A novel obstacle-aware multiple fan-out symmetrical clock tree synthesis

Meng Liu^{a)}, Zhiwei Zhang, Wenqin Sun, and Donglin Wang

Institute of Automation, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, 100190, China

a) liumeng2013@ia.ac.cn

Abstract: Clock tree design plays a critical role in improving chip performance and affecting power. In this paper, we propose a novel symmetrical clock tree synthesis algorithm, including tree architecture planning, matching, merging, embedding and buffer insertion. Obstacle-aware placement and routing are also integrated into the algorithm flow. By using NGSPICE simulation for benchmark circuits, our skew results decrease by 17.2% while using less than 24.5% capacitance resource compared with traditional symmetrical clock tree. Further, we also validated the algorithm in ASIC design.

Keywords: symmetrical clock tree, multiple fan-out, matching algorithm, buffer insertion, obstacle-aware placement and routing

Classification: Integrated circuits

References

- [1] D. Wang, *et al.*: “MaPU: A novel mathematical computing architecture,” IEEE International Symposium on High Performance Computer Architecture (HPCA) (2016) 457 (DOI: [10.1109/HPCA.2016.7446086](https://doi.org/10.1109/HPCA.2016.7446086)).
- [2] M. Donno, *et al.*: “Power-aware clock tree planning,” Proc. of the 2004 International Symposium on Physical Design (2004) 138 (DOI: [10.1145/981066.981097](https://doi.org/10.1145/981066.981097)).
- [3] S. Tam, *et al.*: “Clock generation and distribution for the first IA-64 microprocessor,” IEEE J. Solid-State Circuits **35** (2000) 1545 (DOI: [10.1109/4.881198](https://doi.org/10.1109/4.881198)).
- [4] R. S. Tsay: “An exact zero-skew clock routing algorithm,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **12** (1993) 242 (DOI: [10.1109/43.205004](https://doi.org/10.1109/43.205004)).
- [5] Lee and L. Igor: “Contango: Integrated optimization of SoC clock networks,” Proc. of the Conference on Design, Automation and Test in Europe (DATE) (2010) 1468.
- [6] X. W. Shih and Y. W. Chang: “Fast timing-model independent buffered clock-tree synthesis,” Design Automation Conference (DAC) (2010) 80 (DOI: [10.1145/1837274.1837296](https://doi.org/10.1145/1837274.1837296)).
- [7] J. T. Yan, *et al.*: “Top-down-based symmetrical buffered clock routing,” Proc. of the ACM Great Lakes Symposium on VLSI (2012) 75 (DOI: [10.1145/2206781.2206801](https://doi.org/10.1145/2206781.2206801)).
- [8] C. N. Sze, *et al.*: “Ispd2009 clock network synthesis contest,” Proc. of the 2009 International Symposium on Physical Design (2009) 149 (DOI: [10.1145/1514932.1514965](https://doi.org/10.1145/1514932.1514965)).

- [9] G. Wu, *et al.*: “Detailed placement algorithm for VLSI design with double-row height standard cells,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **35** (2016) 1569 (DOI: [10.1109/TCAD.2015.2511141](https://doi.org/10.1109/TCAD.2015.2511141)).
- [10] H. Zhou, *et al.*: “Efficient minimum spanning tree construction without Delaunay triangulation,” Proc. of the 2001 Asia and South Pacific Design Automation Conference (2001) 192 (DOI: [10.1145/370155.370320](https://doi.org/10.1145/370155.370320)).
- [11] S. Hu, *et al.*: “Fast algorithms for slew constrained minimum cost buffering,” Design Automation Conference (DAC) (2006) 308 (DOI: [10.1145/1146909.1146990](https://doi.org/10.1145/1146909.1146990)).
- [12] R. S. Shelar, *et al.*: “An algorithm for routing with capacitance/distance constraints for clock distribution in microprocessors,” Proc. of the 2009 International Symposium on Physical Design (2009) 141 (DOI: [10.1145/1514932.1514964](https://doi.org/10.1145/1514932.1514964)).

1 Introduction

Clock skew and power consumption are two main concerns for clock tree synthesis. Related papers [1, 2] indicate that clock network typically contributes more than 40% of the entire chip power. On the other hand, as the technology of semiconductor process is scaling down to 10 nm and below, clock tree synthesis becomes even more challenging due to on-chip variation (OCV) effects [3].

There are two common variations of clock trees, traditional clock trees and structured clock trees. The most influential algorithm in traditional clock trees is called deferred-merge embedding (DME) which aims at zero clock skew by using unbalanced buffer and wire compensation [4], apparently, this style tree is not suitable for resisting OCV. Nevertheless, structured trees have always been the ideal way to improve quality of results. H-tree is a classic structured clock tree architecture with nearly equal geometric lengths that can be effective against OCV problem. However, H-tree does not account for the uneven distribution of sinks and can not minimize wire length cost [5]. Compared with H-tree, with inheriting advantages of H-tree, the symmetrical tree-like structure has the characteristics of shorter wire length, smaller number of buffers, and much more flexible structure. The idea of symmetrical clock tree structure is proposed in papers [3, 6, 7]. But clock tree topology in their approaches is formed by the binary tree and trinary tree which means much more pseudo-sinks have to be added to balance sink load. Additionally, low fan-out design will easily cause high clock buffer level number and the level number usually affects the latency and slew of clock tree. Further, in their works [6, 7], since obstacle processing is not considered, it is hard to apply this related structure to practical chip design. In the research from IBM [3], they have discussed the necessity of length matching in structured clock tree design. However, their proposed tree need priority of placement and routing in a custom flow, so their work has a disadvantage on the design flexibility.

In order to overcome the drawbacks of the previous works, we propose a design flow for obstacle-aware multiple fan-out symmetrical clock tree synthesis. We test our proposed algorithm on both public benchmarks [3, 8] and one ASIC physical design [1]. By using our algorithm flow, the level number, skew results and

capacitance cost are all reduced. The contributions of this paper are summarized as follows.

1. The methodology of clustering and merging is proposed for the lack of multiple fan-out clock tree research.
2. A look-up table considering power consumption is built through simulation to guarantee accurate slew and capacitance in the constraint.
3. An obstacle-aware algorithm is proposed to generate the clock tree topology with an overall view on obstacles.

2 Symmetrical clock tree synthesis

2.1 Problem formulation

Given a placed design with a set of n clock sinks $S = \{s_1, s_2, \dots, s_n\}$, with their locations, $\{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_n, y_n\}$, a library of buffers, clock skew constraint, clock slew rate constraint and capacitance constraint [8].

Problem: obtain a symmetrical clock tree with appropriate level planning, topology generation, buffering and routing resources such that the given design constraints are likely to be satisfied.

Fig. 1 shows our design flow. The details of these approaches will be discussed in following parts.

Fig. 1. The proposed algorithmic flow.

2.2 Tree architecture planning

Tree architecture planning can be determined based on sink number n . We have factorized n into prime number combination. For example, if the sink number is 50 and the max fan-out is 5. 50 can be factorized with 5, 5, 2, this means clock tree with 50 sink nodes can be planned as three levels. The first two levels adopt the method of five fan-out tree merging. While the last level only needs to use the binary tree. In the contrast, other related approaches only use binary tree [3, 6, 7] in the current situation, the 50 has to be extended to 64 which can be divided by 2. In other words, 16 pseudo-sinks have to be added to finish the balanced tree design. Apparently, our approach can help save much more pseudo-sinks in the complex design. We have proposed the decision-making approach, as shown in Algorithm 1. The subfunction $f(n)$ is defined to help factorize n . The return value of $f(n)$ is t . If t equals 0, it represents factorization is done. If t equals 1, the algorithm needs to go on. Let n accumulate until it is divisible completely. Based on the given sink number and maximum branching, the procedure gives the valid planning number list and the corresponding number of pseudo-sinks.

Algorithm 1: Tree architecture planning

Input: number of sinks n , max fan-out number m

Output: planning number list p , adjusted number of sinks an

```

1: define the subfunction  $f(n)$  to do the factorization,  $t$  is the return value.
2: for  $i = 2 \rightarrow m$  do
3:   while  $n/i == 0$  do
4:      $x \leftarrow n/i$ 
5:     push_back  $i$  to  $p$ 
6:   end while
7: end for
8: if  $f(n) == 1$  then
9:    $n+ = 1$ 
10: else
11:   get the  $p$  and  $an$ 
12: end if

```

2.3 Greedy matching algorithm

For the given tree architecture planning strategy, clusters can be built based on the distribution of nodes. By using the clustering result, merging which is the basic process of CTS can be done. Thus, reasonable clustering can effectively reduce the wire length cost. The idea of using Edmond's clustering algorithm for binary tree has been discussed in the paper [6]. However, this paper also points out the deficiency of this algorithm for trinary tree or more complex trees clustering, that is, NP-complex problem is formed. Moreover, other related articles also have not discussed this situation [3, 7].

The key to this problem can be written as:

$$cluster_radius = \min\{\max\{R_1, R_2, \dots, R_n\}\}, \quad (1)$$

$cluster_radius$ in equation (1) means the maximum radius value of all clusters and should be as small as possible, apparently, it is a bottleneck problem. R_n is the radius value of one cluster and is computed by the Manhattan distance from the center of gravity to every point in the current cluster. The center of gravity (x_g, y_g) in one cluster is defined as equation (2), where $i, j \in \{0, 1, 2, \dots\}$.

$$x_g = \frac{\sum_{i=1}^n x_i}{n} \quad y_g = \frac{\sum_{i=1}^n y_i}{n} \quad (2)$$

Before the determination of clustering solution, we firstly introduce the merging process in the local cluster. Fig. 2(a) is the first step of building the *tilted rectangular region* (TRR) for each point. The TRR which is a collection of points within a fixed distance of a Manhattan arc can be used to represent potential embedding positions for tree nodes [4]. Additionally, our TRR is composed of the 45-degree or 135-degree line segments. Fig. 2(a) compares the effects of Manhattan rings selection between the strict $cluster_radius$ and relaxed $cluster_radius$. The strict radius of this cluster is computed by $|x_g - x_1| + |y_g - y_1|$. The red dot line in Fig. 2(a) represents the potential locations of the center of gravity

(x_g, y_g) . But in most situations, the center of gravity is not the available location of merging point because of legalization. In the following, we have proved the important significance of relaxed *cluster_radius* by using Euclidean distance.

Theorem 1 If 1.5 times of Manhattan distance r is used, the Manhattan ring can help guarantee the intersection for TRR in each local cluster.

Proof We have marked the Manhattan ring (the solid line) and its Euclidean ring (the dotted line) in the Fig. 2(a). Assume that the blue dot represents the nearest point on the Manhattan ring from (x_1, y_1) , and this nearest distance is r . The red dot which is on the Manhattan ring represents the furthest Euclidean distance from (x_1, y_1) , and this Manhattan distance is also r . Obviously, based on the geometrical relationship, we can set $\sqrt{2}$ times of r as the strict radius distance to cover the potential center of gravity. The relaxed radius distance can be 1.5 times of r to be large enough to amplify the viable area ($1.5r > \sqrt{2}r$). In this way, $1.5r$ can be set as the updated radius distance to guarantee the existence of intersection of TRR. Based on TRR model, we can build the intersection or merging segments among potential merging nodes.

Fig. 2. Nodes with Manhattan rings.

As shown in Fig. 2(b), it describes how five nodes can build the TRR. Each node point builds its own TRR based on a same Manhattan arc, the Manhattan arc is the bottleneck radius of current tree level which is computed as Fig. 2(a), and this arc ensures that the five TRRs have intersection. This gray region is the constructed TRR which represents the potential embedding node locations.

Another issue is how to solve this NP-hard clustering problem. In other words, which nodes can be matched to build TRR. In equation (1), this problem established can be partitioned into a set of subproblems which is suitable for greedy algorithm. In Chris Chu's work [9], they have got the conclusion that both Edmond's maximum weighted matching algorithm and the greedy algorithm provide similar wire length results while the latter approach has a much better runtime. So the greedy matching algorithm (GMA) is chosen which can help find a lower bound of the solution quality at least.

The followings present a five fan-out tree clustering process as an example. Let $G = (V, E)$ be a graph and $w : E \rightarrow \mathfrak{R}$ be a function which assigns a weight to each of the edges of G . The weight table is built based on the Manhattan distance between each two nodes. The group of tightly-connected nodes can be clustered by nearest neighbor (NN) algorithm which is illustrated in Fig. 3(a). We find the nearest node from the beginning node to achieve the relatively minimal distance cost, and repeat this process to get the cluster. The optimal solution is $1 \rightarrow 10 \rightarrow 9 \rightarrow 8 \rightarrow 7$.

Fig. 3. The local cluster construction and GMA process.

Fig. 3(b) describes the GMA process. To simplify the process, we divide the plane into four regions according to the quadrant rule and select a starting direction region. It has proved in [10] that the quadrant partition can exhibit good results which is similar with the octant partition. The node that is furthest from center of gravity can be found in the given direction region, and this node is set as the initial node which we regard it as the boundary node. We need to pay attention to it because it is most likely to cause wire length cost from the center of gravity. The first iteration following counter-clockwise direction will end with the cluster division result and a *cluster_radius* value. Algorithm 2 shows the steps of our methodology for GMA. The whole algorithm ends until all iterations based on different beginning direction regions finish and the optimal strategy can be easily selected. In the following, we have proved the converge of GMA in wire length cost.

Algorithm 2: Greedy matching algorithm

Input: nodes set S with size N , merging nodes number n , graph model

$G = (V, E)$, weight table W

Output: clusters with reasonable nodes

```

1: cluster number  $M = N/n$ 
2: direction region  $V = \{v1, v2, v3, v4\}$ 
3: for  $v \in V$  do
4:   choose the boundary node  $s\_b$  from  $S$ 
5:   for  $i \in M$  do
6:     find minimum  $w$  from  $W$ 
7:     add  $e$  to  $S$ 
8:     remove  $e$  and all edges incident to  $e$  from  $E$ 
9:   end for
10:  get the cluster results
11: end for

```

Fig. 4. Octal partition and the uniqueness property.

Theorem 2 For any distribution of nodes, the *greedy matching algorithm* (GMA) can converge in wire length cost.

Proof The plane with respect to the endpoint s is divided into eight regions from $R1$ to $R8$ by the two rectilinear lines and the two 45 degree lines, as shown in Fig. 4(a). The paper [10] has proven that for the endpoint s , a region R with respect to s has the uniqueness property. In Fig. 4(b), for every pair of points $p, q \in R1$, there exists $\|pg\| < \max(\|sp\|, \|sq\|)$ and $\|pg\|$ represents the distance between p and q . We can prove the radius R_n of each cluster has an upper bound as follows, $R_n \leq \|pg\| < \max(\|sp\|, \|sq\|)$. Since the radius has the limited value, GMA can converge in wire length cost.

2.4 Buffering strategy

To meet the slew-rate constraint, feasible buffers are needed to insert onto clock tree paths [11]. In order to construct the symmetrical clock tree, paths at each level must be guaranteed to have the same number of buffers and corresponding types. Because of the balanced structure, we are more concerned about slew and capacitance parameter. In our approach, we establish the buffer look-up table through simulation by NGSPICE [8]. Starting from the clock root point, the

strategy of buffer insertion including the maximum spacing and buffer types can be determined based on fan-out number for each level through dynamic programming. We also add some power information into the look-up table for the practical ASIC design.

According to the simulation results based on ISPD09 [8], we depict the slew parameter and capacitance parameter curves, as an example shown in Fig. 5(a) and Fig. 5(b). From above figures, we can see capacitance and distance are approximate linear relationship. Based on spacing value, the distance of buffer on the path can be determined in the constraint condition. The characterization curves for different fan-out and wire types can be also easily obtained.

Fig. 5. Slew and capacitance parameter curves. (a) Slew increases with spacing increasing (the wire type is 0, the fan-out number is set to 2, and one buffer1 drives two buffer2). (b) Capacitance increases with spacing increasing (the wire type is 0, the fan-out number is set to 2, and one buffer1 drives two buffer2).

In order to analyze power characteristic, the dynamic power dissipation *power* is given by the following equation (3). *C* is the load capacitance, V_{dd} is the supply voltage, *f* is the clock frequency and α is the switching activity. The parameter δ in equation (4) represents the differential relationship between power consumption and spacing. In other words, it can be transformed into a differential of capacitance and spacing. This value can be evaluated based on slope of capacitance parameter curves.

$$power = \alpha C V_{dd}^2 f \quad (3)$$

$$\delta = \frac{\partial power}{\partial spacing} = \frac{\alpha V_{dd}^2 f \partial C}{\partial spacing} \quad (4)$$

Considering the impact of three kinds of power consumption, they are the cell internal power consumption (*inter-power*), cell leakage power (*leak-power*) and net switching power consumption (*switching-power*), the relation can be expressed as the following equation (5).

$$power = \alpha_1 * inter_power + \alpha_2 * leak_power + \alpha_3 * switching_power \quad (5)$$

α_1 , α_2 and α_3 dominate weights of respective power. Finally, the power information computed can be added into the look-up table. A table of one ASIC design according to TSMC 28 nm technology is summarized in the experiment result section.

2.5 Obstacle-aware placement and routing planning

In chip designs, detailed placement of clock buffer cells inserted has to be considered. These cells must be aligned to placement sites on rows [9] according to technology rule. Assume row direction is consistent and horizontal, by using the following equation (6), the vertical position of corresponding cell can be legalized which is shown in Fig. 6(a),

$$y_i = y_0 + N * Site, \quad (6)$$

where $N \in \{0, 1, 2, \dots\}$ and $Site$ is the row height. Moreover, the overlapping of cells will be checked by using equation (7). We regard blockages and fixed standard cells as obstacles. Assume that each computed cell c_i with a position (x_i, y_i) , a width w_i , a height h_i and each obstacle o_j with a position (x_j, y_j) , a width w_j , a height h_j . The following formulas are used for checking overlapping. The available location of cell will be found by movement easily.

$$\begin{aligned} & ((obs_x_j \leq x_i + w_i \leq obs_x_j + obs_w_j) \vee \\ & (obs_x_j \leq x_i \leq obs_x_j + obs_w_j)) \wedge \\ & ((obs_y_j \leq y_i + h_i \leq obs_y_j + obs_h_j) \vee \\ & (obs_y_j \leq y_i \leq obs_y_j + obs_h_j)) \end{aligned} \quad (7)$$

Algorithm 3: Obstacle-aware planning of routing algorithm

Input: the driver p with (p_x, p_y) , the receiver c with (c_x, c_y) , the obstacle set obs

Output: path points vector $path_point$

```

1: define direction vector  $s$  from  $p$  to  $c$  :  $s = (c_x - p_x, c_y - p_y)$ 
2: for  $i = 1 \rightarrow obs.size()$  do
3:   check if intersects with  $s$  and push back to  $new\_obs$ 
4: end for
5: push back  $p$  to  $path\_point$  and  $parent\_now = p$ 
6: for  $i = 1 \rightarrow new\_obs.size()$  do
7:   find the nearest point  $np$  from  $new\_obs$  based on  $s$ 
8:    $parent\_now = np$ 
9:   push back  $parent\_now$  to  $path\_point$ 
10: end for
```

After tree topology and buffering strategy have been determined, the next step is to finish routing. In our work, rectangular obstacles in layout are considered and we firstly introduce obstacle-aware routing among related papers [3, 6, 7]. Obstacle-aware routing means the path from parent to child needs to avoid obstacles which intersect its shortest path and buffers from the path need to find their reasonable locations, as shown in Fig. 6. The obstacle-aware planning of routing algorithm is shown in Algorithm 3. Firstly, we should identify all wires that intersect obstacles and put every four points from obstacles, child point and parent point into the path points table. Then, beginning from parent, we can find the nearest point and set it as the next path point. And next path point is determined based on the direction vector. After obstacle-aware placement and routing, the

Fig. 6. (a) Obstacle-aware placement. (b) Obstacle-aware routing.

existing router tool can help us finish length matching routing based on the bottleneck length [12].

3 Experimental results

The proposed approach was implemented by using standard C++ language on a PC workstation of Intel Core i7-3537U CPU. Four benchmarks in ISPD09 clock network synthesis contest [8] and IBM benchmark circuits r1–r5 [4] are used to test our symmetrical clock tree synthesis algorithm. For comparison, NGSPICE simulation based on the 45 nm PTM process technology [8] is used to evaluate the quality of result. The results of clock skew and resource usage are shown in Table I and Table II. We report wire capacitance because it serves as a good metric which is directly proportional to wiring and power usage. Compared with Shih's approach [6], we obtain 17.2% decrement on skew result while we also use 24.5% less capacitance resource on the average, with only about 17% runtime overhead.

Table I. Experimental results for ISPD09 benchmarks.

Benchmarks	Sinks	Shih's Approach [6]			Our Approach		
		Skew (ps)	Cap (fF)	Runtime (s)	Skew (ps)	Cap (fF)	Runtime (s)
ispd09f11	121	0.110	95749	0.05	0.093	73595	0.06
ispd09f12	117	0.051	96609	0.05	0.038	72211	0.06
ispd09f21	117	2.321	108755	0.06	2.076	81453	0.07
ispd09f22	91	1.160	69696	0.02	1.986	58721	0.04

Table II. Experimental results for IBM benchmarks.

Benchmarks	Sinks	Shih's Approach [6]			Our Approach		
		Skew (ps)	Cap (fF)	Runtime (s)	Skew (ps)	Cap (fF)	Runtime (s)
r1	267	1.510	13829	0.07	0.323	10778	0.08
r2	598	1.770	31056	0.28	0.338	30572	0.32
r3	862	2.310	44188	1.05	0.407	40903	1.23
r4	1903	2.540	98450	3.35	1.386	88721	3.92
r5	3101	3.010	171228	5.56	2.532	158721	6.45

Fig. 7. Layout of APE. (a) The topology of global clock tree. (b) The layout with length matching.

Fig. 8. The dimensional curve from look-up table.

Additionally, we have applied our symmetrical CTS algorithm into the MaPU core physical design [1]. The MaPU cores in this chip are called APE, which stands for Algebraic Processing Engine. The clock tree in old version which is built purely by commercial EDA tool (IC Compiler) contributes approximately 55% of APE dynamic power. In the new version, the global clock tree is using our symmetrical CTS flow. Fig. 7 shows the clock tree in the APE layout. All clock registers have been clustered and driven by several multi-source drivers. In both Fig. 7(a) and (b), we can see the entire clock tree planning is based on obstacle-aware and very structured.

A dimensional curve from look-up table according to the TSMC 28 nm technology is summarized in Fig. 8. We observe that when the spacing increases, the power increases rapidly. The increase of fanout number does not directly lead to an increase in power consumption. Table III presents the results of commercial and our proposed flow for CTS (*C*: commercial CTS flow, *O*: our proposed CTS flow). The modules of APE, PCIe and RapidIO in MaPU chip are used for comparison. The first row in this table includes the max level number of tree paths, the whole number of clock buffers, WNS (worst negative slack of setup), TNS (total negative slack of setup) and the sum of power consumption. We perform CTS with MCM (multi-corner multi-mode) and results of the worst corner are listed. The early/late timing derate of OCV constrain is 0.96/1.04 separately and the slew constraint is

Table III. CTS results for three modules

	APE		PCIe		RapidIO	
	<i>C</i>	<i>O</i>	<i>C</i>	<i>O</i>	<i>C</i>	<i>O</i>
Level number	36	20	23	13	20	12
Clock buffers	8952	2488	3024	821	2324	587
WNS (ns)	−0.36	−0.12	−0.15	−0.03	−0.15	−0.01
TNS (ns)	−925.7	−325.6	−3.12	−0.42	−27.3	−0.18
Latency (ns)	1.85	1.64	0.92	0.86	0.85	0.76
Power (mW)	1036.2	400.3	73.8	34.5	45.2	25.3

20 ps. In summary, the results in Table III show that all designs with our proposed flow have much less level number of CTS. The clock buffers amount is reduced by 72.7% on average. The improvement in WNS is up to 240 ps and the TNS is much reduced due to the better result of each path. The latency values of all designs are similar to values from commercial EDA tool while ours are controllable. Our proposed flow separately reduces the power consumption by 61%, 53%, 44% because the reduction of buffers amount.

4 Conclusion

In this paper, a novel obstacle-aware synthesis methodology for multiple fan-out symmetrical clock tree is proposed. In the proposed methodology, a greedy matching algorithm (GMA) is used for clustering and can guarantee the wire length optimization. We also consider merging, embedding, buffer insertion and obstacle-aware planning for multiple fan-out situation. The symmetrical clock tree structure design makes it have the ability to resist OCV and no longer limited to the regular floorplan design, while the skew and capacitance results reduce.

Acknowledgments

This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences (under Grant XDA-06010402).