A Dual-MST Approach for Clock Network Synthesis

Jingwei Lu
Electronic and Information Engineering
The Hong Kong Polytechnic University
francesco.ljw@gmail.com

Wing-Kai Chow
Electronic and Information Engineering
The Hong Kong Polytechnic University
williamchowhk@gmail.com

Chiu-Wing Sham
Electronic and Information Engineering
The Hong Kong Polytechnic University
encwsham@polyu.edu.hk

Evangeline F.Y. Young
Computer Science and Engineering
The Chinese University of Hong Kong
fyyoung@cse.cuhk.edu.hk

Abstract—In nanometer-scale VLSI physical design, clock network becomes a major concern on determining the total performance of digital circuit. Clock skew and PVT (Process, Voltage and Temperature) variations contribute a lot to its behavior. Previous works mainly focused on skew and wirelength minimization. It may lead to negative influence towards these process variation factors. In this paper, a novel clock network synthesizer is proposed and several algorithms are introduced for performance improvement. A dual-MST (DMST) geometric matching approach is proposed for topology construction. It can help balancing the tree structure to reduce the variation effect. A recursive buffer insertion technique and a blockage handling method are also presented, and they are developed for proper distribution of buffers and saving of capacitance. Experimental results show that our matching approach is better than the traditional methods, and in particular our synthesizer has better performance compared to the results of the winner in the ISPD 2009 contest.

I. Introduction

In the procedure of VLSI design, the distribution and placement of clock nets play an important role. With the continuous reduction of transistor size, the signal delay of interconnection becomes a dominant factor compared with internal delays of logic cells and macro cells. In modern synchronous digital devices, system performance is greatly determined by circuit phase delay, clock skew and PVT variations. It is thus necessary to pay more attention to clock network synthesis (CNS) from the very beginning of the physical design stage.

Clock skew represents the timing variation of different terminals in the clock net. In order to synchronize one circuit, each terminal or sink must be reached within a specified small time range. Otherwise, the extent of synchronization would become unacceptable. There were plenty of researches focusing on clock skew minimization. Some earlier proposed works concentrated on the distribution of wirelengths between source and terminals to achieve delay equalization. Jackson [1] firstly presented a clock routing algorithm based on a sub-optimal equidistant network with recursive horizontal-vertical partitioning. Later, more improvements were made to reach exact equidistant tree [2] for clock net, and a pathlength skew balancing approach is proposed in [3]. Afterwards, delay balancing using Elmore delay model [4] became prevalent to acquire more accurate information on time delay. Clock network with exact zero skew [5] is proposed by applying balancing method based on the Elmore model. Chao [6] used segment instead of point to represent the set of best merging locations and deferred embedding is applied to reduce the total wirelength. Edahiro mentioned in his work [7] that unbalanced trees would have shorter total wirelength. To supply sufficient driving power, buffer insertion is also commonly involved, especially in multi-level networks [8, 9].

Recently, process variation becomes a focus of attention besides the skew and clock delay problems. It is mainly due to the uncertainty of various factors inside a circuit, such as process [10], voltage and temperature [11]. In order to keep a chip stable and functioning well, methods with better adaptability are widely favored. Many researchers focused on robust algorithms for variation minimization. Techniques such as wire sizing [12], buffer sizing [13] and link insertion [14, 15] are applied. Other researchers proposed related works on chip-level synthesis [16] and logic gates matching [17], etc. In ISPD 2009 [18], a CNS contest was held in which a process variation related objective, called Clock Latency Range (CLR), was formulated and several benchmarks were released, and this brought in more research interests on this CNS problem and increased the comparability of different works on the variation factor.

In this paper, we propose a new clock network synthesizer. Some heuristics are proposed to optimize CLR as well as clock skew and wirelength. It contains three main features: (1) An approach to construct dual-MST (DMST) for geometric matching. (2) A recursive buffer insertion technique for driving power afford and capacitance reduction. (3) A blockage handler to deal with buffer location violation.

The remainder of this paper is organized as follows. Section II includes the problem formulation. Section III is composed of our three main contributions, topology construction, buffer insertion and blockage handling. In section IV, we present our results based on the ISPD09 benchmarks. Finally, we give our conclusion in section V.

II. Problem Formulation

VLSI physical design can be basically divided into two stages: placement and routing. After placement, all the clock pins are fixed, including the source and a set of terminals. Because of the importance of clock synchronization, CNS is usually performed before routing other nets. Hence, wire resources are fully available for the clock nets to utilize all over the cir-

978-1-4244-5766-3/10/$26.00 ©2010 IEEE

Notation	Description
C_s	clock skew of a clock network
S	a set of sinks
s_i	the ith sink in S
d_{s_i}	delay of sink s_i
D_b	buffer delay
D_w	wire delay
C_d	downstream capacitance
C_b	input capacitance of a buffer
d_b	internal delay of a buffer
R_b	output resistance of a buffer
R_w	resistance of a wire
C_w	capacitance of a wire
ρ_R	unit resistance of a wire
ρ_C	unit capacitance of a wire
N_i	a set of nodes at the ith iteration
n_j^i	the jth node at the ith iteration
M_i	matching result of the ith iteration
$c_{j,k}$	the cost of the jth and kth nodes
$dis(n_1, n_2)$	Manhattan distance between n_1 and n_2
C_{n_1}, C_{n_2}	downstream capacitance of n_1 and n_2
D_{n_1}, D_{n_2}	accumulated subtree delays of n_1 and n_2

TABLE I
NOTATIONS.

cuit. Detailed information of the definitions below can be found in the ISPD 2009 Contest website [18].

A. Clock Slew Rate

The restriction on clock slew describes the requirement on the rising and falling signal rate, so as to maintain the signal integrality. It is defined to be the lasting time from 10% to 90% of the signal strength, and the upper limit is assumed to be 100ps. During the simulation of CNS, it is necessary to maintain the signal rising time under this upper limit throughout the whole clock network.

B. Clock Skew

The skew of a clock network means the difference of the source-to-sink delay among all the sinks. S is used to denote the set of sinks which is $\{s_1, \ldots, s_{|S|}\}$. d_{s_i} represents the internal delay between the source and the sink s_i. To minimize the skew, we need to build a clock network with all d_{s_i} as close as to each other as possible. Notice that the skew is not determined by the average delay of the sinks, but by the difference between the maximum and the minimum. The equation for clock skew (C_s) calculation is shown below.

$$C_s = \max\{d_{s_i} | \forall s_i \in S\} - \min\{d_{s_i} | \forall s_i \in S\}$$

C. CLR

Owing to the uncertainty of the manufacturing process, there may be voltage uncertainty at each driving node. The supplied voltage at each driving node may vary from V_{dd_1} to V_{dd_2}. This will affect the accuracy of the buffer delay calculation. SPICE simulations with these two different voltage sources V_{dd_1} and V_{dd_2} are used in this case to simulate the voltage uncertainty. CLR is the main criterion for evaluation. It is determined by the difference between the maximal and minimal clock skew values under the two given voltage sources.

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

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

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

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

$$CLR = \max\{p_{1_{max}}, p_{2_{max}}\} - \min\{p_{1_{min}}, p_{2_{min}}\}$$

¿From the above equations, we can see that CLR, to a certain extent, represents both the pure clock skew and the clock skew due to voltage variation. We will use it together with the pure clock skew for CNS performance evaluation.

D. Resources of Wires and Buffers

We assume two types of buffers and two types of wires. The unit capacitance and resistance cost of wires are predefined as ρ_C and ρ_R. The two types of buffers are both inverted, so the signs of the two signals at a merging point should be the same. The upper limit of the total capacitance is also set to limit the total power consumption. In the rest of this paper, we will use parity check of the number of downstream buffer levels to represent signal sign check. We will also use the capacitance cost to represent the power usage of the network.

Based on the buffer and wire types, we construct a look-up table for slew checking. We run SPICE simulations to find out the maximum distance between two successive buffers with one specified wire type in order not to violate the slew constraint. These two successive buffers may have different sizes (different number of buffers connected in parallel). The input parameters of this table are the buffer type, the size of the first buffer, the size of the second buffer and the wire type. The output of this table is the maximum distance allowed between these two buffers. The procedure of buffer insertion can be simplified with this look-up table.

III. METHODOLOGY

In this paper, we propose two clock network synthesizers applying our dual-MST for geometric matching. DMST is the approach that calculates the delay by Elmore model only. DMSTSS is the approach that also performs SPICE simulations in order to adjust the location of each merging point. The design flow of these two approaches is shown in figure 1.

Based on the approach of DMST, dual-MST is used first for geometric matching with the clock terminals. A new recursive buffer insertion method is developed together with blockage handling to deal with the buffer distribution problem for each matching pair. After matching and buffer insertion, the locations of the merging nodes at the bottom level of the tree can be decided. This procedure is performed again with the merging nodes instead of the clock terminals. The locations of the merging nodes at the next level can then be decided similarly.

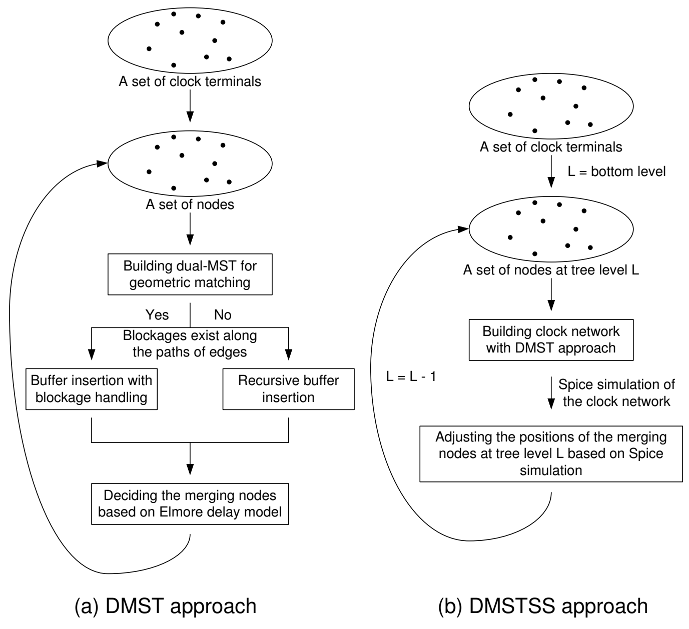
A set of clock terminals
A set of nodes
Building dual-MST for geometric matching
Yes
No
Blockages exist along the paths of edges
Buffer insertion with blockage handling
Recursive buffer insertion
Deciding the merging nodes based on Elmore delay model
(a) DMST approach
A set of clock terminals
L = bottom level
A set of nodes at tree level L
Building clock network with DMST approach
L = L - 1
Spice simulation of the clock network
Adjusting the positions of the merging nodes at tree level L based on Spice simulation
(b) DMSTSS approach

Fig. 1. Design flow of (a) DMST approach and (b) DMSTSS approach

This procedure is performed iteratively until a clock tree is constructed. For the approach of DMSTSS, SPICE simulation is applied. A clock network is built first according to the approach of DMST. The computed clock skew at the merging nodes is zero but the actually clock skew may not be zero based on the SPICE simulation. According to the simulated clock skew, the locations of the merging nodes at the bottom level are adjusted, or some snaking wires are inserted. The clock network is then re-built based on the merging nodes at the bottom level instead of the clock terminals. The clock network will thus be re-built at each level until the location of the final merging node at the top level of tree is fixed. DME technique [6] is also applied. Thus, segment is used instead of points to represent the set of best merging locations and deferred embedding is applied to reduce the total wirelength. The total capacitance of the initial clock network is a good indication for buffer sizing. To determine the size of buffer, one initial tree without parallel buffers is first built for the estimation of the total capacitance. If the estimated total capacitance of the initial clock network is too small, larger buffers (more buffers connected in parallel) can be inserted.

A. Iterative Geometric Matching

An iterative geometric matching technique is developed for topology construction. In the ith iteration, let N_i denote the group of nodes, n_j^i denote the jth node, and M_i denote the matching result (a specific definition of a geometric matching of one iteration can be found in [3]). The cost of the merging of two nodes n_j^i and n_k^i is denoted as $f_c(n_j^i, n_k^i)$, and it is calculated and stored at the beginning of the ith iteration. The Manhattan distance of two nodes is used as the cost in our implementation. The maximal cost of the ith iteration C_{max}^i is denoted as below, and we can get C_{min}^i accordingly.

$$C_{max}^i = \max_{(n_j^i, n_k^i)} \{ f_c\left(n_j^i, n_k^i\right) : \forall \left(n_j^i, n_k^i\right) \in M_i \}$$

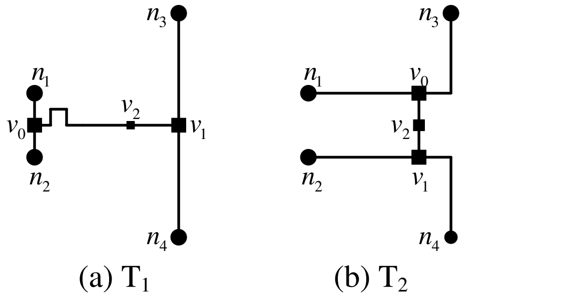

Fig. 2. Comparison of (a) an unbalanced tree and (b) a fully-balanced tree.

Unlike traditional matching algorithms which focused on wirelength minimization [1, 7], our approach wants to build a topology which is close to a fully-balanced tree. In each level, all the connection paths of a fully-balanced tree have identical Manhattan distance, despite their corresponding parents and children nodes are different. An example is shown in figure 2. In T_1, $\{n_1, \dots n_4\}$ is the group of terminals, v_0 and v_1 are the merging nodes at the first level and v_2 is the root. The same naming rule is also applied in 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. However, T_2 is fully-balanced and T_1 is unbalanced with $dis\left(v_2, v_0\right) \neq dis\left(v_2, v_1\right)$. Therefore, the clock skew of T_1 is more sensitive towards process variation factors. Meanwhile, the real time delay is not of linear relationship with wirelength, and asymmetric buffer insertion in T_1 will increase the delay variations. As a matter of fact, fully-balanced trees such as T_2 is preferable in CNS. In our approach, we will get close to this target by means of reducing C_{max}^i.

Procedure 1 partition(N_i)

```
Require: $N_i \leftarrow$ the group of nodes to be merged in the $i$th iteration
  if $|N_i| = 2$ then
    merge$\left(n_1^i, n_2^i\right)$
  else if $|N_i| \leq 1$ then
    return
  else
    Build dual-MST with $|N_i| - 2$ edges inserted.
    Two groups of nodes $N_i'$ and $N_i''$ are formed respectively.
    if $|N_i'|$ is odd and $|N_i''|$ is odd then
      $\left(n_a^i, n_b^i\right) = \arg_{(n_u^i, n_v^i)} \min\{f_c\left(n_u^i, n_v^i\right)$
      $|\forall n_u^i \in N_i', \forall n_v^i \in N_i''\}$
      merge$\left(n_a^i, n_b^i\right)$
      remove $n_a^i$ from $N_i'$
      remove $n_b^i$ from $N_i''$
    end if
    partition($N_i'$)
    partition($N_i''$)
  end if
```

To achieve this target, a novel method to build the dual-MST is developed. It is an iterative approach, therefore we only need to describe its functionality in one iteration. Based on the Kruskal's MST algorithm, our method is made up of a se-

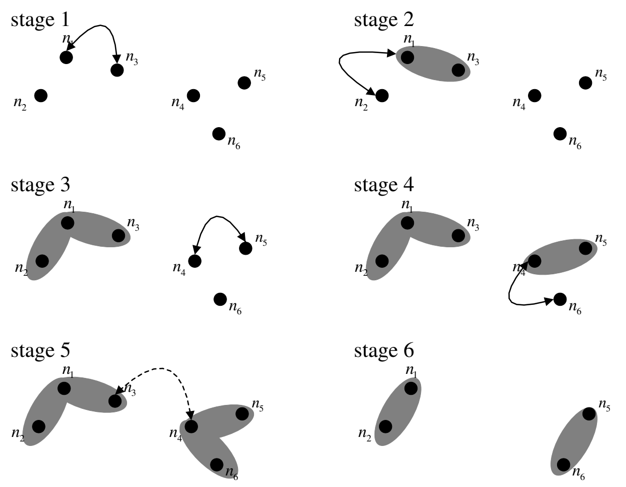

Fig. 3. An example of iterative geometric matching.

quence of recursive bipartitioning processes. The detailed description of constructing a dual-MST is shown in procedure 1. Generally, during the generation of the Kruskal's MST of N_i, we only inserted $|N_i| - 2$ edges (without any cycles involved). As a matter of fact, two subtrees N_i' and N_i'' are generated to form a bipartition. A specific case is illustrated in figure 3, in which $|N_i| = 6$. Let $e_{j,k}$ represent the edge connecting n_j^i and n_k^i. The edges are sorted in ascending order according to their costs, and in this case the sorted edge list would be

$$e_{1,3}, e_{1,2}, e_{4,5}, e_{4,6}, e_{3,4}, e_{2,3}, e_{5,6}, \ldots$$

At first, $|N_i| - 2 = 4$ edges $\{e_{1,3}, e_{1,2}, e_{4,5}, e_{4,6}\}$ are sequentially added, $N_i' = \{n_1, n_2, n_3\}$ and $N_i'' = \{n_4, n_5, n_6\}$. Because $|N_i'|$ and $|N_i''|$ are both odd numbers, we continue to add one more edge from the list (still no cycles involved), which is $e_{3,4}$. Therefore, n_3 and n_4 form a match, $N_i' = \{n_1, n_2\}$ and $N_i'' = \{n_5, n_6\}$. The bipartition procedure continues on N_i' and N_i''. Here $|N_i'| = |N_i''| = 2$, so the recursion of bipartition is terminated according to procedure 1, and two more pairs $\{n_1, n_2\}, \{n_5, n_6\}$ are formed. Finally, the matching result at iteration i, M_i, is $\{(n_1, n_2), (n_3, n_4), (n_5, n_6)\}$ and C_{max}^i is $f_c(n_3, n_4)$.

During each iteration of the matching, every node should be paired according to our dual-MST algorithm (one node is left when $|N_i|$ is odd), so $\lceil \log_2 |S| \rceil$ iterations are performed. At the ith iteration, the time complexity of cost calculations equals $O\left(|N_i|^2\right)$. On average, the number of partitioning stages is $\lceil \log_2 (|N_i|) \rceil$, and the complexity is $\left(\frac{|N_i|}{2^{j-1}}\right)^2$ for the jth stage. Therefore, the total complexity of the ith iteration is still $O\left(|N_i|^2\right)$, and $|N_i| = \lceil \frac{|S|}{2^{i-1}} \rceil$. Finally, the time complexity on average for the whole geometric matching is shown below.

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

Our approach aims at constructing a clock tree topology that is close to a fully-balanced tree. It is only a heuristic with 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 net. Specific comparison can be found in the section of the experimental results.

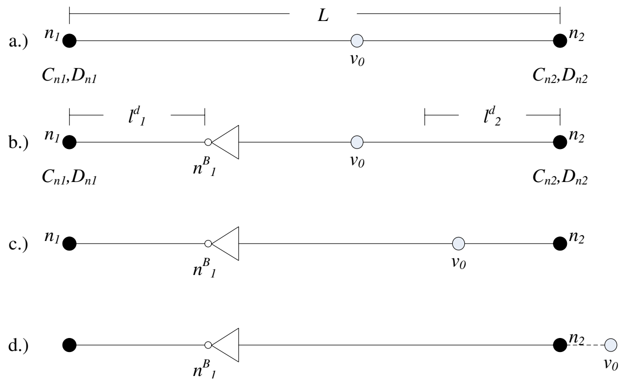

Fig. 4. Design flow of recursive buffer insertion.

B. Recursive Buffer Insertion

After the step of geometric matching, a set of node pairs are obtained. In order to reduce the skew and satisfy the slew limitation, the location of the merging point between a node pair should be determined appropriately. In addition, buffers are inserted if necessary. Elmore RC model is applied for delay estimation. The buffer delay and wire delay are computed by $D_b = d_b + R_b \times C_d$ and $D_w = R_w \times \left(\frac{1}{2} \times C_w + C_d\right)$ respectively. Real-time simulation of signal slew rate costs much more time. Thus, we build a look-up table (T_{slew}) in advance. This table can provide respective driving length for different combinations of buffer and wire types. Reducing the total amount of capacitance is also one of our major targets. Our approach is implemented to cut down the usage of buffer insertion and wire snaking.

An example is illustrated in figure 4 to demonstrate our approach for the construction of clock network. n_1 and n_2 are the two nodes to be merged. The distance between n_1 and n_2 is denoted by L. Their downstream capacitances and sub-tree delays are denoted by C_{n_1} and C_{n_2}, and D_{n_1} and D_{n_2} respectively. Based on their downstream capacitances and the difference of their sub-tree delays, the location of the merging point, v_0 can be determined. The merging process of these two nodes can be completed if and only if both of the following two conditions are satisfied: (1) A buffer at v_0 can drive n_1 and n_2 directly without causing any slew violation. In figure 4a, if a buffer is inserted at v_0, the corresponding driving capacitance should be $C_{n_1} + C_{n_2} + L \times \rho_C$. If the driving capacitance is larger than the value obtained from the look-up table (T_{slew}), the clock slew will be larger than the limitation. (2) The parities of downstream buffer levels of n_1 and n_2 are the same. While any of these two conditions is not satisfied, a recursive buffer insertion technique is applied.

While either one condition is not satisfied, buffer insertion should be performed. We assume that the distance between the merging node v_0 and n_1 is X_1. The distance between v_0 and n_2 is X_2. The maximal driving length of n_1 and n_2 are denoted by l_1^d and l_2^d respectively. The maximal driving length means that if a buffer is inserted further away from l_1^d or l_2^d, the corresponding clock slew will be larger than the limitation. If $X_1 - l_1^d$ is greater than $X_2 - l_2^d$, a buffer will be inserted to drive n_1. The maximal driving length l_1^d of n_1 is obtained from the look-up table (T_{slew}) based on C_{n_1}. A buffer will be inserted at n_1^B in this case. An example is shown in figure 4b. The location of the merging point, v_0 should then be re-computed based on the Elmore RC model such as in figure 4c. If v_0 is not located between n_1^B and n_2 such as in figure 4d, the merging point should be located at n_2. In this case, delay balancing can be obtained by inserting snaking wires. This step will be repeated until condition 1 and 2 are satisfied.

C. Blockage Handling

There may be pre-assigned blockages in the routing region. The connection of two nodes n_1 and n_2 may thus be constrained to avoid buffer insertion on blockages. Complete wire detour followed by post-buffer-insertion will work, but it will cost a lot of resources. Instead, we develop a blockage handling method with free wire propagation and concurrent buffer insertion. It is based on the maze routing technique in global routing. However, we develop several techniques to adapt it to satisfy CNS constraints. We first impose a $m \times n$ grid on the whole routing region, and n_1 and n_2 (n_1 and n_2 are still the segments due to the application of DME approach) are then mapped to two groups of the closest grid points. Two independent maze routers mz_1 and mz_2 with priority queues are utilized for n_1 and n_2, and each queue is sorted by the downstream delay values of its elements. Let D_1 and D_2 denote the downstream delay value of the first element of the two queues. We assume that $D_1 < D_2$ at the beginning, so mz_1 is selected to be the primary router, and it will search the four adjacent grid points of the current position. During the routing of mz_1, wire connection together with buffer insertion is concurrently considered along the path of each node. The downstream delay is then increased, so D_1 will be updated. When D_1 becomes bigger than D_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 function in turns, and the values of D_1 and D_2 stay close to each other. When the two routers meet at the end, their delays will not differ a lot. This method can result in less buffer insertions and wire detours.

An approximate performance comparison is shown in figure 5. We can see that our blockage handler can save buffers and wires, so it will reduce the total capacitance. Notice that the size of the grid graph ($m \times n$) is manually determined in our algorithm, m and n can be scaled up or down in order to have either better CPU time cost or routing quality. In the experiments, m and n are set as 1000.

IV. EXPERIMENTAL RESULTS

In this section, our experimental results are presented. We implement our clock network synthesizer in the C language and the program is executed on the Linux operating system with an Intel Core2 Quad 2.4GHz CPU and 4GB memory. The benchmark circuits used in the experiments are released from the ISPD09 CNS contest [18]. Detailed information of the benchmark circuits is shown in table II. According to the benchmark circuits, two specified types of wires and buffers can be used in the clock network.

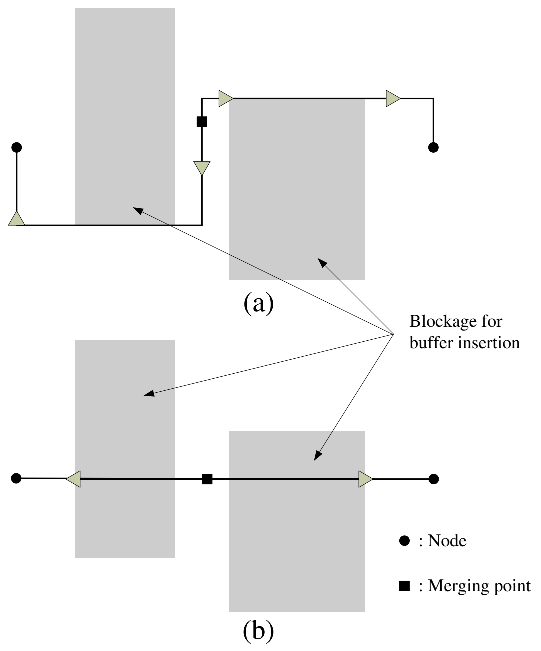

Fig. 5. Outputs of different blockage handling techniques: (a) complete connection detour and (b) our approach.

Circuits	No. of sinks	No. of blockages	Limitation on total capacitance
11	121	0	118000
12	117	0	110000
21	117	0	125000
22	91	0	80000
31	273	88	250000
32	190	99	190000
nb1	330	53	42000
33	209	80	195000
34	157	99	160000
35	193	96	185000
nb2	440	1346	88000

TABLE II
CIRCUIT INFORMATION OF THE BENCHMARKS FROM ISPD 2009.

We implement the techniques of two widely used matching methods MMM [1] and CL [7] 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 [7], the ratio is between $\left(\frac{1}{2}, 1\right)$ so $\frac{2}{3}$ is a proper setting). The average result in table III is obtained from execution on the benchmarks in table II, and SPICE simulation is involved in every case. In table III, CLR (ps), nominal clock skew (ps), total capacitance (fF), total wirelength (nm) and CPU time (seconds) are shown for performance comparison, and the results are the average value of all the 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 general, our matching technique, DMST, can outperform MMM and CL in clock skew, CLR, total capacitance and CPU time.

	CLR (ps)	Skew (ps)	Cap. (fF)	wirelength (mm)	CPU (s)
MMM	13.35	8.75	142498.18	283.06	448
CL	13.79	9.88	123224.61	223.04	650
DMST	12.25	7.72	117587.45	239.62	317

TABLE III
COMPARISON BETWEEN DIFFERENT MATCHING METHODS.

A summary of the performance of our clock network synthesizer is shown in table IV. CLR, capacitance percentage, total wirelength and CPU time are listed. We run our program with two scenarios. DMST is the approach without using SPICE. DMSTSS is the approach with SPICE simulation. By using SPICE simulation, the locations of the merging points are adjusted and the clock skew can be reduced. We compare our results with those obtained from the two winners (T4 and T6) of the CNS contest in ISPD09 and the best CLR (the result without violation of the slew and total capacitance constraints) of each circuit among all the teams in the competition. Notice that the programs of the winners in the contest are executed under the platform with dual-core AMD Opteron 2.8GHz CPU and 128 GB memory.

According to the results in table IV, we can see that the CLR obtained from both DMSTSS and DMST are outstanding. Although SPICE simulation is not applied in DMST, the CLR is still comparable and is even better than for some cases the best result in the ISPD09 contest. In addition, the CPU time used by DMST is extremely small (mostly less than one second). DMST has this efficient performance because it uses a tree structure that is close to a balanced tree. On the other hand, the CLR can still be minimized even though the Elmore delay model is not accurate to evaluate the locations of the merging points. For DMSTSS, the clock skew (based on particular voltage source) can be further minimized by obtaining the delays of subtrees from the SPICE simulations. Although the CPU time used is increased, it is still the fastest approach compared to the winners of the contest. Generally, the average CLR of our approach is only 38.0% of T4, 34.9% of T6, and 51.3% of the best results in the contest. Meanwhile, the average CPU time of our approach is only 1.8% of T4, 4.7% of T6 and 1.7% of the best results from different teams in the contest. CLR is the first criterion for evaluation according to the ISPD 2009 contest, and the experimental result shows the effectiveness of our approach.

V. CONCLUSION

In conclusion, a dual-MST approach for clock network synthesis has been proposed. Several novel techniques are applied to solve the CNS problem. Our DMST approach can give a tree structure that is close to a balanced tree. This can reduce the sensitivity of clock skew against voltage variation. Experimental results show that the CLR can be improved significantly with much shorter CPU time by applying our approach.

VI. ACKNOWLEDGMENTS

The work described in this paper was partially supported by the Competitive Earmarked Research Grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 5262/07E).

REFERENCES

[1] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh. Clock routing for high-performance ics. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 573–579, June 1990.

[2] M. Edahiro and T. Yoshimura. Minimum path-length equi-distant routing. In *Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems*, pages 41–46, 1992.

[3] A. Kahng, J. Cong, and G. Robins. High-performance clock routing based on recursive geometric matching. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 322–327, July 1991.

[4] W. C. Elmore. The transient response of damped linear networks with particular regard to wide band amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948.

[5] R.-S. Tsay. Exact zero skew. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 336–339, Nov 1991.

[6] T. H. Chao, Y. C. Hsu, and J. M. Ho. Zero skew clock net routing. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 518–523, July 1992.

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

[8] L. P. P. P. and van Ginneken. Buffer placement in distributed rc-tree networks for minimal elmore delay. In *International Symposium on Circuits and Systems*, pages 865–868, May 1990.

[9] J. D. Cho and M. Sarrafzadeh. A buffer distribution algorithm for high-speed clock routing. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 537–543, June 1993.

[10] S. Natarajan, S. L. Sam, D. Boning, A. Chandrakasan, R. Vallishayee, and S. Nassif. A methodology for modeling the effects of systematic within-die interconnect and

Circuits	DMSTSS			DMST			T4			T6			Best case		
	CLR	$C\%$	CPU	CLR	$C\%$	CPU	CLR	$C\%$	CPU	CLR	$C\%$	CPU	CLR	$C\%$	CPU
11	12.2	79.3	180	20.5	79.6	0.3	26.7	85.5	14764	32.3	73.9	3892	22.3	89.9	23358
12	10.9	89.3	213	22.2	90.3	0.3	25.7	84.7	13934	32.2	73.5	3944	22.2	87.9	14992
21	12.1	83.2	210	22.0	83.6	0.3	30.5	80.8	14978	34.3	74.3	4587	19.6	86.7	26420
22	9.9	79.4	113	17.1	79.4	0.3	24.5	81.8	7189	30.4	70.0	2005	16.4	85.0	9432
31	13.4	83.4	777	39.1	84.6	1.3	45.1	73.5	40088	51.3	81.5	17333	45.1	73.5	40088
32	11.5	82.4	420	27.7	82.6	0.4	36.9	80.1	3566	40.3	77.4	10599	18.4	89.9	2888
nb1	13.8	82.0	82	26.1	84.3	0.6	NA	NA	NA	19.8	63.1	477	19.8	63.1	477
Average	12.0	82.7	285	25.0	83.5	0.5	31.6	81.1	15753	34.4	73.4	6119	23.4	82.3	16807
33	13.4	83.6	483	22.0	83.0	0.4	NA	NA	NA	NA	NA	NA	NA	NA	NA
34	11.3	85.7	354	26.2	85.7	0.4	NA	NA	NA	NA	NA	NA	NA	NA	NA
35	13.1	85.0	453	20.2	84.9	1.9	NA	NA	NA	NA	NA	NA	NA	NA	NA
nb2	13.1	87.9	202	35.4	89.0	1.6	NA	NA	NA	NA	NA	NA	NA	NA	NA

TABLE IV
COMPARISON OF CLR BETWEEN OUR APPROACH AND THE WINNERS IN THE ISPD09 CLOCK NETWORK SYNTHESIS CONTEST.

device variation on circuit performance. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 172–175, June 2000.

[11] S. Sauter, D. Schmitt-Landsiedel, R. Thewes, and W. Webber. Effect of parameter variations at chip and wafer level on clock skews. *IEEE Transactions on Semiconductor Manufacturing*, 13(4):395–400, November 2000.

[12] I.-M. Liu, T.-L. Chou, D. F. Wong, and 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*, pages 33–38, Nov 2000.

[13] Y. P. Chen and D. F. Wong. An algorithm for zero-skew clock tree routing with buffer insertion. In *Proceedings of European Design and Test Conference.*, pages 230–236, 1996.

[14] A. Rajaram, J. Hu, and R. Mahapatra. Reducing clock skew variability via crosslinks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1176–1182, June 2006.

[15] A. Rajaram and D. Z. Pan. Variation tolerant buffered clock network synthesis with cross links. In *Proceedings of ACM International Symposium on Physical Design*, pages 157–164, April 2006.

[16] A. Rajaram and D. Z. Pan. Robust chip-level clock tree synthesis for soc designs. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 720–723, June 2008.

[17] C.-M. Chang, S.-H. Huang, Y.-K. Ho, J.-Z. Lin, H.-P. Wang, and Y.-S. Lu. Type-matching clock tree for zero skew clock gating. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 714–719, June 2008.

[18] C. N. Sze, P. Restle, G.-J. Nam, and C. Alpert. Ispd2009 clock network synthesis contest. In *Proceedings of ACM International Symposium on Physical Design*, pages 149–150, March 2009.