Register Placement for High-Performance Circuits

Mei-Fang Chiang*, Takumi Okamoto†, and Takeshi Yoshimura*
*Graduate School of IPS, Waseda University, Kitakyushu 808-0135, Japan
†NEC Corporation, Kanagawa 211-8666, Japan

***Abstract*—In modern sub-micron design, achieving low-skew clock distributions is facing challenges for high-performance circuits. Symmetric global clock distribution and clock tree synthesis (CTS) for local clock optimization are used so far, but new methodologies are necessary as the technology node advances. In this paper, we study the register placement problem which is a key component of local clock optimization for high-performance circuit design along with local clock distribution. We formulate it as a minimum weighted maximum independent set problem on a weighted conflict graph and propose a novel efficient two-stage heuristic to solve it. To reduce the graph size, techniques based on register flipping and Manhattan circle are also presented. Experiments show that our heuristic can place all registers without overlaps and achieve significant improvement on the total and maximal register movement.**

I. Introduction

As the technology node continues decreasing and design complexity keeps increasing, there are more challenging problems imposed on circuit design. Especially, synchronous clocking is a dominant strategy for circuits operating with the clock frequency exceeding 1 GHz, which are called high-performance circuits. Any uncertainty in the clock arrival time may reduce the performance, yield, and even cause functional errors. Thus, a high-speed and reliable clock network is required for high-performance circuits. Also, the clock network is a main part of power consumption that can consume 40% of the total power [5] due to its large fanout and rapid switching frequency. To minimize the power, clock uncertainty, and skew which is the difference between clock delays of two registers, a special clock network structure is widely used for current high-performance circuits [13].

The clock structure for high-performance circuits consists of global and local parts. The global clock structure distributes clocks from a root to certain divided areas, and leaves of the distribution are called local clock drivers (LCD). Then, the local clock structure distributes LCDs to registers. The topology used in the global clock is usually balanced/symmetric. The *grid* and *tree* structures are generally used. In the DEC Alpha series of circuits, the gridded clock distribution is used to guarantee low local skew and simple implementation in the early stage [1] [2] [5]. The Symmetric H-tree clock structure is used in IBM S/390 for trivially zero skew [17]. Recently, most high-performance circuits use a mix of grid and tree clock topologies and utilize advantages of both [14]. For example, the clock structure used in IBM Power4 [16] and Power6 [3] consists of many buffered symmetric trees driving a single clock grid. Figure 1 gives an example clock distribution.

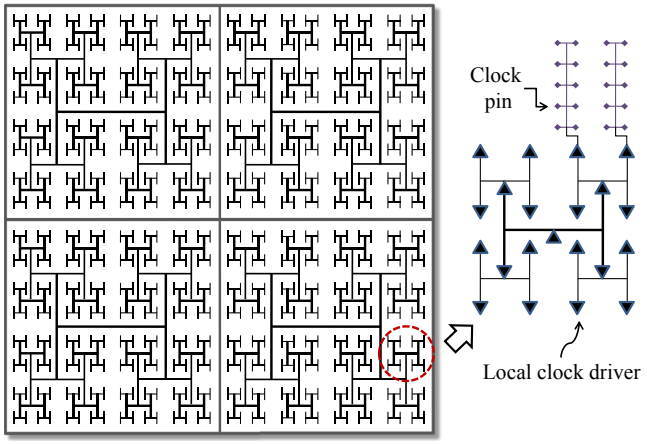

Fig. 1. An example clock structure of high-performance circuits.

After global clock structure design and gate level placement, general clock tree synthesis (CTS) is used for local clock design so far. However, in deep sub-micron design, other special strategies are necessary. See Figure 2 for example. In the previous design flow, cell placement including registers is performed before local clock design. Since CTS is designed according to clock sinks (registers) which are given by the placement, register positions can affect the clock design greatly. There have been some register placement/clamping techniques proposed to facilitate local clock design. While for high-performance circuit design with specified requirements, a local clock network is designed along with gate level placement. By given designated/estimated clock pins, registers are legalized to fit the requirement. Thus, with gate level placement and local clock pin information, registers are required to move to positions of clock pins for ideal zero-skew or positions near clock pins for low-skew as possible, along with constraints and objectives such as power (clock area) and timing (wirelength).

As mentioned above, to facilitate local clock design, there have been previous works about register clamping/placement for general ICs. In [4], a power-aware placement method that performs activity-based register clustering and activity-based net weighting is proposed. The activity-based register clustering technique groups registers into clusters and clumps

This research was supported by Waseda University Global COE Program "International Research and Education Center for Ambient SoC" sponsored by MEXT, Japan

978-3-9810801-5-5/DATE09 © 2009 EDAA

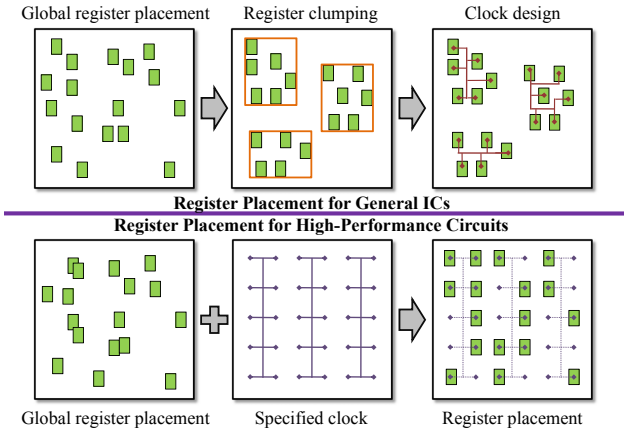

Fig. 2. Register placement for general and high-performance circuit designs.

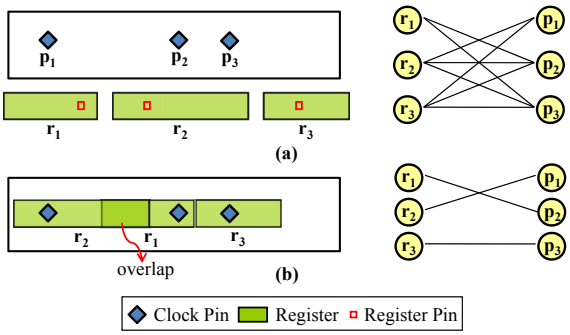

Fig. 3. Linear assignment formulation. (a) A problem and its corresponding assignment graph. (b) An assignment results in placement with an overlap.

registers in the same cluster closer to each other to reduce the capacitance of the clock tree. To navigate register placements to desired locations for further clock routing wirelength and power minimization, some techniques are proposed and carried out in the quadratic placement [9] [10]. The authors used Manhattan-ring based register guidance, center of gravity constraints, and pseudo pins/nets for register navigation. Also, in [15], the authors proposed register clustering and clumping according to registers' activity patterns in a minimal-cut based placer to reduce the clock tree power based on clock gating.

Although there are many effective previous works related to register placement, they are not suitable for high-performance circuits in deep sub-micron design. To our best knowledge, there is no previous work regarding register placement of high-performance circuits with specified clock structures, except for an analogous one. Okamoto *et al.* [12] proposed a design methodology for NEC Electronics' structured ASIC with an embedded clock structure. They carried out register alignment in the final step of their embedded clock-aware placement. Registers are aligned to embedded slots by linear assignment with register movement minimization. However, register placement with respect to clock pins for various sizes of registers cannot be formulated optimally by linear assignment. Figure 3 shows an example that the linear assignment result leads to an incorrect register placement where an overlap exists.

Therefore, to achieve low-skew clock distribution for high-performance circuits in sub-micron design, we studied the register placement which is a key component of local clock structure optimization. First, we formulate it as a minimum weighted maximum independent set (MIS) problem on a weighted conflict graph constructed from the global register placement. Then, we propose a novel two-stage heuristic to solve the problem based on an iteration method. Since MIS problem for a general graph is nondeterministic polynomial-complete (NPC) [7], we also give two techniques to reduce time of the graph construction and of solving MIS. The experimental results show that our heuristic performs well.

The rest of this paper is organized as follows: Section 2 formulates the register placement problem for high-performance circuits. Section 3 describes the weighted conflict graph including graph construction and reduction. Section 4 gives our two-stage heuristic for solving the MIS problem. Experimental results are reported in Section 5, and conclusions are given in Section 6.

II. Problem Formulation

After global clock design and global cell (gate level) placement including registers, we have clock pin information and a resulting register placement. As for clock pins, the topology can vary corresponding to each circuit's clock design. In some cases, they might be estimated points in the early stage to guide the following clock optimization process and will be adjusted in the late stage. From the global cell placement result, information of register placement having overlaps and obstacles such as pre-placed blocks is extracted. As for registers, sizes can vary according to their design. For example, scan flip-flops (SFF) which have scan capability for testing are larger than non-scan ones. Flip-flops having specific functions are usually larger than normal ones. Here, we set registers with a fixed height and variant widths for a simple and practical discussion. Problem with variant-height registers can be easily extended. For each register, a *register pin* is a pin (clock sink) for connection to the clock source pin. The register pin may be somewhere inside a register according to its design. The problem of register placement for high-performance circuits is formulated as follows.

Problem 1: Given a global placement result of registers and fixed-positioned clock pin points, legalize registers with total/maximum movement as small as possible, such that each register pin is exactly placed on the position of a clock pin and no overlaps between registers remain.

Here, we minimize the total/maximum movement of registers to reduce adverse impact on goals of global cell placement, such as net wirelength and timing, and to facilitate the following detail placement of logic cells. Because logic cells are usually 2 – 4 times smaller than registers, it is possible to incrementally optimize them in terms of slight movement of registers after register placement. Even though registers usually have strong connections to logic cells, incremental placement of logic cells afterward can be easily achieved.

A. Minimum Weighted Maximum Independent Set Formulation

For the register placement problem of high-performance circuits, we formulate it as a minimum weighted maximum independent set (MIS) problem on a weighted conflict graph. A *weighted conflict graph* $G = (V, E, W)$ is a weighted undirected graph constructed from a given global register placement and fixed-positioned clock pins. Vertex $v_{i,j} \in V$ exists if and only if a placement of register r_i on the position of clock pin p_j is feasible. An edge $(v_{i,j}, v_{i',j'}) \in E$ exists if and only if $i = i'$, $j = j'$, or vertices $v_{i,j}$ and $v_{i',j'}$, where $i \neq i'$ and $j \neq j'$, result in overlapping placements. Weight $w_{i,j} \in W$ for $v_{i,j}$ is defined as the movement distance of r_i from its original position to p_j. After constructing the conflict graph, the register placement problem can be reduced to a minimum weighted maximum independent set problem defined as follows.

Problem 2: Given a global placement result of registers and fixed-positioned clock pin points, find a minimum weighted maximum independent set V_M from the weighted conflict graph constructed from given information. The register placement is to place register r_i on clock pin p_j according to each vertex $v_{i,j} \in V_M$, where the weight of the independent set $W(V_M) = \sum_{\forall v_{i,j} \in V_M} w_{i,j}$ representing the total/maximal movement is minimized.

Proof: The minimum weighted MIS V_M of weighted conflict graph $G = (V, E, W)$ is a largest vertex set of G such that $\forall v_{i,j}, v_{i',j'} \in V_M, (v_{i,j}, v_{i',j'}) \notin E$. Since an edge $(v_{i,j}, v_{i',j'})$ means that a conflict exists if register r_i is placed on clock pin p_j and register $r_{i'}$ is placed on clock pin $p_{j'}$ simultaneously, V_M of G is a set representing a maximum number of registers placed without conflicts. Minimum weight stands for that the total/maximal movement is minimized for the placement. ■

III. Weighted Conflict Graph

Given a global placement result and fixed-positioned clock pin points, we have position information of registers, clock pins, and obstacles. To facilitate the position transformation and conflict graph construction, the *segment* representation is used. Segment $s_i = [x_1, x_2]$ for clock pin p_i is constructed by starting from p_i and extending bi-direction horizontally until it meets obstacles or boundaries. For one segment, there may be several clock pins on it. Figure 4(b) gives an example segment representation transformed from the given placement shown in Figure 4(a). We could see obviously that segment representation shows not only feasible regions for register placement, but also adjacent clock pins' relationship. The weighted conflict graph, which is introduced as follows, is constructed in terms of segments.

A. Graph Construction

The construction of weighted conflict graph $G = (V, E, W)$ consists of vertex, edge, and weight constructions. For the vertex set construction, we are going to find all feasible candidate placements of every register. First, for each clock pin p_j on segment s_k, we calculate distance d_j^1 and d_j^2 from

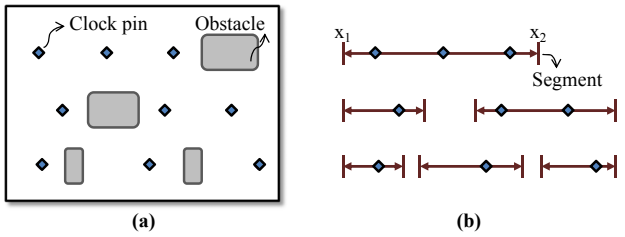

Fig. 4. Segment representation.

p_j to x_1 and x_2 of s_k. Also, for pin t_i in each register r_i, we calculate width l_i^1 and l_i^2 from t_i to the left and right boundary of r_i. Thus, $v_{i,j} \in V$ exists if and only if $l_i^1 \leq d_j^1$ and $l_i^2 \leq d_j^2$, which means register r_i can be feasibly placed on the position of clock pin p_j.

For every two vertices $v_{i,j}$ and $v_{i',j'}$, an edge $(v_{i,j}, v_{i',j'})$ exists if and only if one of the following conditions is satisfied: (1) $i = i'$ because there is only one placement for one register in the final result; (2) $j = j'$ because one clock pin can only be assigned for one register; (3) there is an overlap between feasible placements $v_{i,j}$ and $v_{i',j'}$. To check the overlap, only feasible placements on clock pins of the same segment need to be processed. For $v_{i,j}$ and $v_{i',j'}$, assume that p_j and $p_{j'}$ are on the same segment and p_j is left to $p_{j'}$, an overlap exists if and only if $l_i^2 + l_{i'}^1 > |x_{p_j} - x_{p_{j'}}|$.

After the construction of vertex and edge sets, a weight $w_{i,j}$ is introduced for each vertex $v_{i,j} \in V$. The weight is defined as the movement of a register to the power of α. The movement is the distance between the register pin's initial position and the clock pin position. The weight definition is given by

$$w_{i,j} = (|x_i - x_j| + |y_i - y_j|)^{\alpha}, \tag{1}$$

where (x_i, y_i) and (x_j, y_j) are coordinates of the register pin of r_i and clock pin p_j. The parameter α is defined as 1 and 2 for total and maximal movement minimization, respectively. Figure 5(a) shows an example global register placement and corresponding coordinates; Figure 5(b) gives its weighted conflict graph.

The maximum size of a weighted conflict graph can consist of mn vertices, where m and n are the number of registers and clock pins. Since the MIS problem for a general graph is

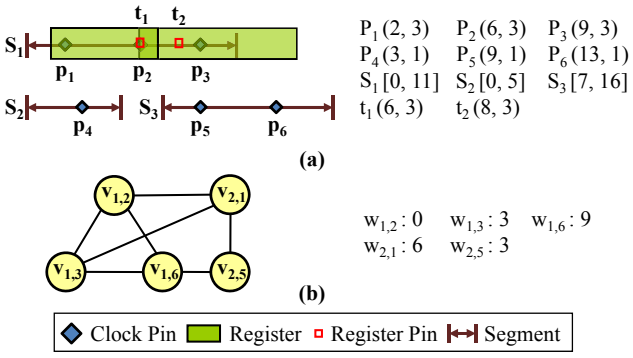

Fig. 5. Weighted conflict graph.

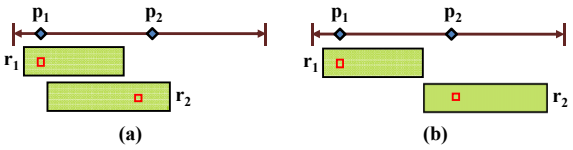

Fig. 6. Register flipping.

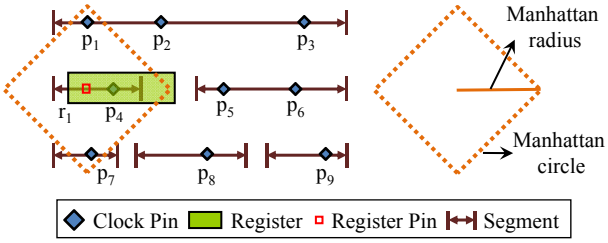

Fig. 7. Manhattan circle.

NPC [7], to reduce the graph-construction and MIS- solving time, we propose two graph reduction techniques. Details are introduced in the following subsection.

B. Graph Reduction Techniques

To reduce the size of a conflict graph to speed-up graph construction and MIS solving times, we apply register flipping and Manhattan ring during the graph construction algorithm.

For a register, its pin may be near the left or right boundary according to its structure/function. If there are some register pins near the right boundary while others near the left, the probability of getting the placement with overlaps may get higher. Then, the edge size of the conflict graph will increase, which makes the MIS solving time longer. Thus, we flip registers horizontally to keep regularity, that is, all registers have its pin near to the left or right boundary, to reduce placement overlaps. Figure 6(b) is the flipping result of Figure 6(a) which has an overlap. Since we flip the whole register instead of changing the register pin's position, it will not affect the register's functionality. The effect of register flipping on the graph reduction depends on the clock structure and registers' design.

During the graph construction, every register is tried to fit on every clock pin to get feasible placements. However, it makes the number of vertices huge if the number of clock pins is much more than that of registers, and includes feasible placements with a far movement. Since our objective is to find a placement with the total/maximal movement minimized, we use Manhattan circle [9] to restrict the vertex construction. Manhattan circle is a $45°$-tilted square with the same Manhattan distance, called Manhattan radius R_M, from the center to any point on it. By using Manhattan circle centered on a register pin, we don't place the register on clock pins outside the circle during the vertex construction. Thus, vertices with far movements are excluded in the reduced vertex set V' which is defined as follows.

$$V' = \{v_{i,j} | \forall v_{i,j} \in V,\ (|x_i - x_j| + |y_i - y_j|) \leq R_M\}$$

Figure 7 shows an example Manhattan circle. In it, vertices are created only if register r_1 can be placed feasibly on those clock pins (p_1, p_4, and p_7) inside the Manhattan circle.

IV. Two-Stage Heuristic for the MIS Problem

It is known that the MIS problem for a general graph is NP-complete, thus it is hard to solve it optimally in polynomial time and an efficient algorithm is desired. Therefore, we propose a two-stage heuristic to solve the MIS problem. Figure 8 gives our heuristic flow and an example. In the first stage, we partition a design into many fixed-area subregions. In each subregion, construct a weighted conflict graph and solve it by an iteration-based MIS heuristic. For the second stage, we construct a weighted conflict graph of the whole design for registers unassigned in the first stage. The register placement result is the union of minimum weighted MISs solved during two stages.

For each weighted conflict graph $G = (V, E, W)$, we solve the minimum weighted MIS problem on it based on an iteration method, which is proposed by Lee *et al.* [8] and performs effectively. We show the heuristic in the following steps.

- Step 1: Construct a priority queue Q containing $v_i \in V$ by 3 keys: the size of register r_i, movement (weight) of vertex v_i, and degree (# of incident edges) of v_i. Vertex v_i has a higher priority if it has a larger size of corresponding r_i, a smaller movement, and a smaller degree.
- Step 2: Extract first k vertices from Q as the vertex set of subgraph $G' = (V', E', W')$, where for all $v_i, v_j \in V'$, $(v_i, v_j) \in E'$ if $(v_i, v_j) \in E$, and for all $v_i \in V'$, $w_i \in W'$ if $w_i \in W$. Get sub-solution S' by solving the minimum weighted MIS problem on G'.
- Step 3: Update graph G by deleting vertex $v_i \in S'$ and vertices adjacent to v_i. Priority queue Q is also updated by new G. Sub-solution S' is included into the minimum weighted MIS S of graph G.
- Step 4 : Repeat step 2 and step 3 until the graph or the queue is empty to get the final minimum weighted MIS

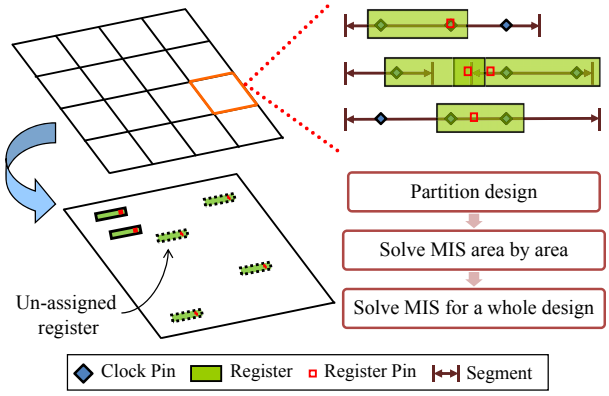

Fig. 8. Two-stage heuristic.

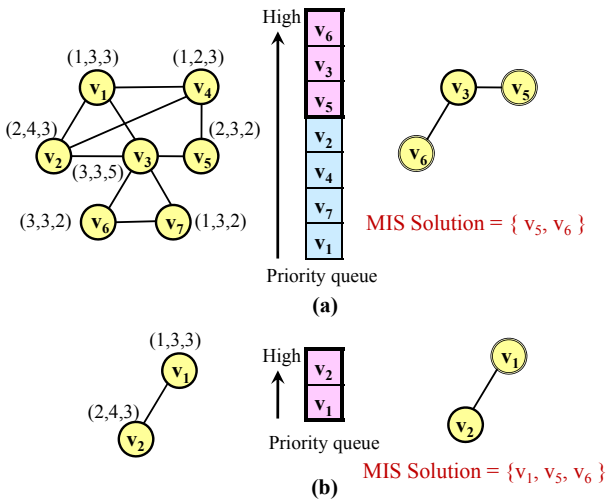

Fig. 9. Example of iteration-based method.

S of graph G.

By the iteration-based method, MIS problem can be solved more efficiently. Meanwhile, the criterion to extract subgraphs, which affects the solution optimality, is important. We give vertices corresponding to larger-size registers higher priority because smaller registers which are flexible for placement can be placed later. Since we target on minimizing the movement of the register placement, vertices of small movements are also given higher priority for processing. As for the degree, since vertices with smaller degrees can cause fewer vertices deleted during the graph update, nearly optimal solution space can be guaranteed. Figure 9 gives an example of the iteration-based method. In Figure 9(a), the priority queue is constructed according to the graph in the left side. Three vertices v_6, v_3, and v_5 are extracted from the queue to form a subgraph shown in the right side. After getting the MIS solution $\{v_5, v_6\}$ of the subgraph, the graph is updated and shown in the left side of Figure 9(b). Finally, subgraph containing v_1 and v_2 is solved and the MIS solution of the whole graph is $\{v_1, v_5, v_6\}$.

V. Experimental Results

The two-stage heuristic for the MIS problem is implemented by using C++ programming language on a Linux-based workstation with 2.66-GHz processor and 2-GB memory. We use a quick almost exact maximum weight clique/independent set solver, named qualex-ms [18], as the MIS solver. To get a minimum weighted MIS, we need to transform the vertex weight w_i to $INT\_MAX - w_i$ by using the maximum weight MIS solver. The parameter k for the sub-graph extraction in the first/second stage is set to 300/1000 which is found well-performed in terms of movement minimization after trying many numbers.

We perform the two-stage heuristic on industrial test cases which are listed in Table I. Global cell placement are done by the commercial tool. In the table, "Name" gives names of test cases, "Size" gives the dimension of each test case in nanometer square, "#Clk. Pin" gives the number of clock pins,

TABLE I
Test cases.

Name	Size (nm^2)	#Clk. Pin	#Reg.	#Seg.
Case1	4799760 x 316800	26875	3076	7527
Case2	4799760 x 316800	26875	3076	7527
Case3	4799760 x 316800	13818	3620	7527
Case4	4799760 x 316800	13818	3620	7527
Case5	5999280 x 4815360	237250	13733	139483
Case6	2699760 x 1866240	32193	21262	24971

"#Reg." gives the number of registers, and "#Seg." shows the number of segments constructed from the design. In Case1 and Case2, the same design and clock structure are used while the size of registers is different. The size of 83.75% of registers in Case1 is $17080nm$ while is $12600nm$ in Case2. The same, Case3 and Case4 have 22.6% of registers with different sizes which is $17080nm$ in Case3 and $12600nm$ in Case4.

For comparison, we implement a bipartite matching based heuristic called BMS on a Linux-based workstation with 2.8-GHz processor and 32-GB memory. (Runtime of BMS is normalized in Table II and Table III by 2.8/2.66 due to different workstations.) Since a general bipartite matching method may result overlaps, we implement BMS by some extensions based on ideas in [12], to achieve placing registers without overlaps. In terms of placing all registers successfully, the comparison can show the performance on minimizing total/maxial movement. The heuristic BMS is implemented as follows. First we group registers of the same size together and sort groups of registers according to the size. Then we process registers group by group in a decreasing order of sizes. During each process, we create slots on clock pins with the size equal to registers. Then, perform minimum weighted bipartite matching for registers and slots. The weight on the edge connecting register r_i and slot t_j is defined as $\alpha \times$(movement from r_i to $t_j)^\beta$, where α and β are user-defined parameters. We set β to 1.5 – 2.0 for total/maximal movement minimization. After processing all group of registers, simulated annealing is used for further optimization.

Table II shows results of the register placement with total movement minimization compared with BMS. In the table, "#Total Reg." gives the number of registers in each case, "#Com. Reg" shows the number of registers placed successfully, "Com. Rate" is equal to "#Com. Reg" divided by "#Total Reg.", and "Total Move." gives the total movement of registers after the placement. As shown in the table, both two heuristics placed registers 100% without overlaps whereas our heuristic reduces total movement by 45% over BMS on average. The increase of runtime is as expected because the MIS formulation is NP-complete, and it is acceptable for industrial cases.

In Table III, results of register placement with maximal movement minimization compared with BMS are listed. "Max Move." gives the maximal movement of registers after the placement. On average, our heuristic reduces the maximal movement by 19% over BMS. Our heuristic even improves

TABLE II
COMPARISON FOR REGISTER PLACEMENT WITH TOTAL MOVEMENT MINIMIZATION.

Test Case	Info.	BMS				Two-Stage Heuristic (Ours)			
	#Total Reg.	#Com. Reg.	Com. Rate (%)	Total Move. (μm)	Total time (s)	#Com. Reg.	Com. Rate (%)	Total Move. (μm)	Total time (s)
Case1	3076	3076	100	202472	25.26	3076	100	193400	141.41
Case2	3076	3076	100	25196	16.84	3076	100	27638	138.37
Case3	3620	3620	100	335719	23.16	3620	100	289826	107.20
Case4	3620	3620	100	282060	28.42	3620	100	276968	66.80
Case5	13733	13733	100	716431	121.05	13733	100	735534	218.22
Case6	21262	21262	100	3629094	790.53	21262	100	2047709	630.00
Comp. (Avg.)		-	100	1.45	0.77	-	100	1.00	1.00

TABLE III
COMPARISON FOR REGISTER PLACEMENT WITH MAXIMAL MOVEMENT MINIMIZATION.

Test Case	Info.	BMS				Two-Stage Heuristic (Ours)			
	#Total Reg.	#Com. Reg.	Com. Rate (%)	Max Move. (μm)	Total time (s)	#Com. Reg.	Com. Rate (%)	Max Move. (μm)	Total time (s)
Case1	3076	3076	100	229.00	25.26	3076	100	154.99	198.01
Case2	3076	3076	100	131.00	16.84	3076	100	43.97	202.74
Case3	3620	3620	100	243.00	23.16	3620	100	343.99	141.00
Case4	3620	3620	100	280.00	28.42	3620	100	272.99	88.15
Case5	13733	13733	100	412.00	121.05	13733	100	284.99	408.81
Case6	21262	21262	100	654.00	790.53	21262	100	539.99	770.46
Comp. (Avg.)		-	100	1.19	0.56	-	100	1.00	1.00

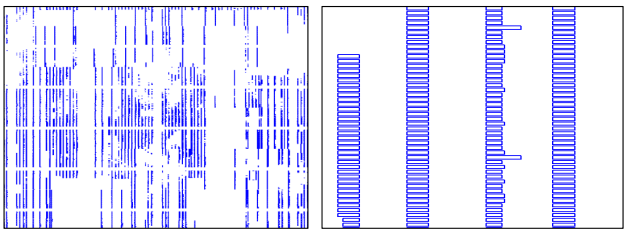

Fig. 10. Register placement result of Case6.

66.44% of maximal movement in Case2. As for the runtime, bipartite matching enjoys the polynomial-time complexity while our heuristic needs more time to achieve higher solution quality. Figure 10 shows placement results for Case6 by our two-stage heuristic.

VI. CONCLUSION

In this paper, we studied the register placement problem which is a key component of local clock structure optimization for high-performance circuits. We presented efficient methods for the weighted conflict graph construction and reduction, and formulated the problem as a minimum weighted maximum independent set problem. Besides, we proposed a novel two-stage heuristic to solve the problem. Experimental results showed the effectiveness of our heuristic.

REFERENCES

[1] C. J. Anderson *et al.*, "Physical design of a fourth-generation POWER GHz circuit," in *Proc. of ISSCC*, pp. 232–233, Feb. 2001.

[2] D. W. Bailey and B. J. Benschneider, "Clocking design and analysis for a 600-MHz Alpha circuit," *IEEE Jonual of Solid-State Circuits*, vol. 33, no. 11, pp. 1627–1633, Nov. 1998.

[3] R. Berridge *et al.*, "IBM POWER6 circuit physical design and design methodology," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 685–714, Nov. 2007.

[4] Y. Cheon *et al.*, "Power-aware placement," in *Proc. of DAC*, pp. 795–800, Jun. 2005.

[5] D. E. Duarte, N. Vijaykrishnan, and M. J. Irwin, "A clock power model to evaluate impact of architectural and technology optimizations," *IEEE Trans. on VLSI Systems*, vol. 10, no. 6, pp. 844–855, Dec. 2002.

[6] B. A. Gieseke *et al.*, "A 600 MHz superscalar RISC circuit with out-of-orderexecution," in *Proc. of ISSCC*, pp. 176–177, Feb. 1997.

[7] R. M. Karp, "Reducibility among combinatorial problems," in *Proc. of Sym. on Complexity Computer Computation*, pp. 85–103, Feb. 1972.

[8] K. Y. Lee and T. C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *Proc. of ASPDAC*, pp. 303–308, Jan. 2006.

[9] Y. Lu *et al.*, "Register placement for low power clock network," in *Proc. of ASPDAC*, pp. 588–593, Jan. 2005.

[10] Y. Lu *et al.*, "Navigating registers in placement for clock network minimization," in *Proc. of DAC*, pp. 176–181, Jun. 2005.

[11] M. Matson *et al.*, "Circuit implementation of a 600 MHz superscalar RISC circuit," in *Proc. of ICCD*, pp. 104–110, Oct. 1998.

[12] T. Okamoto, T. Kimoto, and N. Maeda, "Design methodology and tools for NEC electronics' structured ASIC ISSP," in *Proc. of ISPD*, pp. 90–96, Apr. 2004.

[13] A. Rajagopal, "Clock tree design challenges for robust and low power design," in *Proc. of ISPD*, pp. 168–168, Apr. 2006.

[14] P. J. Restle *et al.*, "A clock districation network for microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 5, pp. 792–799, May 2001.

[15] W. Shen, Y. Cai, X. Hong, and J. Hu, "Activity-aware registers placement for low power gated clock tree construction," in *IEEE Computer Society Annual Symposium on VLSI*, pp. 383–388, Mar. 2007.

[16] J. D. Warnock *et al.*, "The circuit and physical design of the POWER4 circuit," *IBM Journal of Research and Development*, vol. 46, no. 1, pp. 27–51, Jan. 2002.

[17] C. F. Webb *et al.*, "A 400 MHz S/390 circuit," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1665–1675, Feb. 1997.

[18] QUALEX-MS. [Online]. Available: http://www.stasbusygin.org/