INTEGRATION, the VLSI journal 45 (2012) 121–131

Contents lists available at SciVerse ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

A new clock network synthesizer for modern VLSI designs ☆

Jingwei Lu *, Wing-Kai Chow, Chiu-Wing Sham *

Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Article history:
Received 16 February 2011
Received in revised form
3 November 2011
Accepted 4 November 2011
Available online 12 November 2011

Keywords:
Clock tree synthesis
Process variations

ABSTRACT

In nanometer-scale VLSI physical design, clock tree becomes a major concern on determining the total performance of the chip. Both the clock skew and the PVT (process, voltage and temperature) variations contribute a lot to the behavior of the digital circuits. Previous works mainly focused on skew and wirelength minimization. However, it may lead to negative influence on the variation factors. In this paper, a novel clock tree synthesizer is proposed for performance improvement. Several algorithms are introduced to tackle the issues accordingly. A dual-MST geometric approach of perfect matching is developed for symmetric clock tree construction. In addition, a special technique of buffer sizing is also introduced. These two techniques can help balancing the tree structure in order to reduce the variation effect. An iterative buffer insertion technique and the dual-MZ blockage handling technique are also presented. They are developed for proper distribution of buffers and connection of wires, so the dynamic power consumption can be reduced. Additionally, slew table construction and internal nodes relocation are involved to satisfy the slew rate constraint and further reduce the clock skew. Experimental results show that the performance of our synthesizer is better than those of the previous works.

1. Introduction

Clock signals are employed in VLSI digital systems to synchronize the active components of a design. On behalf of this synchronization, a tree is constructed to deliver the clock signal to every sequential cell of the digital circuits. Clock skew represents the timing difference of all the terminals in the clock net. In order to synchronize one circuit, each terminal must be reached within a specified small time range. Otherwise, the extent of synchronization would become unacceptable. Clock skew is a major concern in the clock tree synthesis. It would affect the maximum attainable frequency of operation in the circuits. The clock skew should be maintained inside prescribed latency range in order to ensure the digital function of the system. Additionally, the clock signal is also the single largest source of dynamic power usage in the system. Extra attention should be paid on power cost reduction in the clock tree.

Clock skew minimization is a popular research topic during the past decades. Plenty of research achievements have been proposed by previous researchers. Some earlier proposed works concentrated on the distribution of wirelength between the source and the sinks to achieve delay equalization. Jackson [1] first presented a clock routing algorithm based on a sub-optimal equidistant network, which is constructed in a top-down flow. Later, more improvements were made to reach exact equidistant tree [2] for clock net. A path-length skew balancing approach was proposed in [3], which first introduced the bottom-up geometric subtree merging. Afterwards, delay balancing using Elmore delay model [4] became prevalent to acquire more accurate delay estimates. Clock tree with exact zero skew [5] was proposed by applying balancing method based on the Elmore delay model. The deferred-merging and embedding (DME) technique [6,7] was proposed to achieve the zero clock skew with a shorter wirelength in the clock tree. Later, a clustering based topology generation method [8] for clock tree routing is introduced, which effectively reduce the wirelength within runtime of only $O(N^2)$. In [9] a buffer insertion technique was discussed within runtime of $O(N^2)$, which could generate a buffer distribution with minimal phase delay achieved. This technique was further improved in [10] with $O(N \log N)$ runtime achieved. The problem of routing with restrictions on buffer locations is discussed in [11], which propose a algorithm to achieve the minimal delay in polynomial-time.

In recent years, tolerance on variation became a focus of attention. This is mainly due to the uncertainty of various factors inside a circuit, such as process [12], voltage [13] and temperature [14]. In order to keep a chip stable and functioning well, methods with greater tolerance on variation are widely favored.

☆The work described in this article was partially supported by the General Research Fund from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 5263/10E).

* Corresponding authors.
E-mail addresses: francesco.ljw@gmail.com (J. Lu), williamchowhk@gmail.com (W.-K. Chow), encwsham@poly.edu.hk (C.-W. Sham).

0167-9260/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2011.11.001

Many researchers focused on robust algorithms for variation minimization. Techniques such as wire sizing [15], buffer sizing [16] and link insertion [17,18] are applied. In ISPD 2009 clock network synthesis contest [19], a voltage variation related objective named clock latency range (CLR) was formulated. Several new benchmarks were released accordingly. Subsequently, some related research works were proposed [20–22]. A dual-MST topology generation approach was proposed in [22]. Its objective is to minimize the maximal edge cost during the weighted perfect geometric matching. Besides, several clock gating approaches in logic level [23], register transfer level [24] and physical level [25,26] were proposed. These research works developed clock gating approaches for power reduction in the clock tree.

In this paper, we propose a novel clock network synthesizer.[1] Some heuristics are proposed to optimize CLR as well as clock skew and power usage. It mainly contains the following features: (1) A dual-MST perfect matching algorithm is developed for symmetric clock tree construction. (2) A look-up table based buffer sizing approach is developed for variation tolerance improvement. (3) An iterative buffer insertion technique is developed for delay balancing and capacitance reduction. (4) A dual-MZ blockage handling technique is developed for buffer location distribution and blockage avoidance. SPICE simulation is also applied for accurate delay estimation, the internal nodes of the clock tree can be relocated for further skew minimization. Strict slew constraint (≤ 100 ps) is employed in our synthesizer. We build a slew table for real-time reference during the execution of our program to satisfy this constraint.

In addition to the measurement of CLR, simulation of corner analysis is also applied in the experiments. Simulation of corner analysis can simulate the variation of buffers in different situation. It can provide a common method for performance evaluation of our clock network synthesizer while the process variation of buffer is considered. It can also ensure that our proposed approach is practically variation-tolerant.

The remainder of this paper is organized as follows. Section 2 includes the problem formulation. Section 3 is composed of the major components of our synthesizer: topology generation, buffer sizing, buffer insertion and blockage handling, etc. In Section 4, we present our results based on the ISPD 2009 contest and some other standard benchmarks. Finally, we give our conclusion in Section 5.

2. Problem formulation

Let $T=\{V,E\}$ denote the clock tree. $V=\{v_i|i=1,2,\ldots,m_v\}$ is the set of nodes, and $E=\{e_j|j=1,2,\ldots,m_v-1\}$ is the set of clock edges between the node v_j and its corresponding parent. Let $|e_j|$ denote the length of the edge e_j. We use $S=\{v_k|k=1,2,\ldots,m_s\}$ (where $m_s<m_v$) to denote the set of modules (the sinks, or leaf nodes). d_{s_i} denotes the signal delay of the ith sink. The rest (m_v-m_s) nodes are named internal nodes. We use $|V|$, $|E|$ and $|S|$ to indicate the number of elements in V, E and S, respectively. The leaves are at level 0, and the root is at the highest level. Node v_i is said to be at level n_i if there are n_i edges on the path from v_i to the farthest leaf of the tree. Moreover, we assume that the topology of the clock tree is full binary, and every internal node has exactly two children. The skew of T is the difference between the longest signal delay and the shortest signal delay from the source to any sinks. Detailed definition can be found in the ISPD 2009 contest [19].

2.1. Clock slew rate

The restriction on clock slew describes the requirement on the signal transition time reduction. It is defined to be the rising time from 10% to 90% of the signal strength (90–10% for the falling time, respectively). The upper limit is set to be 100 ps. During the clock tree synthesis, it is necessary to maintain the signal transition time under this upper limit throughout the whole network.

2.2. CLR

Voltage uncertainty is quite common in the real clock trees. The supplied voltage at each driving node may vary between a wide range (V_{dd_1},V_{dd_2}). Changes on the power supply would cause buffer delay to vary from assumption. Clock tree with zero skew could still suffer from unbalancing problems. Two independent SPICE simulations, under different voltage source (V_{dd_1} or V_{dd_2}), are utilized for performance evaluation. The clock latency range (CLR) is the main criterion for this evaluation. It is a combined factor of both the original clock skew and the voltage uncertainty. CLR is determined by the difference between the maximal and minimal clock skew values under the two given voltage sources:

$$p_{1_{max}}=\max\{d_{s_i}|\forall s_i\in S,V_{dd_1}\} \quad (1)$$

$$p_{1_{min}}=\min\{d_{s_i}|\forall s_i\in S,V_{dd_1}\} \quad (2)$$

$$p_{2_{max}}=\max\{d_{s_i}|\forall s_i\in S,V_{dd_2}\} \quad (3)$$

$$p_{2_{min}}=\min\{d_{s_i}|\forall s_i\in S,V_{dd_2}\} \quad (4)$$

$$CLR=\max\{p_{1_{max}},p_{2_{max}}\}-\min\{p_{1_{min}},p_{2_{min}}\} \quad (5)$$

From the above equations, we can see that CLR, to a certain extent, represents both the clock skew and the voltage variation.

2.3. Resources

The models of an interconnect wire and a buffer are shown in Figs. 1 and 2, respectively. In Fig. 1, the length of the wire is denoted as L. The unit capacitance and unit resistance of the wire are denoted as ρ_C and ρ_R. In Fig. 2, d_b is included to indicate the intrinsic delay of the buffer. C_b denotes the input capacitance and R_b denotes the driver resistance of the buffer. Both wire and buffer will contribute to the total capacitance increment. Clock signals

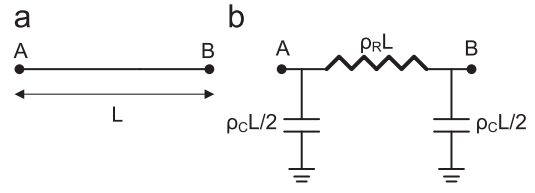

Fig. 1. (a) A segment of wire, (b) π-equivalent circuit model.

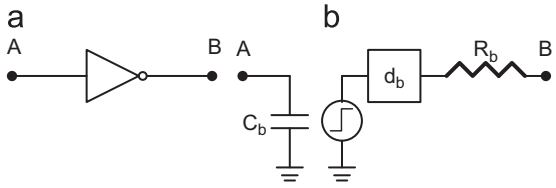

Fig. 2. (a) A buffer, (b) equivalent circuit model.

[1] The approach of dual-MST was presented in Asia and South Pacific Design Automation Conference 2010 [22]. In this paper, clock network synthesizer has been improved by employing the approach of hierarchical buffer and merging point relocation. This paper presents a more complete version of our clock network synthesizer with new and revised methods proposed with improved results. The experimental results of the simulation with corner analysis are newly included and presented.

will be inverted by buffers, so the two clock signals at a merging point should be maintained the same polarity. Meanwhile, Elmore model is utilized for delay computation in our synthesizer, the same as many previous research works applied.

The power consumed by CMOS circuits consists of two components: static and dynamic power. The static power is mostly determined by the feature size and other technology. Therefore, in this paper we only consider dynamic power minimization. The definition of the dynamic power is $P=\frac{1}{2}\alpha C f V_{dd}^2$. C means the total load capacitance on the circuit, f is the frequency of the clock signal and V_{dd} is the power supply. α means the amount of switch times in each clock cycle. For clock tree $\alpha=2$, because there is one rising and one falling edge in each clock period. Since α, f and V_{dd} are constant parameters in the digital circuits, we can use the total load capacitance C as a measure of the power usage. In the following sections, we try to minimize the capacitance on behalf of the power usage reduction. The upper limit of the total capacitance is predefined as an attribute of benchmarks to limit the power consumption.

3. Methodology

In this paper, two novel clock network synthesizers, DMST and DMSTSS, are proposed for clock skew and load capacitance minimization. Both of the two synthesizers are designed in a bottom-up procedure with the application of DME [6,7] technique. DMST is developed based on the construction of dual minimum spanning trees (dual-MST) in geometric matching. Elmore model computation is applied in DMST for delay balancing. DMSTSS is developed based on DMST with the inclusion of delay estimation from SPICE simulation and clock tree tuning.

Original DME method is developed without buffer insertion, and zero skew is achieved by linear delay computation. By applying iterative buffer insertion, we start from two nodes with their accumulated delays and load capacitance in the beginning of the iteration. If these two nodes can be directly merged together without buffer insertion, we compute the segment of merging node by traditional DME technique. Otherwise, we obtain the location segment of the next buffer insertion by computing the intersected segment of the two tilted Manhattan circles which are centered by the two nodes, respectively. The radii of the two circles can be computed from our iterative buffer insertion approach in Section 3.2. As shown in Fig. 5(c), the two radii are $dis(v1,v0)$ and $dis(v2,v0)$, accordingly. As a result, we can combine DME and our iterative buffer insertion together during the nodes merging. Notice that we can only insert buffers outside blockage regions in our work. However, overlapping of buffers are not considered.

DMST is an iterative approach, the general procedure of DMST is shown in Fig. 3(a). At the beginning of the iteration, we have a group of nodes to be merged. This is denoted as V. V can be composed of either clock sinks or internal nodes. A dual-MST based perfect matching technique is applied first to obtain a geometric matching solution upon the node group V. The details of dual-MST is discussed in Section 3.1. During the merging of each matched pair of nodes, a new technique of iterative buffer insertion and wire connection is applied to deal with the buffer distribution problem. The buffer insertion technique and the buffer sizing technique are discussed in Sections 3.2 and 3.3, respectively. If the merging fails because of blockages, this pair is re-merged with an exclusive dual-MZ blockage handling technique. The blockage handling technique is discussed in Section 3.4. After the successful merging of all the nodes in V and the generation of the nodes in V', this procedure is finished. It is then performed iteratively with V replaced by V', until a complete clock tree is constructed.

DMSTSS is a more advanced synthesizer. It is developed based on DMST with real-time SPICE simulation included for delay estimation. The flow of DMSTSS and its relationship with DMST are illustrated in Fig. 3(b). A clock tree is first built up by employing DMST. Hence, the following procedures are repeated based on the number of tree level. Second, SPICE simulation is applied and the whole clock tree is specified with detailed signal delay information. Based on this updated information, the merging nodes of the current tree level are relocated for clock skew minimization. Because of the relocation of the merging nodes, DMST is employed to re-construct the whole tree. The merging nodes of the current tree level are fixed and only the merging nodes of the upper tree level are relocated then. Finally, the clock skew can be fine tuned after several SPICE simulations and relocations of merging nodes. The details of DMSTSS are discussed in Section 3.5. Notice that the synthesis result of DMSTSS is no longer a zero skew tree in terms of the Elmore model, but it has better performance in real circuits work. The details of our techniques are discussed in the following parts.

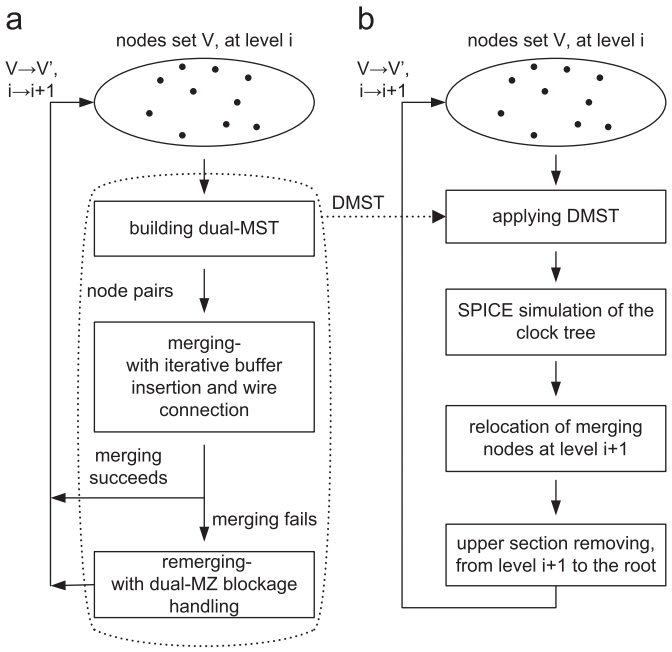

Fig. 3. Design flows of DMST and DMSTSS.

3.1. A dual-MST based geometric perfect matching

Traditional matching algorithms for clock tree synthesis mainly focused on wirelength minimization [1,8]. Our approach focuses on the construction of symmetric topology. In our clock tree, every root-to-leaf path will have similar buffer and wire distribution. This design will make the clock tree more robust towards process variations [27] and less sensitive at process corners [28]. The reason is that the phase delay change due to die-to-die process variations will have the same effect on all the receivers [15]. An example is shown in Fig. 4 to compare the differences between an asymmetric topology and a symmetric topology. $V=\{v_1,\ldots v_4\}$ is a group of sinks. By applying the shortest-path approach in T_1, a clock tree is constructed as Fig. 4(a). v_5 and v_6 are the internal nodes at the first level and v_7 is the root. If the clock tree is constructed to have capacitance symmetric tree branches, the tree is constructed as T_2. In both T_1 and T_2, all the root-to-leaf paths are equidistant, and the total wirelength of T_1 is shorter than that of T_2. T_2 is a capacitance symmetric tree at each tree level, but T_1 is asymmetric at particular tree level. In T_1, the branches in the same level differ, for instance, $dis(v_1,v_5)\neq dis(v_3,v_6)$. While wirelength is not linear to delay, the

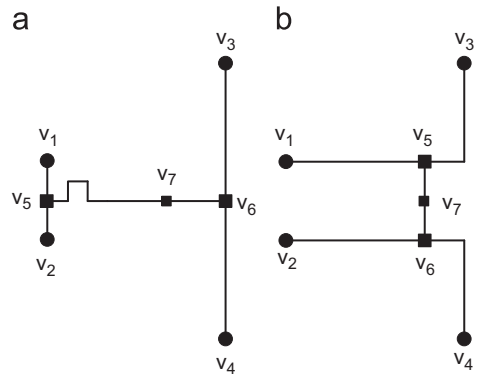

Fig. 4. Comparison of (a) an asymmetric tree and (b) a symmetric tree.

clock skew difference between Elmore delay model and real skew value of T_1 is larger than T_2.It is the reason that the uneven distribution of buffers and wires between the branches of the clock tree may lead to worse CLR and clock skew in most cases with process variations. Thus, T_2 is preferable in our work rather than T_1.

A dual-MST based geometric matching technique is developed for topology generation (a general definition of geometric matching can be found in [3]). It is a weighted perfect matching approach. The Pseudo-code is illustrated in Procedure 1. Given a set of nodes $V=\{v_1,v_2\ldots v_m\}$, we first construct a complete graph $G=\{V,E\}$. Let $|V|$ and $|E|$ denote the number of nodes and edges in the graph G, so $|V|=m$. Since G is a complete graph, every pair of two nodes v_i,v_j is connected by an edge $e_{i,j}$, $E=\{e_{1,2},e_{1,3}\ldots e_{m-1,m}\}$ and $|E|=m(m-1)/2$. The cost of matching two nodes v_i and v_j is denoted as $f_c(e_{i,j})$. In our work we set $f_c(e_{i,j})$ to be the Manhattan distance between v_i and v_j. Let M denote the matching result of G. M is composed of a group of edges and it is a subset of E. The maximal pairing cost of M is denoted as C_{max} and defined as below. We will get close to a symmetric clock tree by reducing C_{max} in each level:

$$C_{max}=\max\{f_c(e_{i,j}):\forall e_{i,j}\in M\} \tag{6}$$

Procedure 1. Partition (G).

Require: $G=\{V,E\}$ is a complete graph, E is sorted in ascending order of $f_c(e_{i,j})$.

```
if |V| ≤ 1 then
  return;
else if |V| = 2 then
  merge (v1,v2);
  return;
else
  Building dual-MST with |V|−2 edges inserted.
  Two subgraphs G′ = {V′,E′} and G″ = {V″,E″} are generated
  Two minimum spanning trees st′ and st″ for V′ and V″ are generated
  if |V′| is odd and |V″| is odd then
    e_{m,n} = arg_{e_{i,j}} min{f_c(e_{i,j}) | ∀n_i ∈ V′, ∀n_j ∈ V″};
    merge (v_m,v_n);
    remove v_m from V′ and v_n from V″;
    remove e_{m,x} from E′, ∀x ∈ V′;
    remove e_{n,y} from E″, ∀y ∈ V″;
  endif
  partition (G′);
  partition (G″);
  return;
end if
```

At the beginning, the pairing cost f_c of every edge of E is computed, sorted in ascending order and stored in memory. Then the matching algorithm based on dual-MST is applied iteratively to generate every matching pair. The input of the approach is a complete graph $G=\{V,E\}$. Similar to the Kruskal's MST algorithm, edges in E are inserted orderly in terms of f_c. During this procedure, no cycle is allowed. When $|V|-2$ edges have been inserted, two disjoint subtrees st' and st'' are generated. Meanwhile, G is divided into two disjoint complete subgraphs $G'=\{V',E'\}$ and $G''=\{V'',E''\}$. Notice that all the nodes of V' are connected by st', and all the nodes of V'' are connected by st'', respectively. It is obvious that st' and st'' are two minimum spanning trees for G' and G'', so dual-MST have been built up. If either $|V'|$ or $|V''|$ is an even number, it goes to the next iteration. Otherwise, an edge subset E_0 will be extracted from E. Every edge $e_{i,j}$ of E_0 has one node v_i from V' and the other node v_j from V''. The edge $e_{m,n}$ of E_0 with the minimum cost value f_c is determined as a matching pair:

$$f_c(e_{m,n})=\min\{f_c(e_{i,j}):\forall e_{i,j}\in E_0\} \tag{7}$$

Notice that this dual-MST based matching approach is a geometric perfect matching. This means $2\times\lfloor|V|/2\rfloor$ nodes will be matched, and $\lceil\log_2|S|\rceil$ levels are performed. Assume that we are at the ith level with the node group V_i. The time complexity of f_c computation equals $O(|V_i|^2)$. On average, the number of partitioning stages is $\lceil\log_2(|V_i|)\rceil$. Thus the time complexity of partitioning is calculated as below:

$$O\left(\sum_{j=1}^{\lceil\log_2(|V_i|)\rceil}\left(\frac{1}{2}\times\frac{|V_i|}{2^{j-1}}\times\left(\frac{|V_i|}{2^{j-1}}-1\right)\times 2^{j-1}\right)\right) \tag{8}$$

Therefore, the total complexity of the ith level is still $O(|V_i|^2)$ on average. While $|V_i|=\lceil|S|/2^{i-1}\rceil$, the time complexity on average for the whole geometric matching is shown below:

$$O\left(\sum_{i=1}^{\lceil\log_2|S|\rceil}\left\lceil\frac{|S|}{2^{i-1}}\right\rceil^2\right)=O(|S|^2) \tag{9}$$

Our approach aims at constructing a symmetric clock tree but it is only a heuristic giving close proximity to the optimal solution. Compared to other geometric matching methods, ours might generate longer wire length but would help to boost the performance of the clock tree. Specific comparison can be found in the section of the experimental results.

3.2. Iterative buffer insertion

In our synthesizer of DMST, the topology is constructed level by level with concurrent buffer insertion and wire connection. This is mainly because accurate subtree information is important at geometric matching in each level. In DMSTSS, we will iteratively remove and rebuild the upper levels of the clock tree on behalf of merging point tuning. Global buffer distribution in the whole clock tree may lead to local cost increment on behalf of other tree sections. However, these benefited tree sections may be removed in the further tuning. Thus, it is improper to distribute the buffers globally along the whole tree. We developed a greedy algorithm of iterative buffer insertion, which is mainly focus on local optimization with skew balancing and load capacitance minimization.

A buffer placement method was proposed in [9] with minimal phase delay achieved. However, it is based on a given topology such that the clock topology should be obtained by some other algorithms. This will affect the performance and flexibility. In addition, clock slew is not considered in this approach, in which buffers is only allowed to be inserted at internal nodes of the clock tree. A similar approach for buffer insertion scheduling is

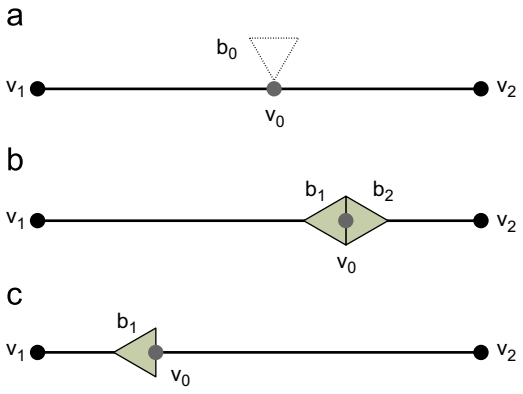

Fig. 5. Design flow of iterative buffer insertion.

proposed in [15]. However, it also enforces a constraint of levelized buffer insertion which restricts the performance. In our work, we developed a simultaneous wire routing, buffer insertion and sizing technique. By applying this technique, the delay and capacitance usage can be reduced. Meanwhile, the global slew constraint is not violated.

An example is illustrated in Fig. 5 to demonstrate our approach for merging two nodes v_1 and v_2. Let $e_{1,2}$ denote the edge connecting the two nodes. Based on their downstream capacitances and subtree delays, the location of the delay balancing point, v_0, can be computed. Let l_{v_1} denote the amount of downstream buffer levels of v_1, and l_{v_2} denote the amount of downstream buffer levels of v_2, respectively. Generally, our method of buffer distribution can be divided into the following two parts.

(1) $l_{v_1} = l_{v_2}$: We assume an upstream buffer b_0 at v_0, as shown in Fig. 5(a). If b_0 can drive v_1 and v_2 directly, the merging of v_1 and v_2 is finished. Otherwise, two back-to-back buffers b_1 and b_2 will be placed in a new delay balancing point along $e_{1,2}$, as shown in Fig. 5(b). The distance between b_1 and b_2 is set to be zero, and they are both located at v_0. Therefore, the current load capacitance left to the upper level is minimized. If no slew violation occurs, the merging is finished. Otherwise, assume there is slew violation along e_{v_1,b_1}. b_2 will be removed and b_1 will be shifted leftwards to its nearest non-slew-violation location. After the insertion of b_1, we enter the next iteration.

(2) $l_{v_1} \neq l_{v_2}$: Assume that $l_{v_1} \leq l_{v_2}$. We will insert a buffer b_1 along $e_{1,2}$, following the rule that b_1 is in the same location of v_0, as shown in Fig. 5(c). Therefore, the current load capacitance left to the upper level is minimized. If no slew violation occurs, we enter the next iteration with the insertion of b_1. Otherwise, there must be slew violation along e_{v_1,b_1}. b_1 will be shifted leftward to its nearest non-slew-violation point. After the insertion of b_1, we enter the next iteration.

Notice that in the next iteration, v_0 replaces v_1 as a input node, and v_2 is remained. Similarly, $e_{0,2}$ replaces $e_{1,2}$. In our discussion of iterative buffer insertion, only straight wire connection is described in details. However, snaking cannot be completely avoided in some particular cases. When the delay difference between the two nodes cannot be solved with straight wire balancing, snaking wire technique can be applied.

3.3. Hierarchical buffer sizing

The slew rate (transition time) of the clock signal should always be observed in order to guarantee the performance of a digital system. Thus, buffer insertion becomes a necessary step in clock tree synthesis. A clock buffer with larger size can provide larger driving power. It gives a smaller slew rate and less clock buffers is needed to be inserted. In addition, the signal delay of a larger buffer is less sensitive to variations. On the other hand, buffers with larger size will consume more power, and larger space is required for placement. Therefore, appropriate buffer sizing is an important issue in balancing the power consumption and variations.

The signal delay of a clock sink is determined by the path from the root to itself. In this paper, it is named by a root-to-leaf signal path. While the insertion of larger buffers is more effective on variation reduction, larger buffers are chosen to be inserted at higher level. It is because these buffers cover a larger number of such root-to-leaf signal paths while the total available capacitance is limited. It means that we can reduce the skew variation with a smaller consumption of buffer capacitance. In addition, the wire capacitance between successive buffers is similar in order to give similar clock slew. Thus, the buffers at the higher level of the tree should be larger than those at the lower level. It is because the load capacitance of a buffer depends on its immediate successive buffers and the similar capacitance produced from wires. As shown in Fig. 6, v_7 at level 2 will cover four downstream clock sinks (v_1,v_2,v_3,v_4) and v_5 at level 1 will cover 2 downstream clock sinks (v_1,v_2). Therefore, we should enlarge the size of the buffers at higher levels to balance the signal delay of more clock sinks. Ideally, the clock buffers can be sized according to the example in Fig. 6. In this example, buffers with larger size will be inserted in upper levels and the smaller buffers will be used in lower level of the clock tree. In this paper, an algorithm is devised to appropriately design the buffer size in descending order from the root to the leaves. Given one type of buffer, we connect a group of buffers in parallel to simulate larger buffer sizes.

Our main target in clock tree synthesis is to minimize the negative effect caused by variation factors. On behalf of this target, every root-to-leaf path should be constructed in similar pattern, with proper buffer location and sizing. Thus, the number of buffer levels in each root-to-leaf path should be exactly the same. Moreover, the size of the buffer insertion in each buffer level should also be the same for all the sinks. Under this pre-condition, the size of the buffers at each insertion point is uniquely determined by its downstream buffer levels. It is a waste of time to devise the size for any specific insertion point during the synthesis. We develop a sizing rule based on some internal attributes of the clock sinks. During the procedure of clock tree synthesis, we compute the proper size for each buffer level in advance and build up a table for buffer sizing reference. Subsequently, all the buffer insertion will follow this sizing table to determine its buffer size.

In order to construct the buffer sizing table, an initial clock tree T_I with the smallest buffer size (single buffer) is constructed first.

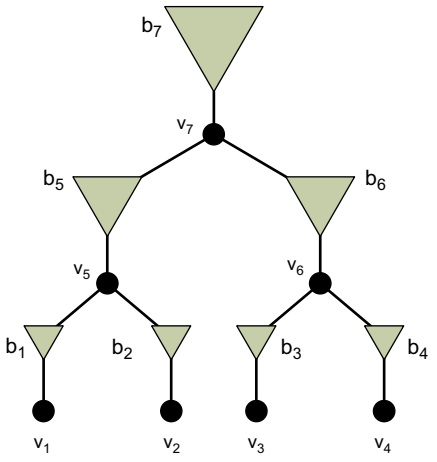

Fig. 6. Example of buffer sizing.

It is constructed by applying the same approach of dual-MST. By computing the difference between the total capacitance of this initial clock tree (*cap*) and the limitation of the total capacitance (*tot_cap*), we can obtain the preliminary information about the available resources for buffer sizing. The buffer size which is the number of parallel buffers of each buffer level is determined accordingly. We define the total amount of different buffer sizes to be an integer m. Let $sz_b(x)$ denote the objective buffer size and x denote the number of downstream buffer levels. According to Fig. 6, $sz_b(x)$ is a monotonically increasing function from the bottom level of the tree. Moreover, we define $L=\{L_0,L_1,\ldots,L_m\}$ to be the number of buffer levels between different buffer sizes, as shown in Fig. 7. When $L_{i-1}\le x<L_i$, $sz_b(x)=i$. L_0 is assumed to be zero, and L_m is assumed to be infinite. In the initial tree T_I, the number of buffer levels between the clock source and the root node of the tree (the first merging node) is denoted as l_{rt}. An example is shown in Fig. 8. The sizes of the buffers inserted at the wire segment from the clock source to the root node of the clock tree are fixed and these should be the buffers with the largest size. The sizes of remaining buffers are determined based on the minimum usage of capacitance and the limitation of total capacitance. The minimum usage of capacitance is contributed by the larger buffers inserted between the clock source and the root node, and the capacitance of the initial clock tree. Thus, if l_{rt} is smaller, the capacitance budget for remaining buffers is larger. Similarly, if *cap* is smaller, the capacitance budget for remaining buffers is also larger. Thus, based on the value of l_{rt}, *cap* and *tot_cap*, the buffer configuration is computed according to the following formula:

$$L_i=\left\lfloor (l_{rt})\times\frac{cap}{tot\_cap}\times\delta_i\right\rfloor+q \tag{10}$$

where *cap* is the used capacitance of the initial clock tree, and *tot_cap* is the total capacitance allowed. δ_i is a parameter that is related to the driving power of i parallel buffers. q is an offset parameter for the equation. Here i means the amount of parallel buffers and $i=1,2,\ldots,m-1$.

The size of buffers cannot be arbitrary in the real cases, and we connect a number of buffers in parallel to enlarge the size. Notice that we do not use node levels but downstream buffer levels to determine the buffer sizes. Let l_{v_i} denote the amount of downstream buffer levels at an internal node v_i. The amounts of buffer levels of its children nodes are denoted by $l^L_{v_i}$ and $l^R_{v_i}$. Hence, the value of l_{v_i} is determined by the following formula:

$$l_{v_i}=\max\{l^L_{v_i},l^R_{v_i}\} \tag{11}$$

$l_{v_i}=0$ if v_i is a sink node. Since it is a bottom-up method, the corresponding buffer levels of the internal nodes can be computed in this way.

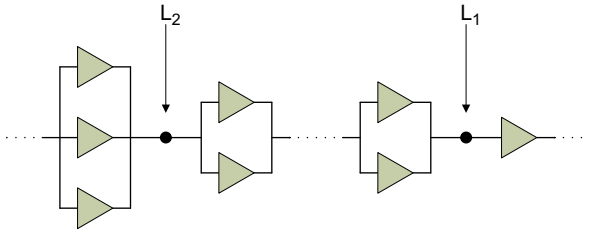

Fig. 7. A clock tree divided by buffer levels L_1 and L_2.

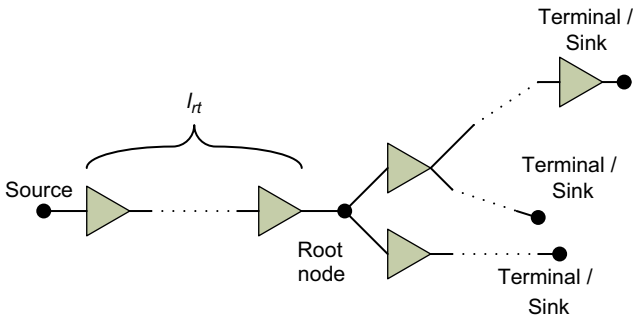

Fig. 8. Definition of l_{rt} and l_{bf}.

3.4. Dual-MZ blockage handling technique

Because of the existence of macro-blocks, there may be pre-assigned obstacles on the chip area. Wires can cover these blockage areas freely, but buffers will cause violation. The connection is thus constrained. Traditionally, clock tree synthesizer can be divided into two steps. They do clock routing first to connect all the sinks to the signal source. Subsequently, buffer sizing, insertion and wire sizing are determined. In case of buffer violation in the second step, clock routing has to detour out of the blockage areas in the first step. An example of possible result of traditional synthesis is shown in Fig. 10(a). Our work is based on a similar approach in [11]. However, we improve it by the application of two rotational maze routers for clock subtree merging. It is fairly effective in delay balancing as well as capacitance reduction.

Instead of above mentioned approach, our approach considers concurrent buffer sizing and insertion during the procedure of clock routing. A novel blockage handling method, dual-MZ, is developed based on the traditional maze routing technique. Several enhancements are developed to make it satisfy the clock network synthesis constraints. We first decompose the chip into a $M\times N$ grid. Assume that v_1 and v_2 are the two nodes to be merged. Two independent maze routers mz_1 and mz_2 are initialized for v_1 and v_2, exclusively. Therefore, a dual-MZ is constructed. The priority queue of each maze router is sorted by the downstream delay values of its elements in ascending orders. Let dly_1 and dly_2 denote the minimum downstream delays of mz_1 and mz_2. Without loss of generality, assume that $dly_1\le dly_2$, and mz_1 is selected to be the primary router. Like ordinary maze routing, it will search the four adjacent grid points in one iteration. The downstream delay of mz_1 is then increased, so dly_1 will be updated. When dly_1 becomes bigger than dly_2, mz_2 will replace mz_1 to be the primary router and continue the routing job. In this way, mz_1 and mz_2 are executed in turns, and the values of dly_1 and dly_2 stay close to each other. A design flow of dual-MZ is shown in Fig. 9. When the two routers meet at the end, their accumulated delays will not differ a lot. The dual-MZ can result in less buffer insertions and wire detours, so the power consumption is reduced. Notice that during this procedure, wire connection together with buffer insertion is concurrently applied along the propagation trace of the two routers, and they both contribute to the delay accumulation for comparison. A possible synthesis result of dual-MZ is shown in Fig. 10(b).

Our dual-MZ can save resources with less buffers and wires cost involved. Notice that the size of the grid graph ($M\times N$) is manually determined in our algorithm. M and N can be scaled in order to derive a tradeoff between routing complexity and routing quality. In our experiments, both M and N are set to be 1000. A synthesis result of the benchmark *ispd09fnb1* is shown in Fig. 11 (the detailed information of *ispd09fnb1* can be found in Table 2). The green stars denote the clock sinks, the blue squares denote the buffers, the purple circle denotes the signal source, the red lines denote the wire connection and the gray rectangles denote the blockage areas. We can see that by applying dual-MZ, clock wire can cover the blockage area freely with proper buffer insertion outside. Resources cost can be reduced by our method efficiently.

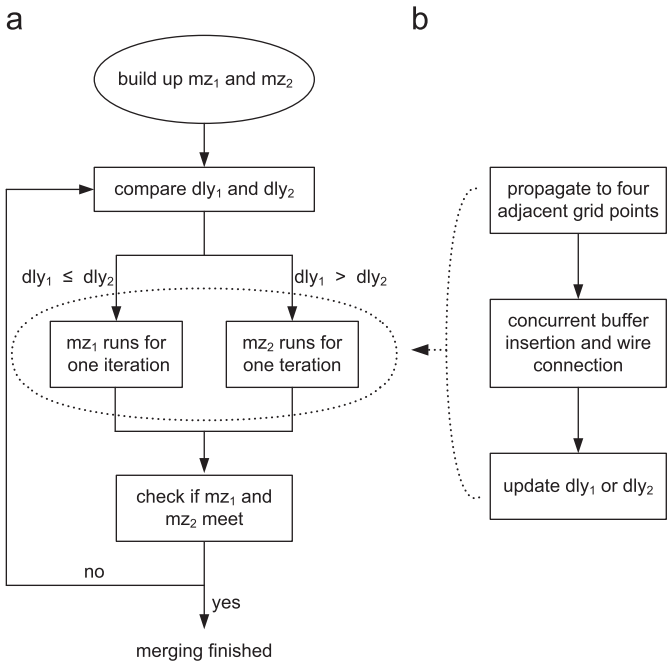

Fig. 9. Design flow of (a) dual-MZ and (b) specific maze routing.

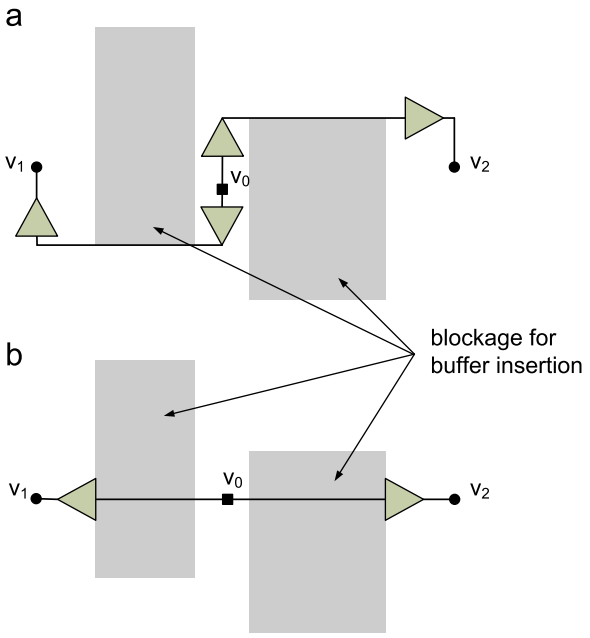

Fig. 10. Possible synthesis results of (a) complete detour and (b) dual-MZ.

3.5. Merging point relocation with SPICE simulation

DMSTSS is an advanced version of DMST with SPICE tuning involved. Given a clock tree, SPICE simulation can provide accurate delay estimation at any point. Based on this additional information, clock tree can be adjusted to improve the deviation caused by Elmore model. Nevertheless, despite improvement on clock skew, the run-time cost of SPICE simulation is quite large. Thus we need to reach a tradeoff between performance and efficiency.

Our merging point relocation is a level-by-level approach. Here the level means not buffer level but the node level in the tree structure. The specific flow is shown in Fig. 3(b). After DMST, a clock tree is available. SPICE simulation of the tree is applied to get the delay estimation. Based on this accurate estimation, all the internal nodes in the current level will be relocated simultaneously. Then the tree sections at upper levels are removed and rebuilt by DMST, based on the set of nodes at the current level. Because of this iterative removing and rebuilding, slew violation caused by merging point relocation can be avoided. Meanwhile, the upper topology of the tree is dynamically improved. DMSTSS is applied in this iterative way, it ends when the root is reached.

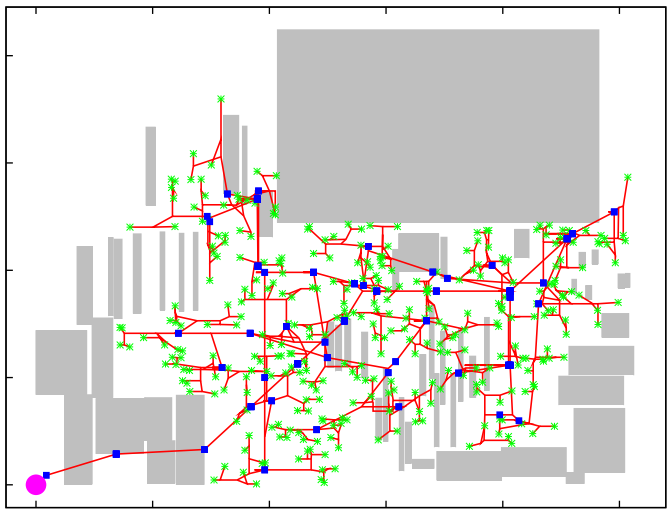

Fig. 11. DMSTSS synthesis result of the benchmark *ispd*09*fnb*1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The relocation of each merging point is based on Elmore delay computation of wire connections. No buffer computation is involved in this tuning approach. Assume that v_0 is the node to be relocated, and v_1 and v_2 are its two children nodes. $e_{0,1}$ and $e_{0,2}$ are the two according edges. We first update the delay values of the nodes with SPICE estimation result. Then a bidirectional search is applied from v_0 to v_1 and v_2, along $e_{0,1}$ and $e_{0,2}$. This search stops when it meets the first buffer on the edge or it meets the ending node. Assume the two stops of the bidirectional search to be n_1 and n_2 on $e_{0,1}$ and $e_{0,2}$, respectively. Then the two wire connections e_{v_0,n_1} and e_{v_0,n_2} are removed and remerged. Notice that here Elmore model is still used in delay computation, but the specific delay values are from SPICE estimation.

3.6. Slew table construction

In our work, slew rate constraint (≤ 100 ps) is engaged along the whole clock tree. Real-time slew tuning will cost a lot of time and decrease the efficiency of the program. We pre-build a look up table for slew reference during the execution of our program. The procedure to construct the slew table can be divided into two categories, single wire and binary branch. At any node, we need to get the maximum driving length from the slew table, with upstream buffer size, downstream buffer size and downstream wirelength as the indexes. This is shown in Fig. 12. We can assume that the current node to be v_0, and l_0 is the maximum driving length for the next buffer insertion b_0. l_0 is what we need to get from the slew table in the program. As shown in Fig. 7, the buffer size sz_{b_0} for the next level can be derived from our buffer sizing table, so it is a know value. At the single wire in Fig. 12(a), downstream information of sz_{b_1} and l_1 are ready to utilize. Therefore, with three available indexes of sz_{b_0}, sz_{b_1} and l_1, we can get l_0 from the slew table. We can get l_0 for a binary branch in the same way, as shown in Fig. 12(b). The only difference is that there are two additional indexes required for the

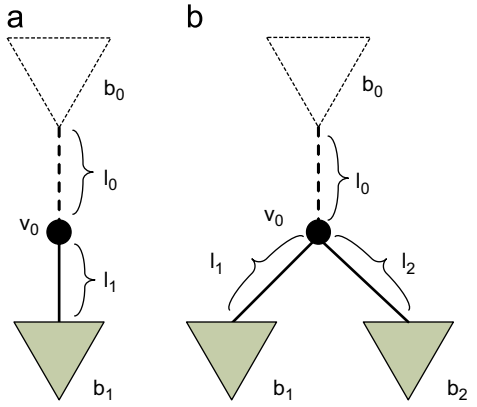

Fig. 12. Driving length reference at (a) single wire and (b) binary branch.

slew table, which are sz_{b_2} and l_2 as the downstream buffer size and wire length of the other branch.

As stated in Section 3.3, buffer size are defined by the number of parallel buffer connection in our work. Thus sz_{b_0}, sz_{b_1} and sz_{b_2} are discrete values (integers actually) and they can be used as table indexes directly. However, wire length is a real continuous value in nanometer scale. Therefore, we quantize the wire length of l_0, l_1 and l_2 during the slew table construction in order to maintain the table size within a proper range. During the clock tree synthesis, a linear interpolation program is applied to transfer the actual wire length into discrete table index.

4. Experimental results

In this section, we present our experimental results are presented. We implement our clock tree synthesizer in the C language and the program is executed on the Linux operating system with an Intel Core2 Quad 2.4 GHz CPU and 4GB memory. In our experiment, one type of wire and one type of buffer are used in our clock tree synthesizer. The unit resistance of the wire is 0.0003 Ω/nm and the unit capacitance of the wire is 0.00016 fF/nm. The specific configuration of the buffer in different sizes is shown in Table 1. In our synthesizer, the maximum buffer size is set to be 6, which is inserted on the connection path between the source and the root. Hence, we list the attributes of different buffers while the sizes of them are less than or equal to 6. This table is generated based on the statistic of our SPICE simulation. C_b denotes the input capacitance, R_b denotes the driver resistance and d_b denotes the internal delay of a buffer, respectively. During the evaluation, two power supplies with $V_{dd_1} = 1.0$ V and $V_{dd_2} = 1.2$ V are employed independently on behalf of the simulation of voltage variation and CLR computation.

Two benchmarks suites are included in our experiment. The first one is released from the ISPD 2009 contest [19] and the second one is released from the ISPD 2010 contest [29]. Detailed information of the benchmark circuits are shown in Tables 2 and 3, respectively.

We implement the techniques of two widely used matching methods MMM [1] and CL [8] in our synthesizer to replace DMST (dual-MST), with the other parts of our CNS unchanged. In CL, the cluster size is maintained to be $\frac{2}{3}$ of the group of nodes (in [8], the ratio is between $(\frac{1}{2},1)$ so $\frac{2}{3}$ is a proper setting). The average result in Table 4 is obtained from execution on the benchmarks in Table 2, and SPICE simulation is involved in every case. The performance comparison between the techniques is illustrated in Table 4. The CLR (ps), nominal clock skew (ps), total capacitance (pF), total wirelength (mm) and CPU time (s) are shown in that table, and the results are the average value of all benchmarks.Compared to the approach of MMM, the approach of CL can reduce the total wirelength of the clock network in sacrifice of the clock skew. Our DMST approach has a longer total wirelength compared to the approach of CL. However, DMST needs a smaller total capacitance because less number of buffers are required for delay balancing. In addition, the uneven distribution of buffers and wires between the branches of the clock tree may lead to worse CLR in most cases. Thus, our matching technique, DMST, can outperform MMM and CL in clock skew, CLR, total capacitance and CPU time.

A summary of the performance of different clock tree synthesizers is shown in Table 6. CLR (ps), capacitance percentage (%) and CPU run-time (s) are listed for performance comparison.

Table 1
Buffer configuration.

Buffer sizes	C_b (fF)	R_b (Ω)	d_b (ps)
1	35	66.9	4.92
2	70	40.5	5.63
3	105	31.3	6.13
4	140	26.4	6.52
5	175	25.0	6.95
6	210	20.7	7.20

Table 2
Circuit information of the benchmarks from ISPD 2009.

Circuits	Chip size (mm^2)	No. of sinks	No. of blockages (Area, %)	CAP limit (pF)
f11	121	121	0 (0)	118
f12	102	117	0 (0)	110
f21	147	117	0 (0)	125
f22	57	91	0 (0)	80
f31	292	273	88 (24)	250
f32	289	190	99 (34)	190
fnb1	5	330	53 (38)	42
f33	234	209	80 (28)	195
f34	256	157	99 (39)	160
f35	234	193	96 (33)	185
fnb2	28	440	1346 (64)	88
Avg.	140	203	169 (24)	140.3

Table 3
Circuit information of the benchmarks from ISPD 2010.

Circuits	Chip size (mm^2)	No. of sinks	No. of blockages
cns01	64	1107	4
cns02	91	2249	1
cns03	1.5	1200	2
cns04	5.7	1845	2
cns05	5.9	1016	1
cns06	1.7	981	0
cns07	3.7	1915	0
cns08	3.7	1134	0

Table 4
Comparison between different matching methods.

Metric	MMM	CL	DMST
CLR (ps)	13.35	13.79	12.25
Skew (ps)	8.75	9.88	7.72
Capacity (pF)	142.5	123.2	117.6
Wirelength (mm)	283.0	223.0	239.6
CPU time (s)	448	650	317

Table 5
Comparison of the computing platforms.

Synthesizers	CPU	MEM
DMSTSS	Intel Core 2 2.4 GHz	4 GB
NTU	Intel Xeon 2.0 GHz	16 GB
NCTU	Intel Xeon 3.0 GHz	32 GB
ISPD09	AMD Opteron 2.8 GHz	128 GB

Table 6
Comparison among DMSTSS and the three synthesizers in ASP-DAC 2010 and the best result in ISPD 2009.

Cases	DMSTSS			NTU [20]		NCTU [21]		ISPD09		
	CLR	C%	CPU	CLR	CPU	CLR	CPU	CLR	C%	CPU
f11	10.1	87.4	208	19.7	4639	18.8	2200	22.3	89.9	23 358
f12	8.8	93.2	227	17.5	4231	15.5	1923	22.2	87.9	14 992
f21	9.7	93.9	258	19.9	4629	17.0	2231	19.6	86.7	26 420
f22	6.9	89.2	136	16.5	3937	16.3	1370	16.4	85.0	9432
f31	11.0	83.4	727	31.1	11 112	22.6	6360	45.1	73.5	40 088
f32	9.9	85.2	485	23.0	7293	20.6	3967	18.4	89.9	2888
nb1	12.1	91.6	95	15.7	3719	14.3	1743	19.8	63.1	477
Avg.	9.8	89.8	305	20.5	5651	17.9	2828	23.4	82.3	16 807
f33	13.6	88.8	531	NA	NA	NA	NA	NA	NA	NA
f34	9.9	91.0	385	NA	NA	NA	NA	NA	NA	NA
f35	11.3	90.8	507	NA	NA	NA	NA	NA	NA	NA
fnb2	13.1	87.9	202	NA	NA	NA	NA	NA	NA	NA

Notice that the capacitance is the sum of the wire capacitance and the buffer capacitance of the clock tree. It means that the effect on wirelength and buffer area can also be illustrated with the "capacitance percentage". DMSTSS is our proposed clock tree synthesizer in this paper. As discussed in Section 3, it is an extended work of DMST applying real-time SPICE simulation for clock tree tuning. We compare our results with two clock network synthesizers published in ASP-DAC 2010, NTU [20],[2] and NCTU [21], (see footnote 2.) and the best result with the smallest CLR of each circuit published in the ISPD 2009 contest (ISPD09). The computing platforms of the above synthesizers are shown in Table 5.

According to the results in Table 6, we can see that the CLR obtained from DMSTSS is outstanding. Generally, the average CLR of DMSTSS is only 47.7% of NTU, 54.6% of NCTU and 41.8% of ISPD09. Meanwhile, the average CPU time of our approach is only 5.4% of NTU, 10.8% of NCTU and 1.8% of ISPD09. CLR, as a combined evaluation index of both clock skew and voltage variation, is the first criterion in comparison. It can be seen that our DMSTSS has great advantage on CLR compared to NTU, NCTU and ISPD09. Meanwhile, our synthesizer DMSTSS is much faster than the other published works.

A summary of the performance of our clock tree synthesizer with fixed buffer sizes is shown in Table 7. No buffer sizing technique is applied here. Notice that N_{bf} is used to define the buffer size of the buffers. CLR, capacitance percentage and CPU time are listed. N_{Bf} is the number of buffers used in every buffer insertion along the whole clock tree. We run our program with DMSTSS. While the buffer size is smaller, the CLR is larger and less capacitance is used. While the buffer size is larger, the CLR is smaller and the total capacitance is increased significantly. We can observe that buffer insertion with fixed size is not a good idea

[2] The publications of NTU and NCTU did not present the total capacitance usage of their research works.

Table 7
Comparison of CLR and capacitance for fixed buffer sizes.

Circuits	DMSTSS					
	Smaller buffer ($N_{Bf}=1$)			Larger buffer ($N_{Bf}=3$)		
	CLR	C%	CPU	CLR	C%	CPU
f11	57.6	49.9	73	23.5	105.1	320
f12	51.4	47.8	64	17.8	98.8	279
f21	58.1	48.0	76	20.6	105.9	341
f22	40.8	46.9	43	15.8	96.4	168
f31	81.0	50.7	295	24.3	107.6	1545
f32	80.3	51.1	163	23.7	110.9	910
fnb1	19.3	52.3	38	14.8	109.8	135
f33	73.6	50.8	225	21.4	102.5	946
f34	74.3	51.2	128	21.4	105.6	606
f35	84.5	50.1	236	24.9	109.6	1000
fnb2	32.1	54.2	95	16.5	123.5	355
Avg.	59.4	50.3	130	20.4	106.8	601

Table 8
Variations of the transistor.

Parameters	NMOS			PMOS		
	Typical	Slow	Fast	Typical	Slow	Fast
$l_{int}(10^{-9})$	3.750	2.875	4.625	3.75	2.875	4.625
v_{th0}	0.404	0.431	0.377	−0.384	−0.411	−0.355
k_1	0.400	0.420	0.379	0.400	0.422	0.378
u_0	0.054	0.051	0.057	0.020	0.018	0.022
$x_j(10^{-8})$	1.40	1.54	1.26	1.40	1.54	1.26

in clock tree synthesis, and there is a tradeoff between load capacitance and CLR. Compared to the results to DMSTSS shown in Table 6, we can see that the capacitance can be utilized more efficiently with significant CLR reduction by using our buffer sizing technique discussed in Section 3.3.

CLR is the metric used in clock network synthesis contest. However, CLR has not yet been applied in industry and it is not the most appropriate approach to verify the performance of clock trees with process variation. We apply corner analysis in simulation to further verify the performance of our clock tree with variations. We simulate over multiple corners of process, power supply, and temperature by applying two sets of parameters (Fast–Fast (FF) and Slow–Slow (SS)) on the typical transistor. Notice that the 45 nm technology node is applied. The details of the parameters for the variations of the transistor is shown in Table 8. Fast–Fast (FF) means that the delay of both the PMOS and NMOS transistors will be smaller and Slow-Slow means that the delay of both the PMOS and NMOS transistors will be greater.

We compare our results with the clock trees of the top two teams in the ISPD 2009 contest. The clock tree solutions of each circuit from these two teams are obtained from the contest. Corner analysis is applied to all clock tree solutions. The results including the worst clock skew of corner analysis, capacitance utilization and run-time are recorded and the results are shown in Table 9. We can see that the result of our DMSTSS is competitive with the best result from the contest. DMSTSS has a smaller skew in some circuits and has a larger skew in others. Although the clock skews obtained from the clock tree solution of Team 4 is slightly better in four circuits, the run-time of the corresponding clock tree construction is always very slow. It is because it applies SPICE simulation comprehensively to reduce the clock skew under normal condition by relocating each merging node of the clock tree iteratively. In general, the skew performance of DMSTSS

Table 9
Comparison of corner analysis simulation among DMSTSS and the synthesizers in ISPD 2009.

Circuits	DMSTSS			Team 4 (ISPD 2009)			Team 6 (ISPD 2009)		
	Skew (ps)	CAP (nF)	CPU (s)	Skew (ps)	CAP (nF)	CPU (s)	Skew (ps)	CAP (nF)	CPU (s)
f11	80.48	103 153.59	208.00	71.88	100 921.96	14 764	83.56	87 156.96	3892
f12	73.42	102 528.98	227.00	72.92	93 189.21	13 934	79.61	80 796.82	3944
f21	85.82	117 430.41	258.00	81.74	100 727.50	14 978	90.78	92 880.35	4578
f22	57.28	71 373.93	136.00	63.19	65 457.94	7189	67.71	56 006.58	2005
f31	126.13	213 002.59	727.00	139.62	183 735.03	40 088	145.63	203 827.34	17 333
f32	115.19	165 195.73	485.00	107.50	152 275.13	3565	120.44	147 037.93	10 599
fnb1	36.44	36 832.34	204.00	Nil	Nil	Nil	32.37	26 501.15	4023

Table 10
Corner analysis simulation of DMSTSS with circuits from ISPD 2010.

Circuits	DMSTSS		
	Skew (ps)	CAP (nF)	CPU (s)
cns01	75.70	228 456	2074.57
cns02	88.70	432 443	6255.68
cns03	46.56	106 628	866.04
cns04	41.77	127 876	1142.16
cns05	41.06	67 894	462.81
cns06	70.87	115 367	867.31
cns07	46.45	141 154	1365.92
cns08	45.96	98 271	746.51

under variation condition is competitive with the winner of the ISPD contest and the run-time of our DMSTSS is significantly faster.

Our DMSTSS synthesizer has good performance on the ISPD 2009 benchmarks shown in Tables 2 and 9. However, the average amount of clock sinks is 203 only and the largest size of the benchmarks only has 440 sinks. These benchmarks are too small and they are less representative in the modern VLSI technology. We also test our DMSTSS with ISPD 2010 benchmarks which are shown in Table 3. The experimental result is shown in Table 10. It shows that our DMSTSS can handle large circuits efficiently. According to corner analysis simulation, the global clock skew is between 40 ps and 88 ps.

5. Conclusion

In conclusion, two clock tree synthesizers, DMST and DMSTSS, have been proposed. Several novel methods, including dual-MST perfect matching, buffer insertion and sizing, dual-MZ blockage handling, clock tree tuning with SPICE simulation and slew table construction are developed to solve the clock network synthesis problem. Our dual-MST based geometric matching method can construct a clock tree of symmetric structure with increased tolerance towards process variation. Our buffer sizing technique is able to utilize the capacitance more efficiently in order to reduce the negative effect of variations. Iterative buffer insertion and dual-MZ blockage handling can solve the problem on connection of a merging pair with delay balancing and saving on resources usage. Internal nodes relocation further tunes the clock tree and reduces the clock skew. Meanwhile, the slew table provides driving length of each buffer insertion during the program execution in order to satisfy the slew rate constraint. Experimental results show that our improved synthesizer has good performance and high efficiency. From the comparison in Table 6 with the previous works based on the ISPD 2009 benchmark suite, we can see our synthesizer with the best performance on CLR and run-time. Based on the promising results in Tables 9 and 10, it consistently shows good performance and robustness of our DMSTSS synthesizer on larger circuits. The result from simulation of corner analysis shows that our topology is more tolerant on the process variations.

References

[1] M.A.B. Jackson, A. Srinivasan, E.S. Kuh, Clock routing for high-performance ICs, in: Proceedings of IEEE/ACM Design Automation Conference, 1990, pp. 573–579.
[2] M. Edahiro, T. Yoshimura, Minimum path-length equi-distant routing, in: Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems, 1992, pp. 41–46.
[3] A. Kahng, J. Cong, G. Robins, High-performance clock routing based on recursive geometric matching, in: Proceedings of IEEE/ACM Design Automation Conference, 1991, pp. 322–327.
[4] W.C. Elmore, The transient response of damped linear networks with particular regard to wide band amplifiers, Journal of Applied Physics 19 (1) (1948) 55–63.
[5] R.-S. Tsay, Exact zero skew, in: Proceedings of IEEE/ACM International Conference on Computer Aided Design, 1991, pp. 336–339.
[6] K.D. Boese, A.B. Kahng, Zero-skew clock routing trees with minimum wirelength, in: Proceedings of 5th the Annual IEEE International ASIC Conference and Exhibition, 1992, pp. 17–21.
[7] T.H. Chao, Y.C. Hsu, J.M. Ho, A.B. Kahng, Zero skew clock routing with minimum wirelength, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 39 (11) (1992) 799–814.
[8] M. Edahiro, A clustering-based optimization algorithm in zero-skew routings, in: Proceedings of IEEE/ACM Design Automation Conference, 1993, pp. 612–616.
[9] L.P.P.P. van Ginneken, Buffer placement in distributed RC-tree networks for minimal Elmore delay, in: Proceedings of International Symposium on Circuits and Systems, 1990, pp. 865–868.
[10] W. Shi, Z. Li, A fast algorithm for optimal buffer insertion, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24 (6) (2005) 879–891.
[11] H. Zhou, D.F. Wong, I.-M. Liu, A. Aziz, Simultaneous routing and buffer insertion with restrictions on buffer locations, in: Proceedings of IEEE/ACM Design Automation Conference, 1999, pp. 96–99.
[12] S. Natarajan, S.L. Sam, D. Boning, A. Chandrakasan, R. Vallishayee, S. Nassif, A methodology for modeling the effects of systematic within-die interconnect and device variation on circuit performance, in: Proceedings of IEEE/ACM Design Automation Conference, 2000, pp. 172–175.
[13] R. Saleh, S.Z. Hussain, S. Rochel, D. Overhauser, Clock skew verification in the presence of IR-Drop in the power distribution network, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 19 (6) (2000) 635–644.
[14] S. Sauter, D. Schmitt-Landsiedel, R. Thewes, W. Webber, Effect of parameter variations at chip and wafer level on clock skews, IEEE Transactions on Semiconductor Manufacturing 13 (4) (2000) 395–400.
[15] I.-M. Liu, T.-L. Chou, D.F. Wong, A. Aziz, Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion, in: Proceedings of IEEE/ACM International Conference on Computer Aided Design, 2000, pp. 33–38.
[16] Y.P. Chen, D.F. Wong, An algorithm for zero-skew clock tree routing with buffer insertion, in: Proceedings of European Design and Test Conference, 1996, pp. 230–236.
[17] A. Rajaram, J. Hu, R. Mahapatra, Reducing clock skew variability via cross-links, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25 (6) (2006) 1176–1182.
[18] A. Rajaram, D.Z. Pan, Variation tolerant buffered clock network synthesis with cross links, in: Proceedings of ACM International Symposium on Physical Design, 2006, pp. 157–164.
[19] C.N. Sze, P. Restle, G.-J. Nam, C. Alpert, ISPD2009 clock network synthesis contest, in: Proceedings of ACM International Symposium on Physical Design, 2009, pp. 149–150.

[20] X.W. Shih, C.C. Cheng, Y.K. Ho, Y.W. Chang, Blockage-avoiding buffered clock-tree synthesis for clock latency-range and skew minimization, in: Proceedings of Asia and South Pacific Design Automation Conference, 2010, pp. 395–400.
[21] W.H. Liu, Y.L. Li, H.C. Chen, Minimizing clock latency range in robust clock tree synthesis, in: Proceedings of Asia and South Pacific Design Automation Conference, 2010, pp. 389–394.
[22] J. Lu, W.K. Chow, C.W. Sham, E.F.Y. Young, A dual-MST approach for clock network synthesis, in: Proceedings of Asia and South Pacific Design Automation Conference, 2010, pp. 467–473.
[23] C.M. Chang, S.H. Huang, Y.K. Ho, J.Z. Lin, H.P. Wang, Y.S. Lu, Type-matching clock tree for zero skew clock gating, in: Proceedings of IEEE/ACM Design Automation Conference, 2008, pp. 714–719.
[24] M. Donno, A. Ivaldi, L. Benini, E. Macii, Clock-tree power optimization based on RTL clock-gating, in: Proceedings of IEEE/ACM Design Automation Conference, 2003, pp. 622–627.
[25] A.H. Farrahi, C. Chen, A. Srivastava, G. Tellez, M. Sarrafzadeh, Activity-driven clock design, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 20 (6) (2001) 705–714.
[26] J. Oh, M. Pedram, Gated clock routing for low-power microprocessor design, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 20 (6) (2001) 715–722.
[27] J.G. Xi, W.W.-M. Dai, Buffer insertion and sizing under process variations for low power clock distribution, in: Proceedings of IEEE/ACM Design Automation Conference, 1995, pp. 383–388.
[28] A. Rajaram, D.Z. Pan, Robust chip-level clock tree synthesis for SOC designs, in: Proceedings of IEEE/ACM Design Automation Conference, 2008, pp. 720–723.
[29] C.N. Sze, ISPD 2010 high performance clock network synthesis contest: benchmark suite and results, in: Proceedings of ACM International Symposium on Physical Design, 2010, pp. 149–150.

Jingwei Lu received the Bachelor degree from Zhejiang University and M.Phil. degree from The Hong Kong Polytechnic University in 2008 and 2010, respectively. He is now a Ph.D. candidate from the University of California, San Diego. His research interests are design automation of VLSI including placement and clock planning.

Wing-Kai Chow received the Bachelor degree from The Hong Kong Polytechnic University in 2009. He has worked as a research assistant in the same university in 2010. He is now working as a research assistant in The Chinese University of Hong Kong. His research interests are design automation of VLSI such as routing and clock planning.

Chiu-Wing Sham received the Bachelor degree (Computer Engineering) and M.Phil. degree from The Chinese University of Hong Kong in 2000 and 2002, respectively, and received the Ph.D. degree from the same university in 2006. He has worked as a research engineer in Synopsys (Shanghai) and as an electronic engineer on the FPGA applications of motion control system in ASM (Hong Kong). He is working as a lecturer in The Hong Kong Polytechnic University since 2006. His research interests are design automation of VLSI such as floorplanning, placement and clock planning, and digital design.