Navigating Registers in Placement for Clock Network Minimization

Yongqiang Lu[a], C. N. Sze[b], Xianlong Hong[a], Qiang Zhou[a],
Yici Cai[a], Liang Huang[a], Jiang Hu[b]

[a]Dept. of Computer Science and Technology
Tsinghua University
Beijing, 100084, China
86-10-62795428
hxl-dcs@mail.tsinghua.edu.cn

[b]Dept. of Electrical Engineering
Texas A&M University
College Station, TX 77843, U.S.A.
1-979-847-8768
jianghu@ee.tamu.edu

ABSTRACT
The progress of VLSI technology is facing two limiting factors: power and variation. Minimizing clock network size can lead to reduced power consumption, less power supply noise, less number of clock buffers and therefore less vulnerability to variations. Previous works on clock network minimization are mostly focused on clock routing and the improvements are often limited by the input register placement. In this work, we propose to navigate registers in cell placement for further clock network size reduction. To solve the conflict between clock network minimization and traditional placement goals, we suggest the following techniques in a quadratic placement framework: (1) Manhattan ring based register guidance; (2) center of gravity constraints for registers; (3) pseudo pin and net; (4) register cluster contraction. These techniques work for both zero skew and prescribed skew designs in both wirelength driven and timing driven placement. Experimental results show that our method can reduce clock net wirelength by 16%~33% with no more than 0.5% increase on signal net wirelength compared with conventional approaches.

Categories and Subject Descriptors
B.7.2 Design Aids

General Terms
Algorithms, Performance, Design

Keywords
Clock network, placement, low power, variation tolerance

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

1. INTRODUCTION
The progress of VLSI technology is facing two limiting factors: power and variation. Clock network is a sub-circuit with paramount importance for both of these two issues. Due to its huge fanout size and frequent switching nature, clock network is a major power consumer that can consume as much as 40% of entire chip power budget [1]. Moreover, the huge current drawn by clock network contributes considerably to power supply noise which is a major source of variations [2]. In addition to being a variation generator, clock network is also a vulnerable victim of variations since it needs to match delays of signals propagated through long distance in different regions.

Minimizing clock network size/wirelength can obviously reduce power consumption and power supply noise. Furthermore, small clock network size/wirelength implies less number of clock buffers which are very sensitive to both power supply noise and process variations [3]. Previous works on clock wirelength minimization are mostly concentrated on the clock routing stage [4-6] or exploiting certain clock skew flexibility [7]. Since clock routing is based on given clock sink (register) locations, the register placement may affect clock network wirelength greatly. For a poor register placement, the wirelength reduction from clock routing can be very limited. In prescribed skew designs [6], if a register has small clock signal delay target and is placed far away from the clock source, then registers with greater delay targets have to have long wire path to the source to satisfy the skew specification even when they are located close to the clock source. Therefore, placing registers in coherence with their delay targets may reduce wire detour and wirelength.

However, registers are strongly connected with logic cells and placing registers only for clock network may affect ordinary placement adversely. Indeed, the mainstream of previous works does not differentiate registers from logic cells in placement, and its objectives usually include minimizing total wirelength of signal nets [9], improving routability [10] and timing performance [11]. In [12], a clock skew aware placement post processing heuristic is suggested to improve timing performance. In a clock driven placement work [13], register locations from a global placement result are adjusted to match leaf nodes of an independently designed balanced binary tree. However, this technique is restricted to only zero skew designs and only evenly distributed clock sinks.

In order to solve the conflict between the clock network minimization and traditional placement goals, we propose the following techniques in a quadratic placement framework: (1) Manhattan ring based register guidance; (2) center of gravity constraints for registers; (3) pseudo pin and net; (4) register cluster contraction. These techniques can navigate registers toward locations in favor of clock network minimization without intrusive interference to traditional placement goals. As a result, register placement is integrated with traditional placement in a seamless manner. Experimental results show that our method can reduce clock network wirelength remarkably with negligible increase on signal wirelength and critical path delay.

2. BACKGROUND

2.1 Clock Network Layout

A clock network delivers a clock signal from the clock source to a set of clock sinks (registers) such that the clock skew constraints are satisfied. C*lock skew* is the difference between clock signal delays to two registers. If clock signal delays to register i and j are denoted as t_i and t_j, respectively, the skew specification $t_i - t_i$ can be obtained in a *skew scheduling* procedure [8] to meet the long path and the short path constraints. The skew specification can also be expressed as *delay target* to each register directly [6]. Given a register placement and skew specification, a clock network, which is usually a tree, is constructed through clock routing [4-6]. In clock routings, the location of the clock tree root is usually at the center among the registers and does not have to overlap with the clock source. The root node can always be connected to the clock source directly, i.e., clock routing algorithms do not rely on the location of the clock source.

2.2 Quadratic Placement

A hypergraph $\boldsymbol{G(V, E)}$ is given to describe a circuit with $\boldsymbol{V = \{ v_1, v_2, \ldots, v_{n+p}\}}$ indicating $\boldsymbol{n}$ movable cells and $\boldsymbol{p}$ fixed cells and $\boldsymbol{E = \{e_1, e_2, \ldots, e_m\}}$ representing the connections of the circuit. Every edge is assigned a weight $\boldsymbol{w(e)}$. The quadratic placement can be formulated as a *Linearly-constrained Quadratic Programming* (LQP) problem:

$$\boldsymbol{Min\ \{\Phi(X,Y) = 1/2X^TCX + d_x{}^TX + 1/2Y^TCY + d_y{}^TY \mid A^{(m)}X = u^{(m)};\ A^{(m)}Y = v^{(m)}\}} \qquad \boldsymbol{(1)}$$

The $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the x-coordinate and y-coordinate vectors of the movable cells. The $\boldsymbol{n \times n}$ matrix $\boldsymbol{C}$ represents the information of weighted edge connections among movable cells. The vector $\boldsymbol{d_x}$ and $\boldsymbol{d_y}$ indicate the connections between the movable cells and the fixed cells. The quadratic programming is performed iteratively together with recursive partitioning [9]. The $\boldsymbol{(u^{(m)},v^{(m)})}$ is the center coordinates of regions at the m-th partitioning level. The equality constraints in (1) force the *center of gravity* of the cells in each region to be at the center of their corresponding region. The partitioning is initially based on the location results of quadratic programming in previous iteration [9]. Then, KL-algorithm [14] based adjustment and terminal propagation [15] are performed. To save the runtime, the hybrid star/clique net model [16] is employed.

The proposed register navigation techniques are also integrated with a path based timing-driven quadratic placement [11]. For each timing critical path, the following Elmore delay based constraint is enforced together with solving the LQP of (1):

$$\sum_{path} f\left(R_{unit}, C_{unit}, C_{pin}\right) \sum_{e_{ij} \in path} \left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 \le T,$$

where R_{unit}, C_{unit} are the unit resistance and capacitance for wires, respectively, C_{pin} is the load capacitance of pins; e_{ij} connecting (x_i,y_i), (x_j, y_j), is an edge on the path and T is the clock cycle time.

3. PROPOSED TECHNIQUES

The basic idea of our approach is to match the distance of a register from the clock root, which does not necessarily overlap with the clock source, with its clock signal delay target. In other words, registers with large (small) delay targets are preferred to be placed far away (close to) the clock root so that the wire detours can be minimized. The major technique is to generate a set of Manhattan rings to indicate preferred distance for registers. Then, registers are navigated toward their corresponding Manhattan rings.

3.1 Manhattan Rings as Guides for Registers

A convenient yet effective gadget indicating the preferred distance from the clock root is a Manhattan ring [4], which is a square tilted by 45° (Figure 1). Every point on the Manhattan ring has the same Manhattan distance (Manhattan radius) from the center. For a register, if its preferred distance R from the clock root is known, we try to place it on a Manhattan ring centered at the clock root with radius R. Before clock routing, the clock root can be approximated by the center of the circuit layout. In prescribed skew designs, there could be a set of co-centered rings (Figure 1(a)) spanning most of the layout area so that each register has a good chance to find a ring nearby.

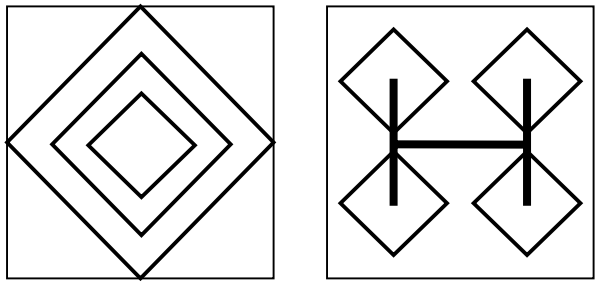

Figure 1. Manhattan rings for prescribed skew in (a) for zero skew in (b).

For zero skew designs, ideally we want to place registers on a single Manhattan ring since every register has the same delay target. However, attracting registers to a single Manhattan ring may leave large areas without registers and logic cells in those areas need huge wirelength to make connections with registers on the ring. Therefore, we propose to apply four (or more) smaller Manhattan rings driven by an H-tree as shown in Figure 1(b). The four small Manhattan rings span more area than a single big ring and therefore reduce the chance of dragging registers far away from their associated logic cells. Of course, the scheme of Figure

1(b) can also be applied for prescribed skew designs. Further, more rings can be applied depending on circuit size.

For prescribed skew designs, there is a single Manhattan ring at the beginning. This single Manhattan ring is called base ring. Along with the partitioning in the placement, the base ring is split into a set of rings with granularity corresponding to the partitioning level. For example, in Figure 2(a), there is one Manhattan arc *P* that spans region A, B and C at certain partitioning level *m*. The dashed lines indicate the partitioning lines. In the next partitioning level *m+1*, two new Manhattan arcs are split out from Manhattan arc *P* as shown in Figure 2(b). The two new Manhattan arcs span six smaller regions of level *m+1* adjacent to *P*.

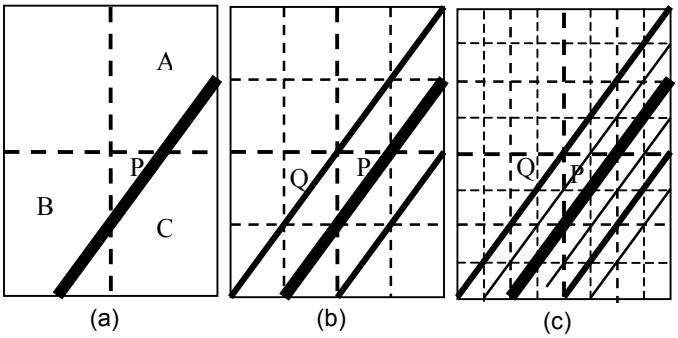

Figure 2. An example of splitting Manhattan arcs at different partitioning levels in (a), (b) and (c).

The splitting toward the clock root is called *inward splitting*. Similarly, the splitting away from the clock root is called *outward splitting*. For example, if the clock root is located in the southeast direction in Figure 2, then the generation of Manhattan arc *Q* is an outward splitting. It is not always necessary to perform splitting on both directions. In Figure 2(c), only the inward splitting is performed for Manhattan arc *Q*.

The Manhattan ring splitting can be applied for zero skew designs as well. In zero skew designs, even though we prefer that all registers are located on the four Manhattan rings as indicated in Figure 1(b), moving some registers toward the centers of the rings will not increase clock wirelength very much and may help on reducing signal wirelength. Therefore, we encourage inward splitting especially when the registers are overly crowded at the Manhattan rings.

The Manhattan ring splitting is performed from partitioning level $K - q$ to level K if there are K partitioning levels in total. In prescribed skew designs, the value of q is determined by the delay target difference. Large difference on delay targets suggests a large value of q so as to increase the scale of the spreading. For zero skew designs, the splitting is performed only at deep partitioning levels and the parameter q is an indication of the degree of spreading or aggressiveness on pushing registers toward preferred distance. A small value of q tends to drive registers closer to a few Manhattan rings. A large value of q may reduce impact to traditional placement goals.

We let the base Manhattan ring be a reference point by associating its radius R_{base} to the average delay targets t_{ave} for all registers. In order to cover the entire chip region evenly, we let the radius of the base Manhattan ring be $R_{base} \approx min(W, H)/4$ where W and H are the circuit width and height, respectively. The preferred distance between register i and the base Manhattan ring can be decided as:

$$\Delta_i = \alpha\sqrt{\frac{2(t_i - t_{ave})}{rc}} \qquad (2)$$

where r and c are wire resistance and capacitance, and α is a user specified parameter. Register i is preferred to be at a Manhattan ring of radius $R_{base} + \Delta_i$ if $t_i > t_{ave}$; or $R_{base} - \Delta_i$ otherwise. The maximum inward and outward splitting is decided by the preferred radius. Starting from partitioning level $K - q$, skew scheduling [8] needs to be performed at each level to generate the delay targets.

3.2 Navigating Registers toward Manhattan Rings

Since registers have strong connections to logic cells, placing registers at preferred Manhattan ring directly may adversely affect the traditional placement goals including signal net wirelength, routability and critical path delay. Therefore, registers need to be placed as close to their corresponding Manhattan rings as possible with the minimal negative impact to traditional placement goals. We propose two techniques that can navigate registers placement in a soft manner: (1) center of gravity constraint for registers and (2) pseudo pin and pseudo net.

3.2.1 Center of Gravity Constraint for Registers

In a GORDIAN-like placement [9], the center of gravity (COG) of the cells in a region is constrained to be at the center of the region in the subsequent quadratic programming. In our approach, we enforce separated COG constraints for registers. More specifically, we force the COG of registers in a region to be at the center of the Manhattan arc in the region. In Figure 3, after the initial quadratic placement, the entire region is quadrisected into four regions at the first partitioning level as shown in Figure 3(a). For the subsequent quadratic placement, we force the COG of registers in each region to be at the small filled circles. The COG of logic cells in each region is constrained to be at the center (star) of the region as before. When the quadratic placement and partitioning proceed, registers are spread out along the Manhattan ring as shown in Figure 3(b). The COG constraints can be applied to bi-sections as well.

The enforcement of the COG constraints can be illustrated through just the x-coordinate part. The x-coordinate vector for cells in a region can be decomposed as $\boldsymbol{X_f}$ for the registers and $\boldsymbol{X_g}$ for logic cells. Then the LQP at certain partitioning level m becomes (x-direction):

$$\begin{aligned} \textbf{Min}: \quad & \phi(X) = 1/2X^T CX + d_x^T X \\ \textbf{s.t.} \quad & A_f^{(m)} X_f = u_f^{(m)} \\ & A_g^{(m)} X_g = u_g^{(m)} \end{aligned} \qquad (3)$$

where $\boldsymbol{u_f}$ is the center of the Manhattan arc and $\boldsymbol{u_g}$ is the center of the region. The two linear equality constraints can be merged as:

$$A^{(m)} X = u^{(m)}, \qquad (4)$$

where $A^{(m)} = \begin{bmatrix} A_f^{(m)} & 0 \\ 0 & A_g^{(m)} \end{bmatrix}$, $u^{(m)} = \begin{bmatrix} u_f^{(m)} \\ u_g^{(m)} \end{bmatrix}$.

Thus, this new formulation retains the same number of variables and the same matrix size as traditional quadratic placement [9]. Since we only enforce the center of gravity, not the exact locations of registers, to be on the Manhattan arc, registers are only attracted to the preferred distance softly and there is no intrusive disturbance to traditional placement goals. This COG technique is primarily applied at early stages of partitioning where the number of registers in each region is relative large. At deep partitioning levels, the number of registers in each region becomes small and the COG constraint becomes less soft. In the extreme case where there is only one register in a region, the COG constraint is equivalent to force the register on the Manhattan arc and such constraint is too rigid to compromise with traditional placement goals. For deep partitioning levels, we switch to another soft navigation technique – pseudo pin/net.

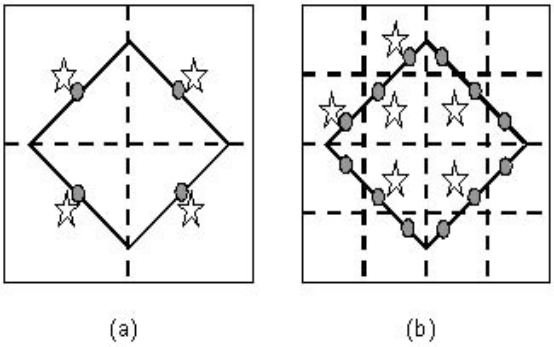

Figure 3. Dashed lines indicate partitioning in placement: level 1 in (a) and level 2 in (b). The small circles are the center of gravity of the registers in each region. The stars are the centers for each region.

3.2.2 Pseudo Pin and Pseudo Net

In the last several iterations of the placement, the COG technique is not applied. Instead, a pseudo pin and a pseudo net are added for each register. The pseudo pin is on the corresponding Manhattan ring and has the minimum distance to the register. The pseudo net connects the register with the pseudo pin. This is illustrated in Figure 4. The pseudo pin and pseudo net are reflected in the vectors d_x and d_y of the LQP formulation (1). For example, the x-direction d_x is changed as:

$$d_x = \begin{bmatrix} d_f \\ d_g \end{bmatrix} \quad \rightarrow \quad d_x' = \begin{bmatrix} d_f + d'_f \\ d_g \end{bmatrix} \qquad (5)$$

where d_f and d_g indicate connections between fixed cells with registers and logic cells, respectively. The additional connections due to the pseudo pin and pseudo net are included in the vector d'_f.

The pseudo pin/net attempts to drag the register closer to the Manhattan ring in the quadratic placement. Again, this technique navigates registers in a soft manner with compromise with traditional placement goals. The compromise can be tuned by changing the weight of this pseudo net. A large weight implies a strong force of pulling the register toward the Manhattan ring.

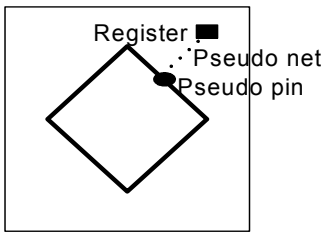

Figure 4. A pseudo pin indicated by filled circle on the Manhattan ring and a pseudo net indicated by the dotted line.

We apply different net weighting schemes according to the delay targets of their corresponding registers. Pseudo nets for registers with small (large) delay targets are assigned with heavy (small) weight. If a register with large delay target is placed closer with clock root, the impact to wirelength may be very limited. This can be shown through the example in Figure 5. On the other hand, placing a register with small delay target far from the clock root may force all root-sink path length to increase to satisfy skew constraints and therefore is more harmful.

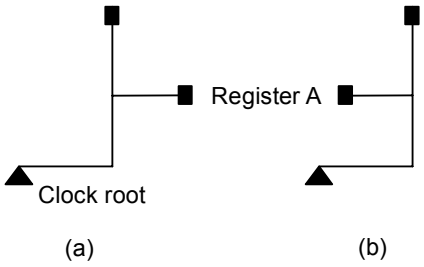

Figure 5. Moving register "A" closer to the clock root from (a) to (b) does not change the wirelength.

In the partitioning adjustment [14], the pseudo nets are also counted in the min cut estimation. Therefore, a register has a good chance to be partitioned to the same region as its corresponding pseudo pin.

3.3 Register Cluster Contraction

If registers are placed close to each other, the clock routing wirelength can also be reduced due to wire path sharing. Thus, we cluster registers with similar delay target and in the same partition region by inserting clique-based pseudo nets among them. The clique-based pseudo nets change the edge connection entries of related registers in the matrix C of Equation (1). For example, the matrix entry for two registers *R1* and *R2* is changed from zero to non-zero if a pseudo net connects them. The registers connected by the same pseudo net can be pulled close to each other after solving the LQP (1) in subsequent iterations.

4. EXPERIMENTAL RESULTS

The proposed method is implemented in C++ and the experiments are performed on a SUN V880 workstation. Table 1 gives the characteristics of the DEF/LEF benchmarks. In these benchmark circuits, 2.7%~21.4% of cells are registers. The skew scheduling is obtained through the method of [8]. Prescribed skew and zero

skew routings are generated through clock routers of [6] and [5], respectively. The wirelength of signal nets are estimated through half perimeter model. Our placement method is compared with a wirelength driven placer [9] and a timing driven placer [11].

Table 1. The characteristics of benchmarks.

Case	#cell	#net	#register	Register %
Test	4596	4674	126	2.7%
Te2	9192	7195	252	2.7%
F1	6090	7513	1302	21.4%
Te4	18384	14341	504	2.7%
F3	62050	62893	7681	12.4%

The experimental results for the traditional wirelength-driven placement [9] are summarized in Table 2. In Table 2, clock tree wirelength and total signal net wirelength are shown for both prescribed skew design and zero skew design. The corresponding results from our method are listed in Table 3. It can be seen that our method can reduce clock tree wirelength by 16%~33% with an average of 0.09% increase on signal wirelength. Actually, the increase on signal wirelength is never greater than 0.5%. Even the total wirelength including both clock and signal net is decreased as well. Therefore, our method indeed achieves our goal of reducing clock net wirelength significantly with negligible impact on signal wirelength. The CPU times for all these methods are also listed.

There are three major techniques we proposed: (1) COG: center of gravity constraints; (2) PPN: pseudo pin and net; (3) RCC: register cluster contraction. Therefore, the results in Table 3 are from COG+PPN+RCC. In order to see the effect of these techniques separately, we show the clock wirelength change from COG, COG+PPN and COG+RCC with respect to the result of [9] in Table 4. For F1 and F2, which have large proportion of registers, the COG technique alone is not adequate and PPN is very necessary. For the other benchmarks, COG alone can make significant improvement as the number of registers is relatively small. In all of these cases, RCC plays an auxiliary role for further improvement.

Our register navigation techniques are integrated with a timing driven placer [11]. The maximum critical path delay from [11] and our method on prescribed skew and zero skew design are shown in Table 5. We can see that our register navigation techniques can retain timing performance in the placement.

5. CONCLUSIONS

Clock network minimization can alleviate the power and variation problem. Distinguished from previous works which are mostly on clock routing, we propose to reduce clock network wirelength by navigating register locations in cell placement. Several techniques are proposed to solve the conflict between clock network minimization and traditional placement goals. Experimental results on benchmark circuits show that our techniques can reduce clock network wirelength for both zero and prescribed skew designs remarkably with negligible increase on signal wirelength and critical path delay.

6. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (NSFC) 60476014 and 90407005, the Specialized Research Fund for the Doctoral Program of Higher Education: SRFDP-20020003008, and the Hi-Tech Research & Development (863) Program of China 2004AA1Z1460. This work is also partially supported by DAC Graduate Scholarship and SRC under contract 2004-TJ-1205.

7. REFERENCES

[1] D. E. Duate, N. Vijaykrishnan and M. J. Irwin, "A clock power model to evaluate impact of architectural and technology optimization," in *IEEE TVLSI*, 10(6): 844-855, Dec. 2002.

[2] S. Zhao, K. Roy and C.-K. Koh, "Estimation of inductive and resistive switching noise on power supply network in deep sub-micron CMOS circuits," in Proc. IEEE ICCD, pp. 65-72, 2000.

[3] S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena and C. Guardiani, "Analysis of the impact of process variations on clock skew," in IEEE Transactions on Semiconductor Manufacturing, 13(4): 401-407, Nov. 2000.

[4] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese and A. B. Kahng, "Zero skew clock routing with minimum wirelength," in IEEE Transactions on Circuits & Systems II: Analog & Digital Signal Processing, 39 (11): 799-814, Nov. 1992.

[5] J. Cong, A. B. Kahng, C.-K. Koh and C.-W. A. Tsao, "Bounded-skew clock and Steiner routing," in ACM TODAES, 3(3): 341-388, Jul. 1998.

[6] R. Chaturvedi and J. Hu, "A simple yet effective merging scheme for prescribed-skew clock routing," in Proc. IEEE ICCD, pp. 282-287, 2003.

[7] C.-W. A. Tsao and C.-K. Koh, "UST/DME: a clock tree router for general skew constraints," in Proc. IEEE/ACM ICCAD, pp. 400 – 405, 2000.

[8] R. B. Deokar and S. S. Sapatnekar, "A graph-theoretic approach to clock skew optimization", in Proc. IEEE ISCAS, pp.1.407- 1.410, 1994.

[9] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," in IEEE TCAD, 10(3): 356-365, Mar. 1991.

[10] M. Wang and M. Sarrafzadeh, "Congestion minimization during placement," in Proc. ACM ISPD, pp. 145- 150, 1999.

[11] W. Hou, X. Hong, W. Wu and Y. Cai, "A path-based timing-driven quadratic placement algorithm," in Proc. IEEE/ACM ASP-DAC, pp.745-748, 2003.

[12] Y. Liu, X. Hong, Y. Cai, W. Wu, "CEP: A clock-driven ECO placement algorithm for standard-cell layout," in Proc. International Conference on ASIC, pp. 118-121, 2001.

[13] N. Venkateswaran and D. Bhatia, "Clock-skew constrained placement for row based designs," in Proc. IEEE ICCD, pp. 219-220, 1998.

[14] B. Kernighan and S. Lin. "An Efficient Heuristic Procedure for Partitioning of Electrical Circuits". Bell System Technical Journal, pp. 291-307, 1970.

[15] A. E. Dunlop and B. W. Kernighan, "A procedure for placement of standard-cell VLSI circuits," IEEE TCAD, vol. CAD-4, pp. 92-98, Jan. 1985.

[16] N. Viswanathan and C.-N Chu, "FastPlace: Efficient Analytical Placement using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model," in Proc. ACM ISPD, pp. 26- 33, 2004.

Table 2. The conventional wirelength-driven placement [9] results.

Test-case	Prescribed skew		Zero skew		CPU (sec)
	Clock WL	Signal WL	Clock WL	Signal WL	
Test	12540	704416	11192	704416	18
Te2	21930	1422462	21666	1422462	36
F1	79372	2387391	80568	2387391	31
Te4	54366	3188446	47766	3188446	80
F3	487833	10904591	434399	10904591	347

Table 3. Our clock-driven placement results.

Test-case	Prescribed skew				Zero skew			
	Clock WL	Clock WL change	Signal WL	CPU (sec)	Clock WL	Clock WL change	Signal WL	CPU (sec)
Test	8439	-32.7%	704467	23	8936	-20.1%	704431	20
Te2	16945	-22.7%	1425685	58	16385	-24.4%	1425879	42
F1	66416	-16.3%	2389333	224	67562	-16.1%	2389177	31
Te4	38262	-29.6%	3189004	110	39987	-16.3%	3188450	89
F3	406173	-16.7%	10918332	1448	363108	-16.4%	10918478	403
Average signal WL change			0.091%		Average signal WL change		0.089%	
Average total WL change			-0.47%		Average total WL change		-0.32%	

Table 4. The clock wirelength change of only COG, COG + PPN and COG + RCC.

Test-case	COG		COG + PPN		COG + RCC	
	P-skew	Zero skew	P-skew	Zero skew	P-skew	Zero skew
Test	-22.5%	-16.1%	-27.2%	-18.0%	-24.2%	-17.3%
Te2	-19.4%	-16.8%	-24.3%	-23.0%	-21.9%	-17.4%
F1	-2.0%	-2.2%	-12.8%	-11.4%	-4.2%	-4.8%
Te4	-19.1%	-13.1%	-29.2%	-15.6%	-22.9%	-13.9%
F3	-2.4%	-2.7%	-12.9%	-15.9%	-4.3%	-4.4%

Table 5. The timing-driven placement results.

Test-case	Maximum critical path delay (ns)			CPU (sec)		
	Timing-driven [11]	Our prescribed skew	Our zero skew	Timing-driven [11]	Our prescribed skew	Our zero skew
Test	1.06	1.07	1.07	83	131	129
Te2	1.67	1.68	1.67	191	293	224
F1	2.65	2.65	2.65	99	335	121
Te4	0.75	0.77	0.76	451	622	537
F3	6.48	6.48	6.49	1681	3031	1774