed lines in Fig. 3) are in two separated regions. Except the boundary region between the two subtrees, the power noise in a local area such as the shaded regions in Fig. 3(a), is not reduced by the application of opposite polarities. This is because power noise is mostly a local effect.
- If the sink locations of the two subsets are intermingled, the approach of [8] results in two intermingled subtrees like Fig. 3(b). In this scenario, the power noise in each local region can be reduced, but the wirelength of the clock network is increased greatly.

Therefore, constructing two subtrees independently [8] either is ineffective for reducing local power noise or suffers from huge wirelength overhead. Moreover, the work of [8] evaluates only the peak current while neither power supply voltage noise nor the impact on delay variation is discussed.

In this paper, we propose to perform fine-grained clock buffer polarity assignment on an existing clock tree. We carry out a buffer type matching on the resulting clock tree to minimize the path unbalance that may arise due to buffer polarity assignment. Then, a clock buffer tuning is carried out to restore the clock skew altered by the polarity assignment. Three existing algorithms are used for polarity assignment: 1) partitioning; 2) 2-coloring on minimum spanning tree; and 3) recursive min-matching. The fine granularity of the assignment im-

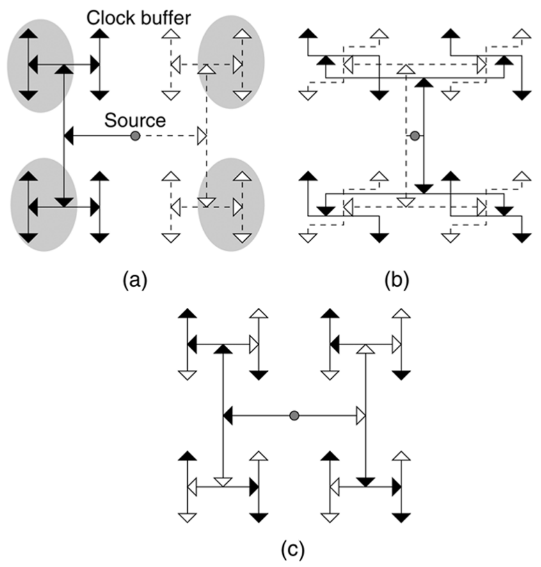

Fig. 3. Constructing two subtrees separately for opposite polarities either cannot reduce local power noise if the two subtrees are spatially apart like in (a), or results in huge wirelength overhead as in (b). We propose to perform fine-grained polarity assignment on an existing clock tree as in (c).

plies that even a very small region usually contains opposite polarities as long as there are more than one clock buffers. By doing so, the clock-induced power noise can be reduced almost everywhere. Please note, our approach does not provide an alternative to the power noise reduction using clock skew scheduling [5]–[7], [9]. Our technique can be combined with these approaches to further improve the power noise results. Also, our technique complements the power noise reduction using the decoupling capacitors [10]–[12] and can reduce the stress on the decoupling capacitor network.

SPICE-based experimental results indicate that our techniques could reduce the average peak current and average delay variations by 50% and 51%, respectively.

II. Impact to Delay Variation

Power/ground noise directly affects gate/buffer delay variation [3]. We present a first order analysis on the impact of clock buffer polarity assignment to gate/buffer delay variations. Without polarity assignment, i.e., with identical polarity for all clock buffers, all clock buffers have either simultaneous rising switchings, which cause decreased $V_{\rm dd}$ and almost no disturbance to $V_{\rm ss}$ [see Fig. 4(a)], or simultaneous falling switchings, which raise $V_{\rm ss}$ but have negligible influence on $V_{\rm dd}$ [see Fig. 4(b)]. With polarity assignment, both $V_{\rm dd}$ and $V_{\rm ss}$ degrade but with less degree [see Fig. 4(c)].

We define power noise $\Delta V_{\rm dd}$ and ground noise $\Delta V_{\rm ss}$ as

$$\Delta V_{\rm dd} = \tilde{V}_{\rm dd} - V_{\rm dd} \quad \Delta V_{\rm ss} = \tilde{V}_{\rm ss} - V_{\rm ss}$$

where $V_{\rm dd}$ and $V_{\rm ss}$ are ideal voltage values, and $\tilde{V}_{\rm dd}$ and $\tilde{V}_{\rm ss}$ are the actual voltages considering noise. As in [3], the power/ground noise can be equivalently evaluated by *differential mode noise*

$$\Delta V_{\rm dif} = \Delta V_{\rm dd} - \Delta V_{\rm ss}$$

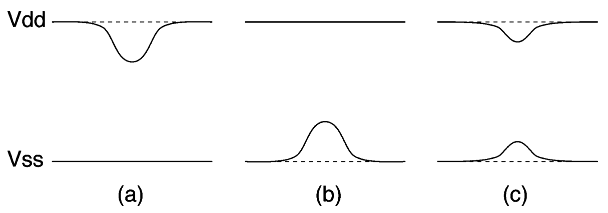

Fig. 4. Power noise in a local region when: (a) all buffers have rising switches; (b) all buffers have falling switchings; and (c) half of the buffers rising while the others falling.

TABLE I
EXAMPLE OF POWER/GROUND NOISE FOR THE THREE CASES IN FIG. 4

Case in Figure 4	ΔV_{dd}	ΔV_{ss}	ΔV_{dif}	ΔV_{com}
All rising (a)	-0.2	0	-0.2	-0.2
All falling (b)	0	0.2	-0.2	0.2
Half rising, half falling (c)	-0.1	0.1	-0.2	0

and *common mode noise*

$$\Delta V_{\text{com}} = \Delta V_{\text{dd}} + \Delta V_{\text{ss}}.$$

In Table I, we list a rough numerical example of power/ground noise for the three cases in Fig. 4. One can see that the polarity assignment does not change the differential mode noise but can reduce the common mode noise to nearly zero. According to [3], the variation of rising delay and falling delay can be expressed as

$$\Delta t_{\text{rise}} = -A \cdot \Delta V_{\text{com}} - B \cdot \Delta V_{\text{dif}} \quad (1)$$

and

$$\Delta t_{\text{fall}} = C \cdot \Delta V_{\text{com}} - D \cdot \Delta V_{\text{dif}} \quad (2)$$

respectively, where A, B, C, and D are all positive constants dependent upon device and technology parameters, the input transition time and the gate output load. The differential mode noise (ΔV_{dif}) affects the delay to charge/discharge the capacitive load at the output of the gate. The larger the value of the differential mode noise faster is the charging and discharging of the capacitive load, i.e., smaller is the gate delay. The common mode noise (ΔV_{com}) contributes to modifying the effective switching threshold of the gate. For a positive common mode noise, the switching threshold of the n/p transistors are higher than the threshold without noise, i.e., the fall delay of the gate is larger with positive common mode noise. Similarly, for negative common mode noise the threshold of n/p transistors are lower, thus, the rise delay of the gate is larger than without common mode noise. According to [3], delay variation of a gate is linearly dependent on differential mode and common mode noise. Thus, rise and fall delay variations are expressed as a linear combination of differential mode noise (ΔV_{dif}) and common mode noise (ΔV_{com}).

The three cases in Fig. 4 result in approximately the same negative value of ΔV_{dif} which contributes to roughly the same amount of delay increase. The case of Fig. 4(a) has more rising delay increase and less falling delay increase due to its negative common mode noise. Symmetrically, the case of Fig. 4(b) has less rising delay increase and more falling delay increase. In contrast, the common mode noise from the case of Fig. 4(c) is almost zero and therefore does not contribute to the delay variation. We consider the example of Table I to find the worst case delay variation of a gate. For simplicity, A, B, C, and D are assumed to be equal to 1. Substituting the values of ΔV_{dif} and ΔV_{com} for Fig. 4(a) and (b) in (1) or (2), the delay variation without polarity assignment

$$\Delta t_{w/o,\text{pol}} = 0.2 + 0.2 = 0.4.$$

The worst case delay variation of the gate due to polarity assignment can be found similarly by substituting values of ΔV_{dif} and ΔV_{com} for Fig. 4(c) in (1) or (2)

$$\Delta t_{w/o,\text{pol}} = 0.0 + 0.2 = 0.2.$$

Hence, clock buffer polarity assignment, which corresponds to Fig. 4(c), can reduce the worst case delay variation compared to using identical polarity [see Fig. 4(b) and (b)].

III. Problem Formulation

Given a buffered clock tree with n buffers, assign either positive or negative signal polarity to every buffer such that peak current reduction is maximized in any region of arbitrary size.

For a region including all of the clock buffers, this objective requires that roughly a half of the buffers have positive polarity and the others have negative polarity. For a small region containing only two clock buffers, this formulation requests one of them is positive and the other is negative.

IV. Polarity Assignment Algorithms

We have extended the application of three existing algorithms to solve the problem formulated in the previous section.

A. Partitioning

First, a graph $G = (V, E)$ is constructed with each node uniquely corresponding to a clock buffer and the node set V covers all of the clock buffers. There is an edge between every pair of nodes, i.e., this is a complete graph. Then, a bipartitioning [14] is performed on G to partition V into two disjoint subsets V_+ and V_- such that $V = V_+ \cup V_-$ and $||V_+| - |V_-|| \leq 1$. The subsets V_+ and V_- correspond to positive and negative polarities, respectively.

If two clock buffers are very close to each other, we prefer to separate them into different subsets (polarities). In a typical graph bipartitioning [14], two nodes with a small edge weight in-between are more likely to be separated into two subsets. Thus, we let the weight of edge (i, j) to be d_{ij} which is the distance between node i and j. Since a typical bipartitioning algorithm minimizes the total weight of edges in the cut, an edge with small weight (or distance) has a large chance to be in the cut and its two end nodes are separated in different subsets. The complexity of bipartitioning algorithm is $O(|V||E|)$ [14]. Since G is a complete graph, the number of edges is proportional to $|V|^2$. Thus, complexity of bipartitioning is $O(|V|^3)$.

B. 2-Coloring on Minimum Spanning Tree

This is a very simple yet effective technique. First, a minimum spanning tree is generated for the nodes representing clock buffers. Again, each edge weight is defined as the distance between its two incident nodes. Then, a 2-coloring procedure is applied on the minimum spanning tree. In 2-coloring, two end nodes of an edge are always assigned with different colors (or polarities). For a tree, there is always a feasible solution for 2-coloring and it can be found easily. Each color corresponds to a polarity. Since the minimum spanning tree algorithm chooses short edges, two nodes close to each other have opposite polarities. The minimum spanning tree is generated using greedy method. We iterate for $|V| - 1$ times to construct the minimum spanning tree, where $|V|$ is the number of nodes in the graph G. During each iteration, we chose the shortest edge from the edges connected to the nodes of the spanning tree obtained in the previous iteration. Since graph G is a complete graph, the total number of edges is proportional to $|V^2|$. Thus, the time complexity of choosing the shortest edge during each iteration is $O(|V|^2)$. Since we have $|V| - 1$ iterations, the complexity of constructing the minimum spanning tree for the graph G is $O(|V|^3)$. The 2-coloring on minimum spanning tree is $O(|E| + |V|)$ [15]. Thus, the complexity of polarity assignment due to 2-coloring on minimum spanning tree is $O(|V|^3)$.

C. Recursive Min-Matching

A graph $G = (V, E)$ same as that in Section IV-A is constructed. Performing min-matching (minimum weighted matching [16]) on this graph results in about $|V|/2$ matched node pairs. In a min-matching, the total weight of the edges between matched nodes is minimized among all possible matchings. Then, we force the two nodes (clock buffers) in the same pair to have opposite polarities. Since the min-matching algorithm normally selects pairs corresponding to small edge weight, the min-matching based polarity assignment tends to let two nearby buffers have opposite polarities.

However, requiring opposite polarities is not a complete assignment for a pair of buffers. For example, for a pair of clock buffers (a, a'), we can either let a be positive and a' be negative (denoted as (a_+, a'_-)), or let a be negative and a' be positive (a_-, a'_+). Both of the polarity permutations satisfy the constraint of being opposite. We denote the former as positive permutation $(a_+, a'_-)_+$ and the latter as negative permutation $(a_-, a'_+)_-$.

The selection of polarity permutation is decided by performing another iteration of min-matching on the node pairs obtained in the first min-matching. In this iteration, the nodes of the graph is composed by the centroids of the node pairs matched in the previous iteration. Each edge weight is defined as the distance between corresponding centroids. If two node pairs $((a, a'), (b, b'))$ is selected to be matched in this iteration of min-matching, we have four different polarity permutations: 1) $((a_+, a'_-)_+, (b_+, b'_-)_+)$; 2) $((a_+, a'_-)_+, (b_-, b'_+)_-)$; 3) $((a_-, a'_+)_-, (b_+, b'_-)_+)$; and 4) $((a_-, a'_+)_-, (b_-, b'_+)_-)$. The notations for these permutations can be abbreviated as $++, +-, -+$ and $--$. These four cases are illustrated in Fig. 5(b). It can be seen that $++$ and $--$ have no difference to the four clock buffers themselves. Similarly, $+-$ and $-+$ are equivalent to each other for the four buffers. But, $++$ and $--$ are different from $+-$ and $-+$. For the example of Fig. 5(b), it is obvious that permutation $++$ (at top) and $--$ (at bottom) are better than permutation $+-$ and $-+$ (in middle). Therefore, we choose $++$ and $--$ which are fully denoted by $((a_+, a'_-)_+, (b_+, b'_-)_+)_+$ and $((a_-, a'_+)_-, (b_-, b'_+)_-)_-$, respectively. The former is called positive permutation and the later is negative permutation. Now we have multiple node groups, each of which contains four nodes. The min-matching and polarity permutation selection can be repeated recursively on them till there is a single group containing all nodes.

We discuss how to decide polarity permutations for two matched node groups in general cases. Suppose two node groups A and B are matched. Each group has positive permutation A_+ and B_+ and negative permutation A_- and B_-. We need to choose $++/--$ or $+-/-+$ for these two groups. Each polarity permutation can be evaluated by a score which is defined as follows. For a polarity permutation such as $++$, for each node v in a group, we consider all nodes of the other group which are nearby, i.e., nodes within certain distance D from v. If a nearby node of the other group has the same polarity as v, then the score of this polarity permutation is added by 1. Such score is counted for all nodes in one group. Finally, the polarity permutation with the smallest score is selected. For the example in Fig. 5(b), the top $(++)$ and bottom $(--)$ permutations have

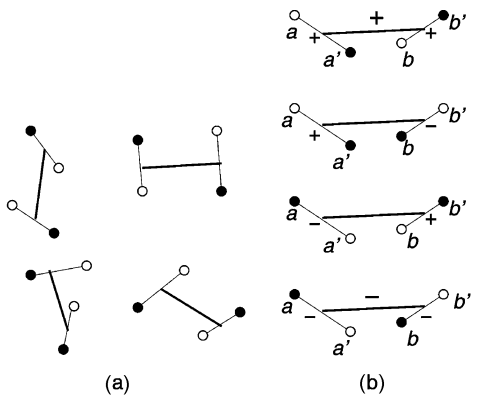

Fig. 5. Recursive min-matching.

```
Subroutine: Polarity_Assignment(G)
Input: Graph $G = (V, E)$
Output: Polarity assignment for each node $v \in V$
1. If $|V| \leq 1$, return
2. A set of node pairs $P \leftarrow Min\_Matching(G)$
3. For each pair $(u, v) \in P$
4.     $p \leftarrow$ centroid of $(u, v)$
5.     Select between $(u_+, v_+)/(u_-, v_-)$,
       $(u_+, v_-)/(u_-, v_+)$
6.     If $(u_+, v_+)/(u_-, v_-)$ is selected
           $p_+ \leftarrow (u_+, v_+)$
           $p_- \leftarrow (u_-, v_-)$
7.     Else
           $p_+ \leftarrow (u_+, v_-)$
           $p_- \leftarrow (u_-, v_+)$
8.     $V \leftarrow V - \{u, v\} + \{p\}$
9. $E \leftarrow$ all pairs in $V$
10.$Polarity\_Assignment(G)$
```

Fig. 6. Algorithm of recursive min-matching-based polarity assignment.

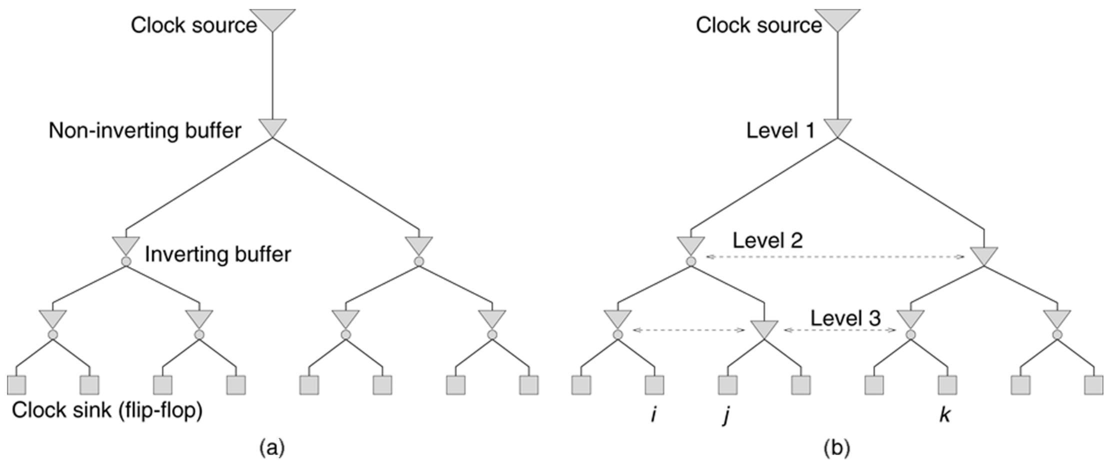

Fig. 7. (a) Original clock tree. (b) Buffer type mismatch occurs at level 2 and level 3 after polarity assignment.

score of 0 while the middle permutations ($+-$ and $-+$) have score of 1. The algorithm of the min-matching based polarity assignment is summarized in Fig. 6.

The complexity of the minimum weighted matching algorithm is $O(|V|^3)$ [16], where $|V|$ is the number of nodes in the graph G. The recursive min-matching algorithm is called $|V|/2$ times to perform polarity assignment to each clock buffer. Thus, the complexity of the recursive min-matching algorithm is $O(|V|^4)$. During each iteration of the min-matching algorithm the number of nodes reduces to half than its previous iteration. Thus, the actual number of nodes for most of the iterations is much smaller than $|V|$. Thus, the run time for the recursive min-matching algorithm is usually much faster than $O(|V|^4)$. We found the CPU run time for s35932 to be 0.603 s, which is reasonable considering the size of the circuit. We implemented the data-structure in such way that, during each iteration the informations about the pair of matched nodes are stored in new memory location. The motivation behind this implementation is to reduce the execution time of the recursive algorithm. For this purpose, we need $2|V|$ extra memory nodes for the complete execution of the algorithm. The spatial complexity is acceptable considering the negligible run time of the recursive min-matching algorithm.

V. Buffer Type Selection and Post Processing

After buffer signal polarity assignment, we need to choose either inverting or non-inverting type for each clock buffer. This procedure is straightforward. If a buffer has the same polarity as its parent buffer, it should use non-inverting type. Otherwise, an inverting type is applied.

In traditional clock tree designs, people prefer to use the same number of buffers on each source-sink path and use the same buffer type at each level [21]. This is illustrated in Fig. 7(a). Such design can make clock skew robust to inter-die process variations. However, our buffer polarity assignment may result in different buffer types at a specific level [see Fig. 7(b)]. Therefore, we try to match the buffer types without affecting signal polarity in a post processing. After the buffer type matching, buffer sizing is performed to restore the original clock skew. Both the buffer type matching and buffer sizing are focused on flip-flops which are sequentially adjacent,[1] because the fundamental timing constraints—setup time and hold constraints, are mainly for sequentially adjacent flip-flops.

A. *Buffer Type Matching*

The buffer type matching is performed for a pair of sequentially adjacent flip-flops at a time. For such a pair i and j, we check the paths from the source to i and j. If there is any buffer type mismatch between the two paths at any level [like level 3 for i and j in Fig. 7(b)], we swap the type of one mismatched buffers with the type of a nearby buffer while we try to maintain the signal polarity distribution unchanged. This procedure is repeated for every pair of sequentially adjacent flip-flops.

The overall buffer type mismatch is evaluated by a mismatch-score defined as

$$\mathrm{MismatchScore} = \sum_{i=1}^{n} \mathrm{criticality}(i,j) \cdot \mathrm{diff}(i,j)$$

where n is the total number of sequentially adjacent pairs, $\mathrm{criticality}(i,j)$ is the criticality between the sink pair (i,j) and $\mathrm{diff}(i,j)$ is the number of buffer types mismatches between the paths from sinks i and j to the root of the tree. For example, the $\mathrm{diff}(i,j)$ and $\mathrm{diff}(j,k)$ values for the clock tree of Fig. 7(b) is 1 and 2, respectively.

The clock skew between flip-flop i and j has to be within a permissible range $[L_{ij}, U_{ij}]$ to satisfy setup time and hold time constraint. The size of the permissible range is represented by $P_{ij} = U_{ij} - L_{ij}$. The distance between i and j is denoted as D_{ij}. Then, the criticality for the pair i and j is estimated by [22]

$$\mathrm{Criticality}(i,j) = \alpha \left(\frac{P_{\min}}{P_{ij}}\right) + (1-\alpha)\left(\frac{D_{ij}}{D_{\max}}\right)$$

where $\alpha \in (0,1)$ is the weight for permissible range, $P_{\min}$ is the minimum permissible range among all flip-flop pairs and $D_{\max}$ is the maximum distance among all pairs. This formula is based on the fact that the skew between a pair of flip-flops is critical if they have a small permissible range and/or they are far apart.

[1]A pair of flip-flops are sequentially adjacent if there is a pure combinational logic path in-between.

The algorithm for buffer type matching proceeds as follows. Initially, we calculate the most critical sink pair i and j that has the maximum product of $\mathrm{criticality}(i,j)$ and $\mathrm{diff}(i,j)$. The path from the two sinks to the root of the tree is traversed in a bottom-up manner and the buffers at each level are compared subsequently. If a mismatch of the buffer type is found, the nearest neighbor of each buffer type is located. The mismatch-scores of the two buffers, i.e., $\mathrm{MS}(i)$ and $\mathrm{MS}(j)$, are calculated assuming that the buffer has been swapped with its neighbor. The mismatch-scores of buffers are compared with the target mismatch-score (MS). The target mismatch-score is calculated once before the algorithm starts and it gets updated each time a buffer pair is swapped. If one or both of the mismatch-scores $(\mathrm{MS}(i), \mathrm{MS}(j))$ are smaller than the target scores then, the buffer pairs that minimizes the target score by greater amount is selected. If the selected buffer pairs on swapping maintain the initial polarity distribution intact, we qualify the buffers for swapping. A pair of buffers can retain the initial polarity distribution only if they have opposite polarities before swapping, then swapping of buffers would also cause a swapping of the polarities. Thus the initial polarity distribution is retained. However, if the swapped buffered nodes have same polarity, then the swap would cause both the polarities to invert and thus alters the initial polarity distribution. The swapping of buffer also affects polarity distribution of the subtree with the swapped buffers as its root. Thus, we traverse from each swapped buffer in a top-down fashion and invert the buffer type of each buffer node in the subtree to maintain the original polarity.

If a buffer pair is qualified for swapping, we update the target mismatch-score with the updated scores, i.e., either $\mathrm{MS}(i)$ or $\mathrm{MS}(j)$, and start over again by finding the critical most sink. On the other hand, if none of the buffer pair is qualified for swapping, then we continue the bottom-up traversal till next mismatch in the buffer type is found or the root node is encountered. If we reach the root then the next critical sink pair is selected and the same process is repeated for the selected sink pairs. If all the sink pairs are exhausted, then no further improvement in mismatch-score is possible and the algorithm stops.

B. Clock Skew Tuning

Since the original clock skew is changed due to the buffer type change in the polarity assignment, we run a clock skew tuning procedure after the buffer type matching to restore the original clock skew. This tuning procedure is same as [17] where the sizes of dummy capacitors are tuned toward desired clock skew. Therefore, the wirelength is not affected in this tuning. Although the area of dummy capacitance is increased, the buffer capacitance is often reduced when non-inverting buffers are replaced by inverting buffers in the polarity assignment. Hence, the overall capacitance is rarely increased as indicated in the experimental results.

VI. Experimental Results

The proposed procedure for power noise reduction was implemented in C on a Linux machine with 2 dual-core Intel Xeon processors of 3.2 GHz and 8 GB RAM. We performed experiments on two sets of benchmark circuits: (a) ISCAS89 sequential circuits and (b) r1–r5 downloaded from GSRC Bookshelf [18]. The reason we employ ISCAS89 benchmark is that it has logic information and the reason for r1–r5 benchmark is that it is larger in size. The characteristics of the test cases are shown in Table II. The table indicates the number of clock sinks and the buffers for each test case.

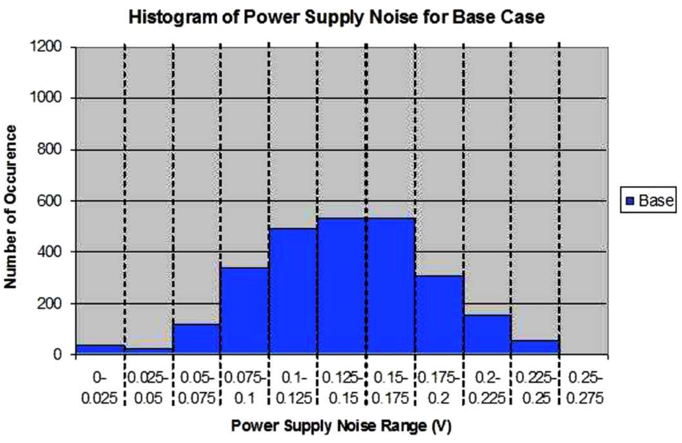

Fig. 8. Histogram of power supply noise for base case of s38584.

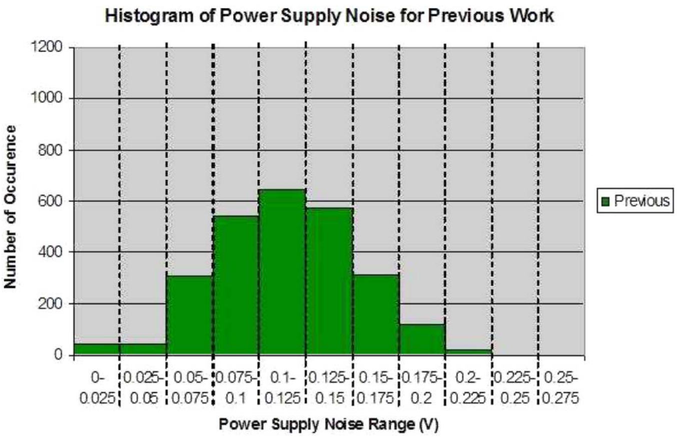

Fig. 9. Histogram of power supply noise for previous work of s38584.

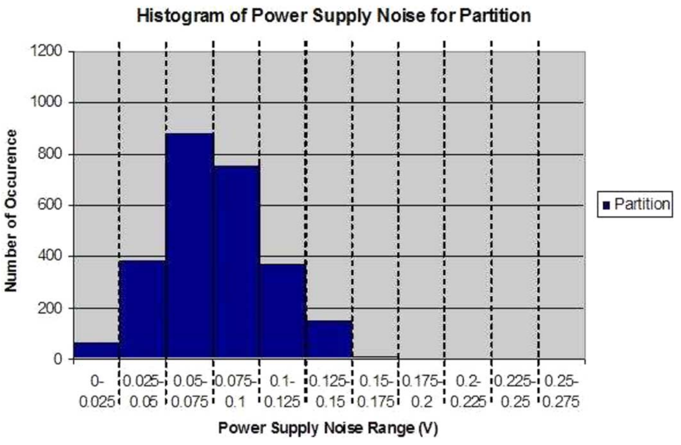

Fig. 10. Histogram of power supply noise for partition of s38584.

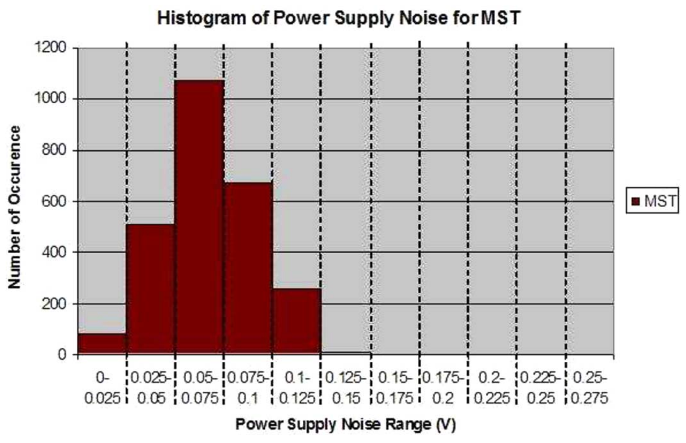

Fig. 11. Histogram of power supply noise for MST of s38584.

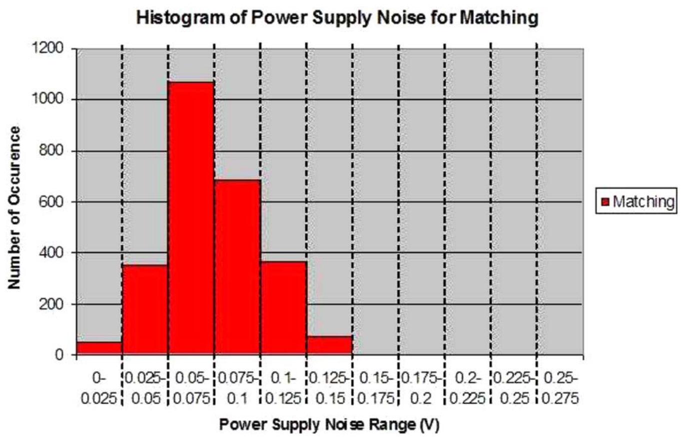

Fig. 12. Histogram of power supply noise for matching of s38584.

For ISCAS89 benchmark circuits, the combinational logic gates were synthesized in Design Analyzer from Synopsys and placed using Silicon Ensemble from Cadence. The placement of the logic gates were done for 180-nm library downloaded from the website [19]. The clock tree is then constructed using DME [20] and the clock buffers are placed similar as [21]. The logic gates were replaced by time-varying current sources connected between power and ground at grid points determined from the placement result of Silicon Ensemble. We replace the logic gates by current sources to make the power grid simulation feasible. The clock buffers are connected to the power and ground grids at locations determined from the clock tree construction. For SPICE simulation, we used 180-nm model card obtained from [23] and $V_{\rm dd}$ was set to 2.5 V. We chose 180-nm model card for SPICE simulation to maintain consistency between placement result and the SPICE simulation.

The r1–r5 benchmark circuits were obtained from [18]. The buffered clock tree was generated using the algorithm in [17]. Since r1–r5 does not have logic gate information, we conducted experiments on a standalone clock tree. For SPICE simulation, we used 65-nm BSIM4 model card obtained from [23] and set $V_{\rm dd}$ to 1.0 V.

To measure the effectiveness of our technique, we perform simulations to determine the peak current, power supply noise, delay variation, power consumption, total capacitance and

TABLE II
TESTCASES

Case	# Sinks	# Buffers
S9234	135	20
S5378	164	25
S13207	503	77
S38584	1426	235
S35932	1728	286
r1	267	37
r2	598	171
r3	861	59
r4	1903	303
r5	3101	441

global skew. We measure the previously mentioned parameters for base case (initial clock tree with no polarity assignment), previous work [8] and the three algorithms proposed. For each parameters, we insert several sampling points in the circuit to measure the value during SPICE transient simulation. We record the worst case at each sampling point. For power noise, peak current and power consumption parameters, the sampling points are on the power grid. The sampling points for skew measurement is set at the sink locations. For delay variation, we introduce few logic gates into our simulation structure. The logic gates are connected to power and ground grids at points selected randomly.

The results for ISCAS89 and r1–r5 benchmark circuits are summarized in Tables III–five columns. The first set of columns presents the results of the base case. The second set presents the previous work [8]. This is followed by a set of three columns that present the the three proposed algorithms, i.e., Partition, MST, and Matching. In Tables III–, and IX, we report normalized averages for both the benchmarks separately. The final row in each table shows the normalized average for r1–r5 benchmark circuits, the other row with normalized average is for ISCAS89 benchmark. The normalized average is used to compare our procedure with the base case as well as the technique described in [8]. For Tables VII and VIII, we indicate the average values.

In the data Tables III–VI, and VIII, we report the average and the maximum results among these worst case values from different sampling points. In the data Table IX, we report the resource consumption for both benchmark circuits. For each of the five cases in Table IX, we report average value of the total power consumption, total capacitance, and CPU run time, respectively. The total capacitance is the sum of tuning capacitance and buffer capacitance. For CPU run time, we include the CPU run time to generate SPICE files for each of the algorithm and does not include the SPICE run time.

We post process the clock tree obtained (after assigning different polarities) to tune the skew by techniques suggested in [17]. By doing that, we bring the skew to be less than the required skew bound for all the test cases. The skew bound was set to 35 ps for all the test cases. The skew results are reported in Table VII.

The skew due to variation was determined for ISCAS89 benchmark circuits. The result for skew was obtained by running 1000 Monte Carlo simulations for each case. The following parameters are varied: (a) channel length; (b) threshold voltage; and (c) temperature. The above parameters were varied with mean as nominal value and a standard deviation of 3%. In

TABLE III
RESULTS FOR PEAK CURRENT (mA)

Case	Base Case		Previous Work [8]		Partition		MST		Matching	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
s5378	50.2	108.1	42.2	87.0	21.9	47.3	24.1	51.7	23.1	52.9
s9234	41.30	69.4	34.6	60.1	22.0	40.1	19.1	33.4	18.0	32.0
s13207	127.2	222.3	115.6	210.0	79.4	140.1	62.3	109.8	69.0	130.0
s35932	95.1	154.1	90.3	180.5	53.0	92.4	52.4	85.5	51.4	94.4
s38584	87.5	144.9	73.9	122.9	51.1	89.3	44.6	72.0	50.2	88.6
Nor Ave.	1.00	1.00	0.87	0.88	0.55	0.56	0.50	0.50	0.51	0.53
r1	15.9	24.5	12.4	20.2	7.9	11.9	8.7	14.3	8.5	13.4
r2	43.8	82.1	35.1	71.1	26.1	46.3	25.8	45.7	26.1	48.1
r3	21.6	37.0	20.4	37.5	11.2	20.0	12.0	20.4	11.2	28.5
r4	80.7	156.4	70.5	138.4	46.2	85.9	40.7	74.8	40.7	77.4
r5	111.9	156.7	94.6	132.7	64.2	90.1	63.6	90.8	60.8	85.0
Nor Ave.	1.00	1.00	0.85	0.88	0.55	0.54	0.54	0.54	0.54	0.53

TABLE IV
RESULTS FOR POWER NOISE (mV)

Case	Base Case		Previous Work [8]		Partition		MST		Matching	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
s5378	42.2	92.8	39.8	83.8	21.5	48.8	23.1	46.6	22.7	45.8
s9234	34.3	69.1	27.5	62.5	20.1	50.7	16.4	50.1	15.5	50.1
s13207	170.0	247.8	149.2	239.0	97.5	143.1	80.9	125.6	91.2	91.2
s35932	169.6	298.0	155.4	295.2	92.1	182.4	88.8	167.7	85.2	167.5
s38584	140.0	255.0	114.0	219.4	76.4	183.1	67.2	143.1	73.9	159.2
Nor Ave.	1.00	1.00	0.87	0.92	0.55	0.63	0.50	0.57	0.51	0.60
r1	6.2	10.1	5.4	9.2	3.0	4.4	3.4	4.9	3.2	4.3
r2	15.6	25.7	12.8	23.2	10.0	16.5	10.0	16.1	10.2	17.7
r3	9.6	13.4	8.8	13.0	5.0	8.1	5.6	8.9	5.2	7.6
r4	36.5	51.7	30.2	43.3	22.5	32.8	18.8	27.2	18.5	26.0
r5	61.4	79.1	53.4	75.4	32.4	43.8	43.8	30.5	41.2	42.7
Nor Ave.	1.00	1.00	0.86	0.91	0.56	0.57	0.55	0.56	0.54	0.54

TABLE V
RESULTS FOR GROUND NOISE (mV)

Case	Base Case		Previous Work [8]		Partition		MST		Matching	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
s5378	44.8	83.6	39.7	78.2	20.3	53.0	22.5	42.4	21.1	42.5
s9234	31.3	63.2	27.4	60.1	19.5	39.6	14.5	39.3	13.4	39.4
s13207	156.3	227.8	147.2	227.3	104.1	156.1	79.0	122.1	85.1	129.0
s35932	154.0	295.0	152.0	290.0	87.7	174.5	88.1	176.7	87.8	184.9
s38584	128.1	245.4	110.1	193.9	77.4	168.9	69.1	140.2	79.2	168.3
Nor Ave.	1.00	1.00	0.91	0.93	0.58	0.64	0.51	0.56	0.52	0.60
r1	7.1	10.8	6.1	10.3	3.2	4.5	3.6	5.1	3.9	4.7
r2	18.5	30.3	14.6	27.0	10.4	17.8	10.1	16.9	10.3	17.8
r3	10.6	15.3	10.2	15.0	5.6	9.0	5.8	9.2	5.4	8.3
r4	39.8	56.4	33.8	51.3	21.4	30.8	19.6	28.4	19.9	28.6
r5	58.5	78.2	53.7	72.4	31.9	42.9	30.0	40.5	30.5	41.5
Nor Ave.	1.00	1.00	0.87	0.93	0.52	0.54	0.52	0.53	0.53	0.52

Table VIII, we report the average and maximum values of the skew due to variation.

The histograms of power supply noise for the testcase s38584 are shown in Figs. 8–12, respectively. We consider 2600 sampling points for each case. The x-axis represents the power supply noise range, each range is of 0.025 V width; the y-axis reports the number of sampling points within a range. From Fig. 8, the largest power supply noise range for the base case is 0.25–0.275 V and most of the sampling points are within the range 0.1–0.175 V. The largest power supply noise range for the previous work (Fig. 9) is 0.2–0.225 V and most frequent power supply noise range for sampling points is 0.075–0.15 V. The histograms in our approaches have narrow distribution of power supply noise and most of the sampling points are in low noise ranges. For MST (see Fig. 11), the largest power supply noise range is 0.125–0.15 V and most of the sampling points are in the range 0.05–0.1 V. Due to large volume of results, we show histograms for one representative test case, s38584 is chosen due to its larger size.

The following observations could be drawn from the results.

- Our techniques clearly dominate the method suggested in [8] in terms of peak current, power supply noise, delay variation and power consumption.
- The reduction in peak current is significant, 46%–50% and 45%–47% respectively for ISCAS89 and r1–r5 benchmark circuits. In fact, in few cases it could lead up to more than 50% peak current reduction. Such high reductions in peak current have a direct positive impact on circuit reliability.

TABLE VI
RESULTS FOR DELAY VARIATION (PS)

Case	Base Case		Previous Work [8]		Partition		MST		Matching	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
s5378	0.72	0.96	0.52	0.73	0.33	0.36	0.28	0.30	0.25	0.29
s9234	0.50	0.70	0.39	0.57	0.35	0.45	0.31	0.42	0.33	0.43
s13207	1.60	2.10	1.40	2.00	1.05	1.20	0.80	0.91	0.81	0.93
s35932	3.30	3.60	3.10	3.30	1.62	1.81	1.51	1.70	1.43	1.60
s38584	2.80	2.90	2.56	2.71	1.55	1.70	1.41	1.59	1.50	1.70
Nor Ave.	1.00	1.00	0.84	0.87	0.57	0.53	0.49	0.46	0.50	0.48
r1	0.41	0.48	0.38	0.40	0.21	0.24	0.19	0.23	0.22	0.25
r2	1.29	1.33	1.23	1.32	0.75	0.77	0.80	0.83	0.76	0.78
r3	0.53	0.6	0.42	0.47	0.28	0.33	0.27	0.32	0.25	0.30
r4	2.00	2.16	1.95	2.10	1.05	1.15	1.02	1.06	1.10	1.21
r5	2.98	3.24	2.71	2.95	1.54	1.62	1.65	1.80	1.70	1.82
Nor Ave.	1.00	1.00	0.91	0.90	0.53	0.53	0.53	0.53	0.54	0.54

TABLE VII
RESULTS FOR NOMINAL SKEW (PS)

Case	Base Case	Previous Work [8]	Partition	MST	Matching
s5378	4.9	15.7	16.0	17.7	20.0
s9234	4.0	14.1	22.0	18.6	19.0
s13207	10.2	12.4	15.5	18.3	19.8
s35932	30.0	33.0	35.0	25.0	35
s38584	28.0	35.0	29.0	35.0	32.0
Ave.	15.4	22.0	23.5	22.9	25.2
r1	5.4	7.2	9.6	9.0	10.2
r2	31.5	31.4	35.2	31.5	35.0
r3	20.0	18.5	19.0	14.1	18.6
r4	12.5	27.2	19.4	22.8	19.3
r5	12.8	12.4	11.7	19.6	17.8
Ave.	16.5	19.3	19.0	19.4	20.2

TABLE VIII
RESULTS FOR SKEW DUE TO VARIATION FOR ISCAS89 (PS)

Case	Base Case		Previous Work [8]		Partition		MST		Matching	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
s5378	48.1	93.6	50.0	94.6	46.4	84.8	50.1	92.0	49.8	90.5
s9234	5.5	6.8	17.9	19.0	19.2	20.1	18.7	19.2	19.0	19.4
s13207	76.2	116.8	72.7	105.1	76.0	115.5	75.4	109.2	72.9	106.1
s35932	184.6	265.0	173.4	278.9	181.1	265.0	166.5	248.7	183.5	267.0
s38584	129.5	182.4	173.4	193.1	133.2	199	123.5	177.8	136.4	189.3
Ave.	88.8	139.9	90.4	138.1	91.2	136.9	86.9	129.4	92.3	134.5

TABLE IX
RESULTS FOR TOTAL RESOURCE CONSUMPTION (POWER IN MILLIWATTS, CAP IN PF AND CPU TIME IN SECONDS)

Case	Base Case			Previous Work [8]			Partition			MST			Matching		
	Power	Cap	CPU	Power	Cap	CPU	Power	Cap	CPU	Power	Cap	CPU	Power	Cap	CPU
s5378	38.2	7.72	0.200	36.4	7.59	0.210	23.7	4.62	0.253	26.6	5.11	0.275	27.4	5.31	0.245
s9234	31.8	6.10	0.190	30.2	5.71	0.195	25.0	4.86	0.240	22.3	3.81	0.256	21.0	3.46	0.245
s13207	117.8	56.0	0.249	105.6	54.7	0.301	94.6	50.5	0.320	88.5	47.6	0.331	88.2	46.1	0.315
s35932	526.0	129.2	0.545	548.0	157.3	0.712	480.0	97.9	0.745	443.6	130.0	0.81	471.0	127.2	0.803
s38584	394.9	90.00	0.539	409.2	107.8	0.695	330.3	67.40	0.701	294.0	82.40	0.750	313.6	73.70	0.719
Nor Ave.	1.00	1.00	1.00	0.97	1.06	1.17	0.79	0.76	1.29	0.75	0.81	1.38	0.76	0.77	1.32
r1	1.4	0.41	0.340	1.4	0.52	0.350	1.1	0.42	0.480	1.1	0.42	0.460	1.0	0.46	0.440
r2	6.1	1.90	0.615	6.4	3.34	0.750	4.4	1.84	0.723	4.5	2.13	0.810	4.3	1.81	0.796
r3	2.7	0.75	0.450	2.8	1.01	0.510	2.0	0.84	0.535	2.2	0.95	0.565	2.1	0.84	0.490
r4	10.6	3.35	0.645	10.7	4.85	0.754	7.8	3.74	0.812	8.2	5.21	0.843	7.5	5.13	0.821
r5	14.0	4.90	1.400	14.9	6.10	1.560	10.5	5.34	1.610	10.9	5.77	1.730	10.0	5.13	1.712
Nor Ave.	1.00	1.00	1.00	1.04	1.41	1.13	0.75	1.06	1.23	0.78	1.22	1.29	0.73	1.14	1.23

- The power supply noise and ground noise come down by 45%–50% (45%–46%) and 42%–49% (47%–48%), respectively, for ISCAS89 (r1–r5) benchmark. For s38584, a power supply noise reduction of 52% is achieved. This result indicates that our algorithm is efficient in reducing power supply noise even for bigger and more practical circuits.
- The delay variation reduces by 43%–51% and 46%–47%, respectively, for ISCAS89 and r1–r5 benchmarks. The impact of our algorithm for delay variation reduction is higher

for the bigger clock networks. This trend is encouraging as it indicates that our algorithm scales favorably for bigger nets.

- The total power consumption reduces by 21%–25% (22%–27%) for the three different algorithms. The power reduction can be attributed to the use of different buffer types in the polarity assigned clock tree.
- The total capacitance reduction is in the range 19%–24% for ISCAS89 benchmark. However, for r1–r5 the capacitance increases in the range 6%–22%. The increase in capacitance for our case in r1–r5 is due to smaller buffer sizes used for 65-nm technology. Thus the tuning capacitance is more than the total buffer capacitance.
- The nominal skew values for three different algorithms are reduced to that of the base case. The skew bound is set to 20 ps for small and medium sized testcases, the skew bound is 35 ps for the large test cases. The only exeception is for r_2 testcase. Since the skew of initial tree is poor for r_2, we chose the skew to be 35 ps instead of 20 ps bound.
- The skew due to variation is calculated for ISCAS89 benchmark circuits. The average and maximum value of skew in our case is almost same as that for the base case. Thus the proposed technique has similar variation tolerance as the base case. For few testcases our algorithm is found to reduce the skew due to variation compared to base case. As an examples for s35932, the average skew due to variation for MST algorithm is reduced by 10% compared to the base case. Thus, in addition to reducing peak current and power supply noise, in few cases, our algorithm can also provide more robustness to skew due to variation.
- Since the run-time is negligible, our technique offers the flexibility of trying all three approaches and picking the one that offers the best results.

VII. Conclusion

In this paper, we propose techniques to reduce the clock network induced power supply noise by assigning different polarities to the clock buffers in an existing clock tree. We detail three different algorithms for the same problem. Experimental results indicate significant reduction in peak current, power supply noise and delay variations. Such reductions in peak current and delay variations lead to more reliable clock networks.

Acknowledgment

The authors would like to thank S. Ahmed from Freescale Semiconductor for some helpful discussions. They would also like to thank the reviewers for their valuable comments and identification of the novelty of the proposed work and recognizing the challenges faced in demonstrating its feasibility.

References

[1] 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 Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 6, pp. 635–644, Jun. 2000.

[2] G. Bai, S. Bobba, and I. N. Hajj, "Static timing analysis including power supply noise effect on propagation delay in VLSI circuits," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2001, pp. 295–300.

[3] L. H. Chen, M. Marek-Sadowska, and F. Brewer, "Buffer delay change in the presence of power and ground noise," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 3, pp. 461–473, Jun. 2003.

[4] S. S. Sapatnekar and H. Su, "Analysis and optimization of power grids," *IEEE Des. Test Comput.*, vol. 20, no. 3, pp. 7–15, May–Jun. 2003.

[5] A. Vittal, H. Ha, F. Brewer, and M. Marek-Sadowska, "Clock skew optimization for ground bounce control," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1996, pp. 395–399.

[6] L. Benini, P. Vuillod, A. Bogliolo, and G. De Micheli, "Clock skew optimization for peak current reduction," *J. VLSI Signal Process.*, vol. l16, no. 2/3, pp. 117–130, Jun./Jul. 1997.

[7] W.-C. D. Lam, C.-K. Koh, and C.-W. A. Tsao, "Power supply noise suppression via clock skew scheduling," in *Proc. IEEE Int. Symp. Quality Electron. Des.*, Mar. 2002, pp. 355–360.

[8] Y.-T. Nieh, S.-H. Huang, and S.-Y. Hsu, "Minimizing peak current via opposite-phase clock tree," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2005, pp. 182–185.

[9] Z. Yu, M. C. Papaefthymiou, and X. Liu, "Skew spreading for peak current reduction," in *Proc. ACM Great Lakes Symp. VLSI*, Mar. 2007, pp. 461–464.

[10] M. Grazino, G. Masera, G. Piccinini, and M. Zamboni, "Automated power supply noise reduction via optimized distributed capacitors insertion," in *Proc. IEEE Southw. Symp. Mixed-Signal Des.*, Feb. 2001, pp. 167–172.

[11] S. Zhao, K. Roy, and C.-K. Koh, "Decoupling capacitance allocation for power supply noise suppression," in *Proc. ACM Int. Symp. Phys. Des.*, Apr. 2001, pp. 66–71.

[12] T. Nakura, M. Ikeda, and K. Asada, "Preliminary experiments for power supply noise reduction using stubs," in *Proc. IEEE Asia-Pac. Conf. Adv. Syst. Integr. Circuits*, Aug. 2004, pp. 286–289.

[13] K. Wang, Y. Ran, H. Jiang, and M. Marek-Sadowska, "General skew constrained clock network sizing based on sequential linear programming," *IEEE Trans. Comput.-Aided Des.*, vol. 24, no. 5, pp. 773–782, May 2005.

[14] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: A survey," *Integr.: VLSI J.*, vol. 19, no. 1–2, pp. 1–81, Aug. 1995.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms.*. Cambridge, MA: The MIT Press, 2001.

[16] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.*. New York: Dover, 1998.

[17] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri, A. Rajaram, P. McGuinness, and C. Alpert, "Practical techniques to reduce skew and its variations in buffered clock networks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 591–595.

[18] GSRC Bookshelf, BST, "Bounded-skew clock tree routing," Univ. California, San Diego, 2002. [Online]. Available: http://vlsicad.ucsd.edu/GSRC/bookshelf/slot/BST

[19] VLSI Computer Architecture Research, Stillwater, OK, "FreePDK: Unleashing VLSI to the masses," 2005. [Online]. Available: http://vcag.ecen.okstate.edu/projects/scells/download/

[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 Trans. Circuits Syst. II, Analog Dig. Signal Process.*, vol. 39, no. 11, pp. 799–814, Nov. 1992.

[21] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage, "Skew and delay optimization for reliable buffered clock trees," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1993, pp. 556–562.

[22] G. Venkataraman, C. N. Sze, and J. Hu, "Skew scheduling and clock routing for improved tolerance to process variations," in *Proc. ACM/IEEE Asia South Pac. Des. Autom. Conf.*, Jan. 2005, pp. 594–599.

[23] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 2000, pp. 201–204.

Rupak Samanta received the B.E. degree (honors) in electrical engineering from Sambalpur University, Orissa, India, in 2000, and the M.Tech. degree (honors) from Indian Institute of Technology (IIT), Mumbai, India, in 2003. He is currently pursuing the Ph.D. degree in computer engineering from Texas A&M University, College Station.

Between 2003 to 2004, he worked as a Design Engineer with Agere Systems, Bangalore, India. He is currently an intern Circuit Designer with Intel, Austin, TX. His research interests include physical design, variation tolerant circuit design, and low power techniques.

Ganesh Venkataraman received the B.E. degree (honors) in electrical and electronics engineering and the M.Sc. degree (honors) in physics from the Birla Institute of Technology and Science, Pilani, India, both in 2000, the M.S. degree in electrical and computer engineering from University of Iowa, Iowa City, in 2003, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2007.

Since 2007, he has been with Magma Design Automation, San Jose, CA, where he is currently a Senior Member of the Technical Staff. His research interests include VLSI physical design/optimization, variation tolerant clock distribution, low power, graph theory, and combinatorial optimization. He has several publications related to VLSI physical design, process variations and low power.

Dr. Venkataraman was a recipient of the prestigious Graduate Merit Fellowship from Texas A&M University.

Jiang Hu (S'98–M'01) received the B.S. degree in optical engineering from Zhejiang University, Zhejiang, China, in 1990, and the M.S. degree in physics and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 1997 and 2001, respectively.

Currently, he is an Assistant Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station. From 2000 to 2002, he was with IBM Microelectronics Division, Austin, TX, where he received six U.S. patents. His research interests include computer-aided design for VLSI circuits, especially on physical design for robustness and manufacturability.

Dr. Hu was a recipient of an IBM First Plateau Invention Achievement Award from IBM Microelectronics Division and a Best Paper Award from the ACM/IEEE Design Automation Conference in 2001. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He has served as technical program committee member of DAC, ICCAD, ISPD, DATE, ISQED, ICCD, and ISCAS.