Multi-Domain Clock Skew Scheduling-Aware Register Placement to Optimize Clock Distribution Network

Naser MohammadZadeh[1], Minoo Mirsaeedi[1], Ali Jahanian[2], Morteza Saheb Zamani[1]
[1] Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran
[2] Department of Electrical and Computer Engineering, Shahid Beheshti University, G. C. , Tehran, Iran
{szamani,mohammadzadeh,mirsaeedi}@aut.ac.ir {jahanian@sbu.ac.ir}

Abstract. **Multi-domain clock skew scheduling is a cost effective technique for performance improvement. However, the required wire length and area overhead due to phase shifters for realizing such clock scheduler may be considerable if registers are placed without considering assigned skews. Focusing on this issue, in this paper, we propose a skew scheduling-aware register placement algorithm that enables clock tree optimization by considering domains assigned to registers in placement. Our experimental results show that the proposed approach remarkably decreases clock wire length and clock network power consumption at the cost of a slight increase in total wire length.**

I. INTRODUCTION

In a sequential circuit, clock signals may not arrive at all registers at the same time due to the differences of interconnect delays in the clock distribution network. The consequent differences in clock arrival times are known as clock skews. Since meeting the setup and hold constraints of a sequential circuit is very complicated in the presence of clock skews, an approach that has been followed by [1], [2], [3], and [4] is to design the clock distribution network ensuring zero clock skew.

Clock skew schedulers [5], [9], [12] on the other hand, view the clock skew as a manageable resource rather than a liability. They intentionally introduce skews to registers to improve the circuit performance. The designated skews are then implemented by specific layout of the clock distribution network. However, in practice, a skew schedule with a large set of arbitrary values cannot be realized in a reliable manner. This is because the implementation of dedicated delays using additional buffers and interconnects is highly susceptible to intra-die variations of process parameters.

On the other hand, instead of tuning clock skews of registers, retiming can physically relocate registers to balance the delays without changing the functionality of the circuit [6]. It was observed in [5] that retiming and clock skew scheduling are discrete and continuous optimizations with the same effect. The equivalence between retiming and skew scheduling has been used in prior researches [7], [8], [9], [10]. Although retiming is a powerful sequential optimization technique, its practical usage is limited due to its impact on the verification process, e.g. equivalence checking and functional simulation. Furthermore, the use of retiming to maximize the performance may cause a steep increase in the number of registers [11], that requires a larger effort for clock distribution and results in higher power consumption.

Recently, multi-domain clock skew scheduling has been proposed in [12]. Multiple clocking domains are routinely used in designs to realize target clock frequency and also to address specific timing requirements. For example, a special clocking domain that delivers a phase-shifted clock signal to the registers close to the chip inputs and outputs is regularly used to achieve timing closure for ports with extreme constraints on their arrival and required times. The motivation behind the multi-domain skew scheduling is based on the fact that large phase shifts between clocking domains can be implemented reliably by using dedicated, possibly expensive circuit components such as “structured clock buffers" [13]. In [12], Ravindran et al. showed that a clock skew schedule using a few domains combined with a small within-domain latency can reliably implement the full optimization potential of clock skew scheduling. They proposed an algorithm based on a branch-and-bound search to assign registers to clock domains and also they modified Burns' algorithm [14] to compute the skew values.

For a user-specified number of domains, the algorithm in [12] computes the optimal skew for each domain. However, the user has no control on the distribution of the domains. Furthermore, the algorithm does not consider delay padding [15], a technique that fixes hold time violations by inserting extra delays on short paths without increasing the delay on any long paths. In other words, the clock period obtained by the algorithm in [12] may be sub-optimal if delay padding is allowed, as demonstrated in [16], [17], and [18]. Therefore, the authors of [19] formulated the clock skew scheduling problem on a user-specified finite set of prescribed clock domains. They proposed a polynomial-time algorithm that finds an optimal domain assignment for each register such that the clock period is minimized with possible delay padding. The algorithm performs skew scheduling with respect to the user-specified requirements and also considers how to insert extra padding such that both setup and hold time

978-3-9810801-5-5/DATE09 © 2009 EDAA

constraints are satisfied with the minimal clock period. However, the authors of [19] have not considered register placement and did not propose any clock distribution network structure.

In [20], a clock scheduling algorithm has been proposed that first clusters registers based on their locations and then modifies the clusters to improve the clock period. The algorithm may route the clock network inefficiently because it does not have any regular clock routing plan. Moreover, this algorithm generates a skew schedule with a large set of arbitrary values that, in practice, may not be realized in a reliable manner.

In this paper, we propose a register placement strategy that considers the multi-domain clock skew scheduling scheme. We also present a new clock distribution network structure to efficiently distribute registers. The proposed register placement algorithm uses clock scheduling information obtained from the algorithm proposed in [19] to place registers according to the suggested clock distribution network structure. In other words, both the *domain information* and *the proposed clock network structure* contribute in the placement of registers. The approach not only optimizes clock period but also decreases the number of delay elements. Moreover, it decreases clock wire length that leads to lower clock power consumption. Our approach may have negative impact on the traditional placement objectives such as total wire length of regular nets. As experimental results show, the clock wire length reduction is remarkable while the total wire length overhead is small.

The rest of this paper is organized as follows. In Section 2, we review basic concepts about skew scheduling and partitioning-based placement. Section 3 proposes our clock distribution network structure. In Section 4, our clock skew-aware register placement strategy is proposed. Section 5 includes experimental results. Finally, Section 6 concludes the paper.

II. Basic Concepts

A. Clock Distribution Network Synthesis and Skew Scheduling

A clock distribution network delivers the clock signal from the clock source to a set of clock sinks (registers) such that the clock signal delay to each sink satisfies certain constraints. Usually, due to the differences in interconnect delays; clock signals do not arrive at all registers at the same time. The consequent differences in clock arrival time are known as clock skew. If clock signal delays to registers R_i and R_j are denoted as s_i and s_j respectively, considering the shortest and longest combinational path between R_i and R_j, setup and hold time constraints are as follows:

$$\begin{aligned} s_i - s_j &\le T_{clk} - T_{setup}^{j} - D_{ij}^{\max} = w_{ij} \\ s_j - s_i &\le D_{ij}^{\min} - T_{hold} = w'_{ij} \end{aligned} \tag{1}$$

where T_{clk} is the clock period and D_{ij}^{max} and D_{ij}^{min} represent the maximum and minimum path delays, respectively. Path delays can be calculated by a static timing analysis tool.

The skew specification (prescribed skew) $s_i - s_j$ can be calculated by a prescribed skew scheduling procedure [19] to consider constraints between R_i and R_j. The skew specification can also be expressed as the delay target for each register directly. A clock distribution network, which is usually a tree rooted at the clock source, is constructed through clock routing [21] for a given register placement and skew specification, such that the skew specification is satisfied. The skew scheduling and clock routing procedures can also be integrated together [22].

B. Hierarchical Placement

Placement is a classical problem in VLSI physical design. Among different proposed placement methods, multi-level hierarchical techniques are regarded indispensable to solve placement problem especially for large circuits [23].

A typical top-down hierarchical placement approach can be generalized as follows: at a given hierarchical level, the layout area is partitioned into several bins. All cells in the circuit are assigned into these bins such that some placement objectives, like total wire length are optimized. If a cell is assigned to a particular bin, it will be placed within the area of this bin in the final layout. At next levels, each bin is partitioned into finer bins bounded to the parent bin. Thus more and more detailed information about physical locations of cells can be acquired in each level. The iterative partitioning stage terminates when there are only a few cells in each placement bin.

Since the construction of clock network has become more critical during recent years, some register placement methods have been proposed [24], [25], and [26]. In addition to traditional placement objectives, register placers try to place registers such that clock net length is optimized. Since the location of each cell is determined step by step in hierarchical placement approaches, most of the register placers develop their idea based on such approaches [27].

III. Proposed Clock Distribution Network Structure

Clock routers may be zero-skew or bounded skew. In bounded skew clock tree generation, the main challenging matter is how to achieve the required skew for each register. Multi-domain skew scheduling is one of the widely accepted methods to alleviate the area and power consumption overheads of aggressive delay elements along the clock tree.

Some of the proposed clock tree construction methods [18] try to guide most of the delay elements to the main stem of the clock tree to reduce the required delay elements. However, since registers are fixed in such methods, the quality of the final clock tree is strictly dependant on the initial placement of the registers. For example, if co-domain registers (registers that are in the same domain) are distributed far from each other, the total required delay elements may grow significantly because more delay elements are absorbed by the clock tree branches.

Therefore, fixing the registers before clock skew scheduling may result in large number of delay elements. Thus, it is more preferable to perform placement under consideration of clock

skew scheduling information and based on a specific clock tree infrastructure. In addition to providing skew for all skew domains, such clock tree infrastructure should not put any strict constraint on the placement procedure. Therefore, we propose a hierarchical structure for bounded skew clock distribution network.

The proposed clock tree structure in consists of a hierarchical set of Manhattan circles [24] in which each circle is skewed with respect to higher level. It is noting that all registers which are located on a circle experience the same skew since each point on the Manhattan circle has the same Manhattan distance (Manhattan radius) from its center.

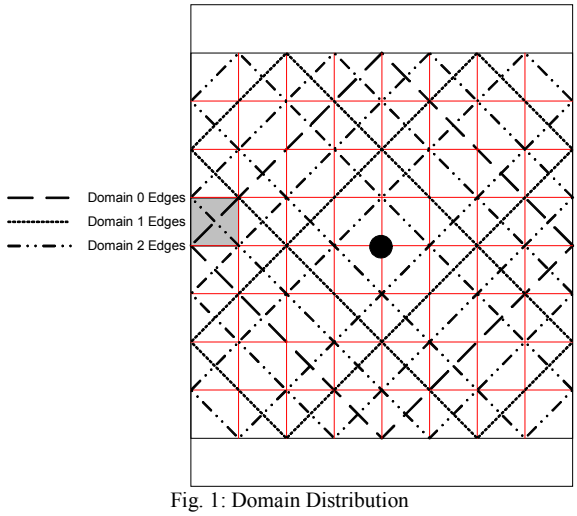

Fig. 1: Domain Distribution

The proposed clock network structure is shown in Fig. 1 for a design with three domains. The vertical and horizontal lines show the borders of partitioning bins after three levels of the hierarchical partitioning-based procedure execution. Domain zero consists of one Manhattan circle whose center is the center of the chip. This Manhattan circle is shown in Fig. 1 by dashed lines. Manhattan circles of domain one, shown as dotted lines, are constructed on the edges of domain zero. The centers of these Manhattan circles are the middle points of the domain zero edges. Generally, Manhattan circles of each domain are constructed on the edges of the previous domain and their centers are the middle points of the edges of the previous domain.

Further to efficient utilization of delay elements along the clock tree, every skew domain is fairly accessible within the design area. Therefore, each register can be placed very close to where it would be placed by a regular placement algorithm. As a consequence, such clock distribution network structure has little total wire length overhead. It is assumed that the clock signal is delivered to the Manhattan circles of domain i+1 using an H-tree rooted at the center of the Manhattan circle of domain i. Therefore, due to the regularity of the network, its construction imposes negligible process and run time overhead.

In clock routing, the location of the clock tree root does not have to be the same as the clock source. The root node can always be connected to the clock source directly. Therefore, clock routing algorithms do not rely on the location of the clock source. Without loss of generality, the clock source is assumed to be at the center of the chip. Another assumption in Fig. 1 is that the chip height is greater than the chip width. This assumption also does not lose generality.

As stated before, delay elements are used to construct different skew domain. These elements are placed at the centers of the Manhattan circles. The number of such elements is equal to the total number of Manhattan circles having at least one register with the same domain on their edges. In other words, a delay element is placed if the Manhattan circle surrounding it contains at least one register whose domain is equal to that Manhattan circle.

In order to construct the final clock tree, we connect all co-domain registers located in a common partition using a Minimum Steiner tree. Then, since each point along the Manhattan circle can easily be routed to the delay element located at its center, the nearest point of such Steiner tree to the gravity center of registers should be routed to the associated Manhattan circle.

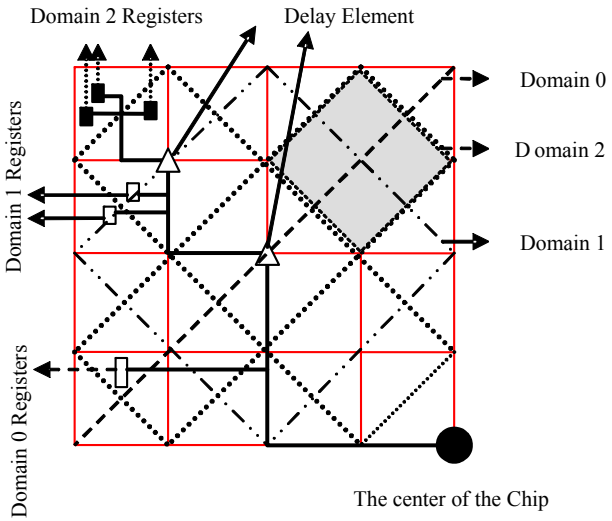

Fig. 2: Part of the clock tree generated by our method. The shaded Manhattan circle does not have any register, so no delay element has been placed at the center of it.

Although such a clock routing method does not guarantee zero skew within co-domain registers, but it can result in negligible skew because each partition has small area. Moreover, since each partition contains a few cells (up to 6 cells), only a small number of co-domain registers may be located in one partition. Therefore, the Steiner tree construction process does not impose considerable run time overhead. Fig. 2 shows upper left quarter of the clock tree structure generated by our method. The delay elements are shown by triangles. As the figure shows, registers are connected to proper delay element and each delay element is connected to lower level delay element. This procedure continues until the center of the chip is reached.

IV. Our Skew-Aware Placement Algorithm

The following definitions are necessary to describe our skew-aware placement algorithm.

Definition 1. A partitioning bin is *legal* for all registers of a specified skew domain, if it intersects with a Manhattan circle of that domain. For example the bin shaded in Fig. 1 is a *legal* bin for all registers of domains 0 and 2.

Definition 2. Considering timing constraints for each register R_k, its *permissible skew range* (PSR) is defined as follows:

$$\text{PSR}=\left[\max\left(skew_j - w'_{kj}, skew_i - w_{ik}\right), \min\left(skew_i + w'_{ik}, skew_j + w_{kj}\right)\right] \quad \text{for all } R_i \in \text{Predecessors}(R_k),\ R_j \in \text{Successors}(R_k) \tag{2}$$

where w_{ik} and w'_{ik} were defined in Eq. (1) and $skew_i$ and $skew_j$ are clock skew at input of the registers R_i and R_j, respectively.

Definition 3. *Migration overhead* ($MO_{i,k}$) of each register R_i from its current location to bin k can be evaluated as:

$$MO_{ik} = \alpha.PSR(R_i, Bin_k) + \beta WL(R_i, Bin_k) \tag{3}$$

where PSR is the permissible skew range of R_i when Bin_k is a legal bin and WL is the wire length overhead due to the register migration. $WL(R_i, Bin_k)$ can be formally expressed as:

$$WL(R_i, B_k) = \begin{cases} 0 & \text{if } B_i = B_k \\ D_{B_i,B_k} & \text{otherwise} \end{cases} \tag{4}$$

where B_i represents the bin currently containing R_i and $D_{Bi,Bk}$ is equal to the Manhattan distance between centers of bins B_i and B_k.

Proposed Placement Algorithm

1: Assign the optimal skew to each register [19]
2: Run partitioning algorithm [28].
3: Refine the partitioning results based on registers' domains and proposed clock network.
4: if (the # of cells in each global bin > stopping threshold)
5: level = level + 1
Go to 2.
6: Place cells in each bin.

Fig. 3: The proposed placement algorithm

Our placement algorithm is shown in Fig. 3. First, for a user-specified number of domains, using the formula proposed in [19], the optimal skew is assigned to each register such that minimum clock period would be acquired. Using domains assigned to registers in first stage and clock structure proposed in previous section our partitioning-based placement algorithm guides registers such that they are placed close to the associated Manhattan circles in the clock structure. Accordingly, during an iterative procedure, using a multi-level partitioning engine such as hMetis [28], cells of each bin are partitioned into two separate bins such that aside from reducing the cut-sizes between bins, the registers are placed in *legal* bins.

In the last stage, after last level of partitioning, the cells assigned to bins are placed by a modified version of Dragon algorithm [30] that places the registers on or close to their domains edges.

During skew scheduling, in order to maximize clock frequency, it has been tried to assign subsequent registers into different skew domains. Therefore, the skew between registers in the same domain is not the matter of concern because such registers are not usually subsequent. On the other hand, in some cases where subsequent registers are assigned into the same skew domain, the large slack of their associated combinational path makes them robust against clock skew. During the proposed register placement, it is attempted to provide the largest skew permissible range for each register in order to make it robust against possible clock skew violations as described in Equation 4. Moreover, the most significant wire length overhead occurs due to domain 0 registers because such registers should be held near the Manhattan ring of domain 0 during the placement. On the other hand, Manhattan rings of higher domains are more distributed within the design and are available fairly on the chip. Therefore, register assignment to such domains have little wire length overhead. In other words, more skewed domains are preferred to domain 0 with respect to total wire length overhead. Accordingly, our register placement attempts to assign most of the registers to more skewed domains.

V. Experimental Results

We implemented the proposed placement algorithm in C++ on a 3 GHz Pentium IV with 1 gigabytes of memory. The attempted benchmarks were chosen randomly from IWLS'05 [29] which were synthesized using 180 nm technology with six layers of metal. Table 1 contains the characteristics of the benchmarks.

TABLE 1: The Characteristic Of Benchmarks

Benchmark	#cells	#registers	Proportion (%) #registers / #cells
s38417	8278	1564	18.8
s38584	6724	1178	17.5
s5378	1294	163	12.5
Ethernet	46771	10544	22.5
pci-bridge32	16816	1498	8.9

The number of domains is an important factor to evaluate the proposed idea. The analysis done in [19] shows that 4 is a good choice for the number of domains, so the results were calculated with 4 domains.

TABLE 2: EXPERIMENTAL RESULRS IN TERM OF CLOCK WIRE LENGTH AND TOTAL WIRE LENGTH

Benchmark	Clock wire length (um$\times 10^8$)			Total wire length excluding clock net (um$\times 10^9$)			Total wire length (um$\times 10^9$)			Clock period before/after clock skew scheduling (ns)
	WDP	MDAP	Improvement (%)	WDP	MDAP	Overhead (%)	WDP	MDAP	Overhead /Improvement (%)	
s38417	2.5206	2.3200	8.64	1.28	1.29	0.77	1.53206	1.522	+0.66	1262/100
s38584	1.3767	1.3210	4.21	1.01	1.03	1.9	1.14767	1.1621	-1.24	724/100
s5378	0.1616	0.1523	6.10	0.163	0.163	0	0.17916	0.17823	+0.52	662/143
Ethernet	36.1039	24.6404	46.52	16.5	17.1	3.5	20.11039	19.56404	+2.79	1484/100
pci-bridge32	6.0511	4.9396	22.50	2.87	3.26	11.96	3.47511	3.75396	-7.42	1158/675
Average			**17.6**			**3.62**			**-0.93**	

To evaluate the proposed algorithm, the placement process was implemented in two cases and their results were compared. The first case is a wire length-driven placement (WDP) that uses original Dragon and hMetis algorithms and the second one is our multi-domain clock skew scheduling-aware placement (MDAP) which uses modified Dragon and hMetis algorithms to consider the clock skew domain information for the cell and register placement.

Table 2 provides experimental results in terms of clock wire length and total wire length. As mentioned previously, in the proposed clock network, the clock length consists of the total wire length of H-trees as the back-bone of the network, Steiner trees within co-domain registers and the routes between some points on the perimeter of Manhattan circles and their centers. The column *Clock wire length* provides a comparison of clock length obtained in the proposed method and WDP. The last column of Table 2 shows the period of clock before and after clock skew scheduling. In small designs, the achievable improvements are not as much as more complex and larger ones due to the restricted number of registers. The column *Total wire length* includes the total wire length for all nets including the clock net. Plus sign in the column *Overhead/Improvement* of Table 2 means that the total wire length has been improved and minus sign means the total wire length has been aggravated. These results show that clock length improvement is large in comparison to total wire length overhead. Experiments show that considerable improvement was achieved in clock length at the cost of small overhead in total wire length (17.6% clock wire length reduction vs. 0.93% total wire length overhead).

Table 3 shows clock net power consumption in micro-watts per megahertz in the two cases. Since our algorithm inserts restricted number of delay elements along the clock network, the power consumption overhead of delay elements is not remarkable. Accordingly, total power consumption would be dominated by total wire length power consumption. Therefore, considering large activity of clock signal, the proposed algorithm can be considered more efficient in term of power consumption, compared with a wire-length driven placement. As these results show, we have about %18.78 improvement (on average) in clock power.

TABLE 3: CLOCK POWER CONSUMPTION (μw per MHz)

Benchmarks	WDP	MDAP	Gain (%)
s38417	20.95	20.61	1.63
s38584	11.44	11.31	1.15
s5378	1.34	1.10	22.12
Ethernet	300.05	204.77	46.52
pci-bridge32	49.67	40.54	22.50
Average			**18.78**

Compared with a wire length driven placement algorithm, the proposed algorithm has around 10% run time overhead. Such process effort is consumed to calculate the migration overhead of each register and update registers' permissible ranges.

A. Post-placement timing constraints verification

The first constraint for clock skew scheduling is timing closure. Our timing reports show that none of timing constraints have violated after the proposed placement process. As stated in Section 2, we use pre-placement timing analysis to assign registers to domains. Due to lack of enough information about location of cells, the pre-placement static timing analysis is not as accurate as the post-placement timing analysis. In other words, it is concerned that setup and hold time constraints may be violated after placement. Therefore, we use a standard timing analysis tool [30] after placement to verify our skew scheduling algorithm and examine that timing constraints have not been violated. The timing constraints have been calculated based on minimum clock period obtained from multi-domain clock skew scheduling algorithm.

VI. CONCLUSION

In this paper, we introduced a placement strategy considering multi-domain clock skew scheduling and a new clock network structure to improve the clock wire length and power consumption. In the proposed methodology, each register is assigned to one domain based on timing information extracted from post-synthesis data. Experimental results show that considering register skew information during placement reduces clock routing wire length and

power consumption at the cost of a slight increase in total wire length. The proposed algorithm can be more effective for large and complicated circuits with larger register/cell proportion.

REFERENCES

[1] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh. "Clock routing for high-performance IC's," In Proc. *DAC*, 1990.

[2] A. B. Kahng, J. Cong, and G. Robins, "High-performance clock routing based on recursive geometric matching," In Proc. *DAC*, 1991.

[3] S. Pullela, N. Menezes, and L. T. Pillage, "Skew and delay optimization for reliable buffered clock trees," In Proc. *ICCAD*, 1993.

[4] R. S. Tsay, "Exact Zero Skew," In Proc. *ICCAD*, 1991.

[5] J. P. Fishburn, "Clock Skew Optimization," In *IEEE Transactions on Computers*, Vol. 39, No.7, 1990.

[6] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," In *Advanced Research in VLSI*, 1983.

[7] H. G. Martin "Retiming by combination of relocation and clock delay adjustment," In Proc. *DAC*, 1993.

[8] B. Lockyear and C. Ebeling "Minimizing the effect of clock skew via circuit retiming," Technical Report, Dept. of Computer Science and Engineering, University of Washington, 1993.

[9] L. F. Chao and E. H. M. Sha, "Retiming and cock skew scheduling for synchronous systems," In Proc. *ISCAS*, 1994.

[10] S. S. Sapatnekar and R. B. Deokar, "Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits," In *TCAD*, Volume 15, Issue 10, 1996.

[11] G. Even, I. Y. Spillinger, and L. Stok, "Retiming revisited and reversed," In *TCAD*, Volume 15, Issue 3, 1996.

[12] K. Ravindran, A. Kuehlmann, and E. Sentovich, "Multi-domain clock skew scheduling," In Proc. *ICCAD*, 2003.

[13] K. M. Carrig, "Chip clocking effect on performance for IBMs sa-27e ASIC," In *IBM Micronews*, 2000.

[14] S. M. Burns, "Performance analysis and optimization of asynchronous circuits," *PhD thesis*, California Institute of Technology, Computer Science Department, 1991.

[15] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," In Proc. *ICCAD*, 1993.

[16] Y. Kohira and A. Takahashi, "Clock period minimization method of semi-synchronous circuits by delay insertion," In *IEICE Transaction Fundamentals*, Volume 88-A, Issue 4, 2005.

[17] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," In Proc. *International Symposium of Microarchitecture*, 2003.

[18] B. Taskin and I. Kourtev, "Delay insertion method in clock skew scheduling," In Proc. *ISPD*, 2005.

[19] C. Lin and H. Zhou, "Clock skew scheduling with delay padding for prescribed skew domains," In Proc. *ASP-DAC*, 2007.

[20] M. Saitoh, M. Azuma, and A. Takahashi, "Clustering based fast clock scheduling for light clock-tree," In Proc. *DATE*, 2001.

[21] C. A. Tsao and C. Koh, "UST/DME: a clock tree router for general skew constraints," In *ACM Transactions on Design Automation of Electronic Systems*, Volume 7, Issue 3,2002.

[22] G. Venkataraman, C. N. Sze, and J. Hu, "Skew scheduling and clock routing for improved tolerance to process variation," In Proc. *ASP-DAC*, 2005.

[23] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can recursive bisection produce routable placements," In Proc. *DAC*, 2000.

[24] Y. Lu and et al, "Register placement for low power clock distribution network," In Proc. *ASP-DAC,* 2005.

[25] W. Shen, Y. Cai, X. Hong, and J. Hu, "Activity and register placement aware gated clock network design," In Proc. *ISPD*, 2008.

[26] L. Huang and et al, "Clock network minimization methodology based on incremental placement," In Proc. *ASPDAC*, 2005.

[27] Y. Lu, C. Sze, X. Hong, "Navigating register placement for low power clock network design," In *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Volume E88-A , Issue 12, 2005.

[28] Karypis and et al., "Multilevel hypergraph partitioning: application in VLSI domain," In Proc. *DAC*, 1997.

[29] http://www.iwls.org/iwls2005/benchmarks.html

[30] T. Taghavi, X. Yang, and B.K. Choi, "Dragon2005: large scale mixed size placement tool," In Proc. *ISPD*, 2005.

[31] Magma Tool, http://www.magma.com/