less than the DICE configuration (hence incurring in a smaller overhead in layout and area). Moreover, the proposed cell has been simulated, and assessed for critical charge, power consumption, and delay to overcome the problems encountered in [8]. Using HSPICE, simulation results have confirmed that the proposed memory cell accomplishes the highest soft error tolerance through hardening (it has more than twice the critical charge than the 6T unhardened configuration) and an impressive power-delay product compared with the other hardened design commonly referred to as DICE. Therefore, the proposed hardened cell demonstrates superior resistance to soft errors and excellent performance metric as required for high performance memory design. Monte Carlo simulation has confirmed that the soft error hardening of the proposed memory cell is accomplished also in the presence of process variations.

References

[1] R. C. Baumann, "Soft errors in advanced semiconductor devices-part I: The three radiation sources," *IEEE Trans. Device Mater. Reliab.*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[2] C. Detcheverry, C. Detcheverry, C. Dachs, E. Lorfevre, C. Sudre, G. Bruguier, J. M. Palau, J. Gasiot, and R. Ecoffet, "SEU critical charge and sensitive area in a submicron CMOS technology," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 12, pp. 2266–2273, Dec. 1997.

[3] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 6, pp. 583–602, Jun. 2003.

[4] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron CMOS technology," *IEEE Trans. Nucl. Sci.*, vol. 43, no. 12, pp. 2874–2878, Dec. 1996.

[5] M. Omana, D. Rossi, and C. Metra, "Novel transient fault hardened static latch," in *Proc. IEEE ITC*, 2003, pp. 886–892.

[6] M. Omana, D. Rossi, and C. Metra, "Latch susceptibility to transient faults and new hardening approach," *IEEE Trans. Comput.*, vol. 56, no. 9, pp. 1255–1268, Sep. 2007.

[7] W. Wang and H. Gong, "Edge triggered pulse latch design with delayed latching edge for radiation hardened application," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 12, pp. 3626–3630, Dec. 2004.

[8] M. Nicolaidis, R. Perez, and D. Alexandrescu, "Low-cost highly-robust hardened cells using blocking feedback transistors," in *Proc. IEEE VTS*, Apr. 2008, pp. 371–376.

[9] Y. Sasaki, K. Namba, and H. Ito, "Soft error masking circuit and latch using Schmitt trigger circuit," in *Proc. IEEE DFTS*, Oct. 2006, pp. 327–335.

[10] J. M. Cazeaux, D. Rossi, M. Omana, C. Metra, and A. Chatterjee, "On transistor level gate sizing for increased robustness to transient faults," in *Proc. IEEE IOLTS*, 2005, pp. 23–28.

[11] C. -C. Wang, C.-F. Wu, R.-T. Hwang, and C.-H. Kao, "Single-ended SRAM with high test coverage and short test time," *IEEE J. Solid-State Circuits*, vol. 35, no. 1, pp. 114–118, Jan. 2000.

[12] "Berkeley Predictive Technology Model Website," 2007. [Online]. Available: http://www.eas.asu.edu/~ptm/

[13] S. Lin, Y. B. Kim, and F. Lombardi, "Soft-error hardening designs of nanoscale CMOS latches," in *Proc. IEEE VTS*, May 2009, pp. 41–46.

[14] S. Lin, Y. B. Kim, and F. Lombardi, "A novel design technique for soft error hardening of a nanoscale CMOS memory," in *Proc. IEEE MWSCAS*, Aug. 2009, pp. 679–682.

[15] "The MOSIS service," Marina del Rey, CA, 2009. [Online]. Available: http://www.mosis.org/Technical/Designrules/scmos/scmos-main.html

Efficient Package Pin-Out Planning With System Interconnects Optimization for Package-Board Codesign

Ren-Jie Lee and Hung-Ming Chen

***Abstract*—In conventional package design, engineers designate the ball grid array (BGA) pin-out manually, this always postpones the time-to-market (TTM) of products due to the turn-around between package and design houses. Recent papers propose a method of automatically generating the pin-out and taking signal integrity (SI), power delivery integrity (PI), and routability (RA) into account simultaneously by pin-block design and floorplanning, thus dramatically speeding up the developing time. However, this approach ignores the considerations of shorter path length and equilength/length matching in routing printed circuit board (PCB) trace and pin-out assignment for high-speed interface IP designs, such as USB and PCI Express. Since these features are the most important performance metrics during chip-package-board codesign, in this paper we propose the ideas to optimize the system interconnects during package pin-out design. These ideas keep the same minimized package size as aforementioned recent work and ensure that SI, PI, and RA can still be considered with significant reduction in design cost. It is achieved by relaxing the restriction of pin-block side and order on the package, usually specified by package designers. The experimental results on industrial chipset design cases show that the average improvement of our pin-block planner is over 40% when comparing the design cost with the previous work, among which we have one case accommodated over a thousand pins. Our ideas also work for any kind of pin-block or pin-group configurations.**

***Index Terms*—Pin-out planning, package-board codesign, system interconnects optimization.**

I. Introduction

As silicon technology scales, more and more circuits could be integrated into a single chip. The amounts of input/output (I/O) signals increase dramatically per unit area. This trend significantly arises the complication in package designs and signal interaction between package and board [1], [2]. The complete package-board codesign methodology should preserve the signal integrity (SI), power delivery integrity (PI), and routability (RA) of high-speed signals routing from package to printed circuit board (PCB) while optimizing the package size. One codesign approach regarding the automation of pin-out designation was published very recently in [3]. In this method, an experienced engineer has to determine the pin configuration chart based on the location of PCB components. Next, the proposed signal-pin patterns are selected for pin-blocks construction in package design where SI, PI, and RA have been accounted for after placing pin-blocks. It also proposes a near optimal approach to minimizing package size by mathematical (linear) programming. Finally, this methodology obtains the final pin assignment by applying a rather intuitive floorplanner which bends the pin-blocks located in the excess areas and fills them into the adjacent empty areas.

However, the cost function in [3] only considers the package size, this work exposes some weaknesses, shown as follows.

- The method in [3] ignores the connections between the ball grid array (BGA) pins and high-speed interface IP designs, which are

Manuscript received July 23, 2009; revised October 28, 2009 and January 12, 2010. First published February 25, 2010; current version published April 27, 2011. This work was supported in part by the National Science Council of Taiwan ROC under Grant NSC 97-2221-E-009-174-MY3.

The authors are with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: rjlee@vda.ee.nctu.edu.tw; hmchen@mail.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2010.2041562

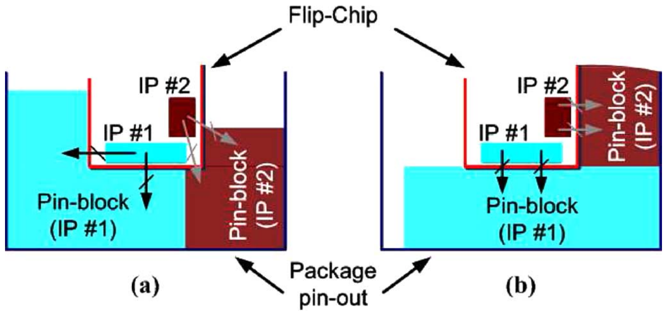


Fig. 1. Placement of pin-blocks and IPs. (a) Shows the worse pin-out assignment where the pin-block located around the package corner cannot meet the objectives of shorter path length and equilength (length matching consideration) on package routing. (b) Shows that our novel planning algorithms can overcome the drawbacks in (a).

hard macros located in chip, such as Universal Serial Bus (USB) and PCI Express interface. For the purpose of enhancing performance, the package routing for aforementioned IPs should consider shorter path length and balanced nets. Since the I/O pads in IPs are all fixed, the pin-block bent into two parts or located at the package corner will not meet these requirements. Fig. 1(a) shows the scenario caused by a poor pin-out.

- In addition to the considerations of pin-out assignment for IPs, the pin-out planner should also regard the general requirements of equilength or length matching for routing PCB traces. Fig. 2(a) shows the pin-block floorplanning results of [3]. When the floorplanner locates pin-blocks within the unsuitable region, it will cause longer wirelength in PCB escape routing. The longer wirelength illustrated with the darker lines in Fig. 2 will lead to greater efforts in achieving equilength in PCB routing task. Unfortunately, designers must predefine the placement side and order for all pin-blocks in previous approach, it then has no opportunity to change this circumstance due to these strictly specified configurations.

In order to improve the tasks of package routing for high-speed IPs as well as the PCB routing, the main objectives of this paper are to place pin-blocks near the preferred region, and to minimize the total wirelength and consider equilength in PCB escape routing as shown in Figs. 1(b) and 2(b).

In this paper, we develop an improved pin-block planner to overcome the drawbacks mentioned above. Our methodology applies simulated annealing based heuristic. By defining range constraints and using a specially-designed representation for pin-block placement, the proposed method not only optimizes the location of pin-blocks, but also minimizes the wirelength. The rest of this paper is organized as follows. We first define the constraints of pin-block planning in Section II. Section III describes an improved pin-block planner with cyclic number set (CNS) representation, and formulates the cost function with placement region violation. Section IV shows the experimental results based on the real and larger industry cases. Finally, we draw the conclusions in Section V.

II. Pin-Out Planning in Optimizing Package Performance and Board Wire-Planning

In the typical design flow, designers determine the pin configuration chart based on experience about the locations of PCB components and the characteristics of each signal group. The pin configuration chart defines all critical parameters including the distribution region (side), placement sequence (order), selected signal-pin pattern, and the number of power pins. According to the definition of this chart, the designer can finish the pin groups (or blocks) construction for all signal groups. Next, all pin-blocks will be placed along the defined side and order in which the first placed pin-block is located at the fixed location. Finally, after obtaining a rough pin-out designation and estimating the minimum package size, the pin-block floorplanning algorithm bends the pin-blocks allocated in the excess regions and shifts them into the adjacent empty regions. As a result, this shifting technique usually produces the bent pin-blocks located in the package corner without considering the package design for high-speed interface IPs such as USB and PCI Express. Moreover, the constraints defined in pin configuration chart restrict the margin and flexibility for optimizing the final pin-out.

In order to loosen the restriction from designers and to obtain a better pin-block placement, we have applied the concepts of defining the pre-placed modules, boundary constraints and range constraints in the tasks of floorplanning [4]–[6] and placement [7]–[9] to redefine a new set of constraints as follows. In general, the power/ground pins used for supplying power to core logic are arranged within the core block (*Core*). While power/ground pins are at the center of package and located beneath die, the current return path will be shorter and the heat generated from die can be transferred out through these pins [10]. For these reasons, the core block will be restricted by pre-placed constraint and placed at the center of pin-out designation. This constraint is shown as follows:

- $R_{\text{core}} = \{(x_p, y_p) | w_4 + 1 \le x_p \le w_4 + w_{\text{core}}, h_1 + 1 \le y_p \le h_1 + h_{\text{core}}\}$

where (x_p, y_p) is the coordinate of pin p; w_4, w_{core}, h_1, and h_{core} are the width/height shown in Fig. 3.

According to the location of components connecting with the pin-blocks, we define a new term *RangeSide* for each pin-block instead of placement side defined by designers. Fig. 3 shows an example where the pin-blocks are defined in *RangeSide*1 when the corresponding components are located in the south of PCB board. Therefore, all pins constrained in *RangeSide*1 must be located within the shaded region and routed toward the south to connect with components. Along the same rule, the *RangeSide*2, *RangeSide*3, and *RangeSide*4 are defined for the pin-blocks if the corresponding components are located in the east, north and west of PCB board, respectively. The detailed range constraints for each side are listed as follows $((x_p, y_p) \notin R_{\text{core}})$:

- $\text{RangeSide1} = \{(x_p, y_p) | 1 \le x_p \le w_4 + w_{\text{core}} + w_2, 1 \le y_p \le h_1 + h_{\text{core}}/2\}$;
- $\text{RangeSide2} = \{(x_p, y_p) | w_4 + w_{\text{core}}/2 + 1 \le x_p \le w_4 + w_{\text{core}} + w_2, 1 \le y_p \le h_1 + h_{\text{core}} + h_3\}$;
- $\text{RangeSide3} = \{(x_p, y_p) | 1 \le x_p \le w_4 + w_{\text{core}} + w_2, h_1 + h_{\text{core}}/2 + 1 \le y_p \le h_1 + h_{\text{core}} + h_3\}$;
- $\text{RangeSide4} = \{(x_p, y_p) | 1 \le x_p \le w_4 + w_{\text{core}}/2, 1 \le y_p \le h_1 + h_{\text{core}} + h_3\}$.

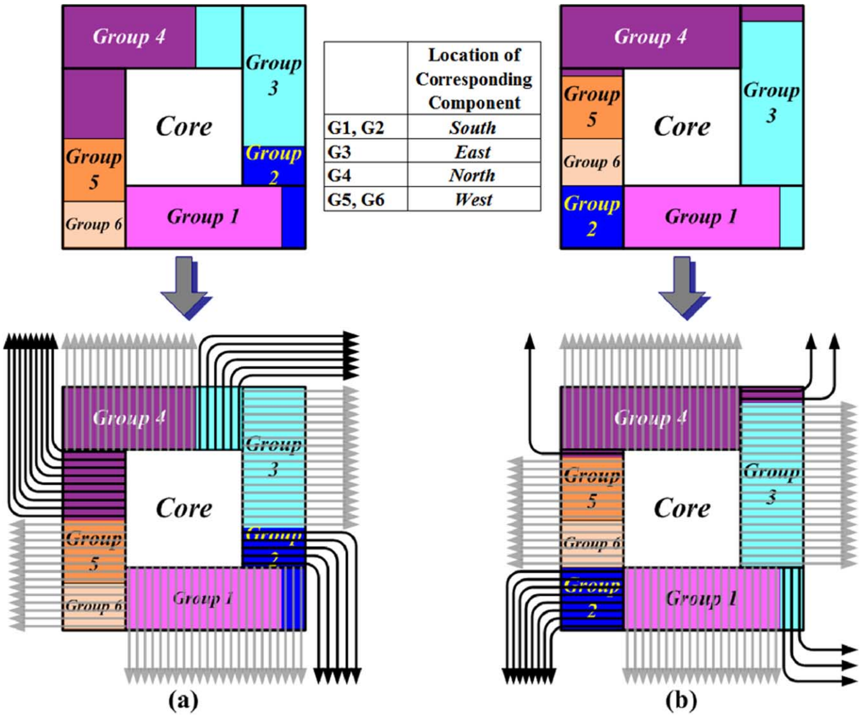
Comparing with the placement side constraint added by [3], the range constraints define the larger space for placing pin-blocks, thus offering the opportunities of improving pin-out designation. In addition to the optimization issue, our proposed pin-block planner also retains the feasibility of package design while satisfying all placement constraints including the preplaced and range constraints.

III. Range Constrained Pin-Block Planning With System Interconnects Optimization

As described in Section II, we will consider the core region (*Core*) as a preplaced module which must be placed in the center of the final pin-out. Besides, pin-blocks will be treated as range-constrained modules and located within given rectangular regions such that no pin-blocks are overlapping. This section presents a pin-block planning heuristic. It applies the algorithm which is based on simulated annealing (called SA) by using a specific CNS representation.

A. SA Pin-Block Planner

In this method, we use the results of [3] as the initial solution (they can be replaced by other grouping configurations). This pin-block planner eases the restriction of placement side and applies simulated



	Location of Corresponding Component
G1, G2	South
G3	East
G4	North
G5, G6	West

Fig. 2. Two results of pin-block floorplanning. (a) Shows the result of [3], it causes the longer wirelength (the darker lines) in PCB escape routing due to bad pin-block allocations. (b) Shows the result from our ideas which provides the shorter wirelength and obtains equilength routing for most pins.

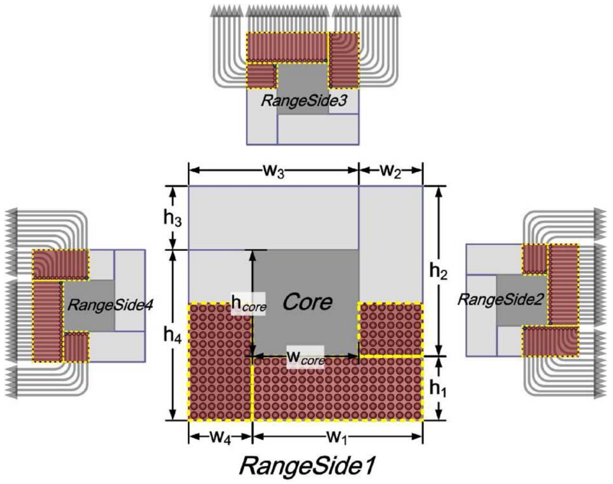


Fig. 3. Our practical range constraints for assigning pin-out. The pin-blocks are restricted in *RangeSide* 1, 2, 3, and 4 (each individual shaded region) when the corresponding components are in the south, east, north and west of PCB board, respectively.

annealing based heuristic with range constraints. First we introduce a special representation for pin-block planning, then we describe the floorplanning approach.

1) CNS Representation: The fundamental problem to floorplanning or placement lies in the representation of geometric relationship among modules [11]. Based on the consideration of the constraints and flexibility in pin-block planner, we propose a CNS representation. This representation is specially designed for pin-block planning since it can represent the adjacent relationship between blocks and the starting point when arranging pin-out. It can also describe all variables in perturbation.

Fig. 4 illustrates the CNS, the parentheses followed by an index represent the *RangeSide*, and those indices *I, II, III*, and *IV* represent *RangeSide*1, *RangeSide*2, *RangeSide*3, and *RangeSide*4, respectively. Pin-block groups constrained in each *RangeSide* are denoted as a number set within the parenthesis. Moreover, the placement sequence of pin-blocks is determined by the order of number set. For instance, the location of pin-blocks shown in Fig. 4(a) is represented as $\mathrm{CNS} = (1)_{\mathrm{I}}(2,3)_{\mathrm{II}}(4)_{\mathrm{III}}(5,6)_{\mathrm{IV}}$. It presents that *RangeSide*1 is the first *RangeSide* randomly selected by the planner, and the first group to be placed in this *RangeSide* is *group*1. *RangeSide*2, the next selected *RangeSide*, contains two groups where the placement order is *group*2, *group*3. Moreover, *RangeSide*2 follows the *RangeSide*1, *RangeSide*3 follows the *RangeSide*2, and so forth.

Unlike other representations in floorplanning/placement which are complicated and inapplicable for pin-block planning, the CNS representation describes the physical region and the relationship among pin-blocks. Once the CNS has been determined based on designer input, the planner can easily place the pin-blocks. Compared with the pin-block floorplanner in [3], which used 2-D array to store the locations for all pins, our planner can simply and efficiently transform the representation to real pin-block placement.

2) Simulated Annealing Based CNS Floorplanning: The features of CNS presented above simplify the transformation between representation and pin-block placement. They also facilitate the optimization of pin-block planning in our SA-based algorithm. The optimization process is described as follows.

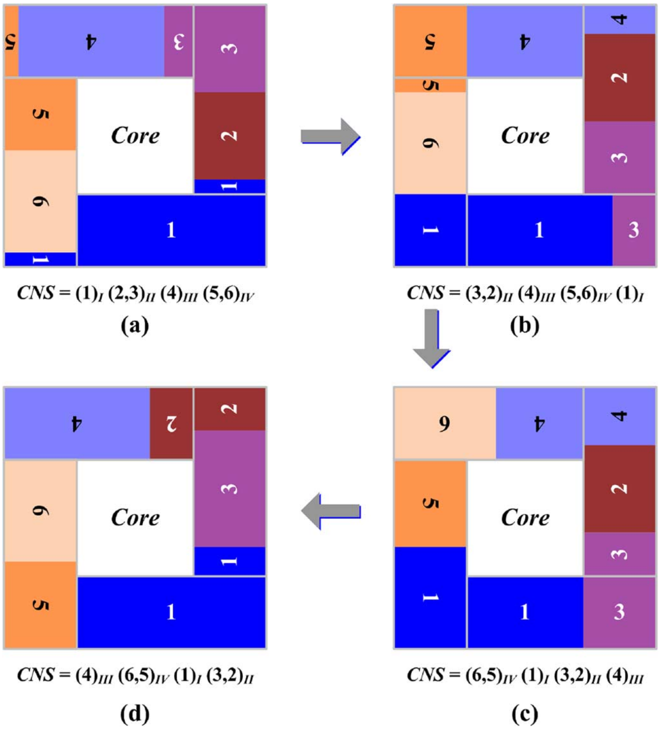
- Solution Perturbation and Neighborhood Structure:

  Step 1: Randomly select one *RangeSide* from the CNS of initial (or previous) solution.

  — **Move**: Randomly choose two groups in this *RangeSide*, then exchange their sequence.

  Step 2: Randomly decide the first pin location of the updated first group then place the pin-block.

  Step 3: The rest of groups defined in the selected *RangeSide* are placed along the updated sequence determined in previous move.



CNS = $(1)_I\ (2,3)_{II}\ (4)_{III}\ (5,6)_{IV}$
(a)

CNS = $(3,2)_{II}\ (4)_{III}\ (5,6)_{IV}\ (1)_I$
(b)

CNS = $(4)_{III}\ (6,5)_{IV}\ (1)_I\ (3,2)_{II}$
(d)

CNS = $(6,5)_{IV}\ (1)_I\ (3,2)_{II}\ (4)_{III}$
(c)

Fig. 4. Illustration of CNS representation and examples of perturbation process. (a) Shows the initial configuration. (b) Shows the first perturbation case, the *RangeSide*2 has been selected and its group orders are exchanged (Step 1). The first pin location of *Group*3 is randomly decided, then the planner places all pins in *RangeSide*2 (Step 2 and 3). Following the updated CNS, the groups defined in the remainder of *RangeSide* are placed (Step 4). (c) and (d) show another two perturbation cases.

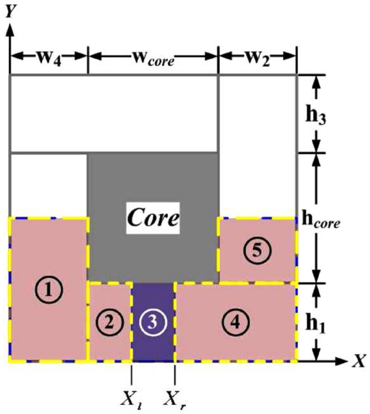


Fig. 5. Estimations of the cost for *RangeSide*1. The cost/penalty is the placement deviation induced when pin-blocks are placed away from the defined region ($X_l \le x_p \le X_r$).

Step 4: The remainder of groups defined in the other *RangeSide* are placed according to the sequence determined in previous solution.

Step 5: Save the updated CNS representation for the new solution.

To produce a feasible solution, after randomly selecting one *RangeSide* from the CNS of previous solution, our pin-block planner randomly chooses two groups in the selected *RangeSide* and swaps their sequence thus modifying the CNS. The rest of steps are proceeded depending on the perturbed CNS. Fig. 4 shows the examples of perturbation processes, (a) is the initial/previous solution and the placement of pin-blocks starts from *group*1 in *RangeSide*1. Since the *RangeSide* has been perturbed, the planner revises the CNS and the placement is reinitiated from *RangeSide*2 as shown in Fig. 4(b). According to the move, the group orders in *RangeSide*2 are exchanged (first step). Next, the first pin location of *group*3 is randomly decided, and the planner places the pins of *group*3 and *group*2 (second and third steps). After these steps, the rest of groups must be located along the range constraints and the sequence described in the perturbed CNS (fourth step). Finally, our method saves the updated CNS of modified pin-block location for next iteration (fifth step). Fig. 4(c) and (d) show the other two perturbation cases.

- Annealing Schedule: our SA planner uses the following schedule to minimize the cost function, then obtains an optimized pin-out.
  —$T_0 = 100$; $\alpha = 0.9$; $M = 5$; $Maxtime = 500$.
  where T_0 is the initial temperature, α is the cooling rate, M represents the time until the next parameter update, and *Maxtime* is total allowed time for the annealing process. After obtaining the initial solution, the perturbation procedure is iteratively invoked to perturb this given solution and get new solution until the total allowed time is exceeded.

B. Optimizing Objective Function

For optimizing the pin-out designation, we use the penalty term, which is the deviation of desired pin-block location, as our cost function. To emphasize the location difference, its value is set to be the square of distance estimated between the pin location and the defined placement boundary. An example is shown in Fig. 5, the designer can define a preferred boundary as the constrained region ($X_l \le x_p \le X_r$) for assigning pins according to the size and floorplan of corresponding IPs. Therefore, signal pins will obtain zero penalty when they are placed within the preferred region. The detailed estimation of penalty term in *RangeSide*1 is formulated as follows.

- *Region* 1: Penalty $= (|y_p| + |w_4 - X_l|)^2$ when $1 \le x_p \le w_4$, $1 \le y_p \le (h_1 + h_{\text{core}}/2)$.
- *Region* 2: Penalty $= |x_p - X_l|^2$ when $w_4 + 1 \le x_p \le X_l$, $1 \le y_p \le h_1$.
- *Region* 3: Penalty $= 0$ when $X_l \le x_p \le X_r$, $1 \le y_p \le h_1$.
- *Region* 4: Penalty $= |X_r - x_p|^2$ when $X_r \le x_p \le (w_4 + w_{\text{core}} + w_2)$, $1 \le y_p \le h_1$.
- *Region* 5: Penalty $= [|X_r - (w_4 + w_{\text{core}} + w_2)| + |y_p - h_1|]^2$ when $(w_4 + w_{\text{core}} + 1) \le x_p \le (w_4 + w_{\text{core}} + w_2)$, $(h_1 + 1) \le y_p \le (h_1 + h_{\text{core}}/2)$.

Since designers usually connect power/ground pins with power/ground planes by using the nearest vias, penalties which are added by power/ground pins located outside the constrained region will be ignored in our proposed method. By minimizing the total cost, our methodology not only decreases the signal-net length but also locates the pin-blocks near the defined boundary. Therefore, the pin-block planner can match most of the requirements of shorter path length and equilength on package design and PCB routing.

IV. Experimental Results and Discussions

We have implemented our methodology in C++ and the platform is on Intel Pentium M 1.7 GHz with 512 MB memory. Five industrial chipset cases, which act as bridges of all components on motherboard are used as our benchmarks (shown in Table I). In our experiments, the penalty term (in Section III-B) which is the placement deviation is considered as our cost function. For the reason of acquiring shorter path length and equilength (length-matching consideration) on package design and PCB routing, the designer can define a preferred region then force the pin-blocks to be planned in that boundary by minimizing the

TABLE I
SUMMARY OF FIVE TEST CASES WHICH HAVE ENTIRELY DIFFERENT CHARACTERISTICS. THE GROUP NUMBER IS THE AMOUNT OF INTERFACES BETWEEN CHIPSET AND INDIVIDUAL COMPONENTS

	Group NO.	Signal Pin NO.	Power Pin NO.	Total Pin NO.
Test Case I	6	254	80	334
Test Case II	6	346	48	394
Test Case III	20	510	168	678
Test Case IV	25	504	216	720
Test Case V	27	770	232	1002

TABLE II
COMPARISONS OF PENALTY TERM (PLACEMENT DEVIATION) FOR [3] AND SA PIN-BLOCK PLANNER. THE RESULTS SHOW THAT OUR APPROACH HAS SIGNIFICANT IMPROVEMENT IN ALL TEST CASES ("IMP." IS THE IMPROVEMENT ON THE PENALTY TERM)

	[3]	SA		
	Penalty	Penalty	$Imp.$ (%)	Time
Test Case I	6200	2619	$+57.76$	$< 2.0\ min$
Test Case II	8708	5802	$+33.37$	$< 3.5\ min$
Test Case III	27048	14818	$+45.25$	$< 6.0\ min$
Test Case IV	31590	16961	$+46.31$	$< 6.0\ min$
Test Case V	77614	51079	$+34.19$	$< 9.5\ min$
$Avg.$	–	–	$+43.38$	–

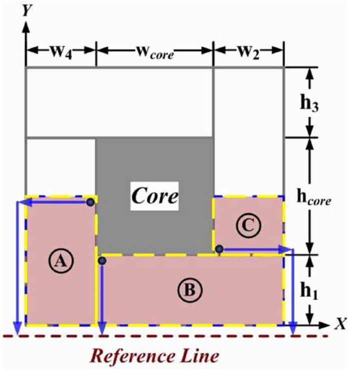


Fig. 6. Wirelength estimation for *RangeSide*1. The wirelength is calculated in Manhattan distance from signal pin to the reference line (dotted line on the bottom).

TABLE III
COMPARISONS OF WIRELENGTH WITH APPROACHES IN [3] AND SA PIN-BLOCK PLANNER. THE RESULTS SHOW THAT OUR IMPROVED METHOD HAS POSITIVE IMPROVEMENT OVER [3] EXCEPT THE TEST CASE I ("IMP." IS THE IMPROVEMENT ON THE TOTAL WIRELENGTH "WL")

	[3]	SA	
	WL	WL	$Imp.$ (%)
Test Case I	1199	1216	-1.42
Test Case II	1712	1650	$+3.62$
Test Case III	2406	2226	$+7.48$
Test Case IV	2442	2173	$+11.02$
Test Case V	4124	3592	$+12.90$
$Avg.$	–	–	$+6.72$

penalty term. In our experiments, we set the center area of each package side as the preferred region as shown in Fig. 5.

Experimental results are presented as the comparisons of the SA pin-block planner and our implementation for [3]. Although the SA planner needs more runtime, the results shown in Table II demonstrate that the SA planner is better than the previous work in average. Table II also shows that SA planner has positive improvement in penalty term when compared with that in [3], and the runtime of designating and optimizing final pin-out for all test cases is less than ten minutes. For the design which has enormous pin-block groups, our approaches can obtain the significant improvement.

As described in the definition of *RangeSide*, signal pins located in *RangeSide*1 will route nets toward the south of PCB board then connect with the components. When our algorithm finds the minimum cost, it is to drive the pin-blocks to move to the center of *RangeSide*1 thus theoretically minimizing the signal-net length. Therefore, the optimized pin-out designation is evaluated by means of calculating the performance metric, the total wirelength. Fig. 6 shows an example of wirelength estimation for pins located in *RangeSide*1. It is estimated in Manhattan distance from signal pin to the reference line (indicated in a dotted line) of each package side. The wirelength estimation for *RangeSide*1 are listed as follows.

- *Region* A: $\mathrm{WireLength} = |x_p| + |y_p|$ when $1 \leq x_p \leq w_4$, $1 \leq y_p \leq (h_1 + h_{\mathrm{core}}/2)$.
- *Region* B: $\mathrm{WireLength} = |y_p|$ when $(w_4 + 1) \leq x_p \leq (w_4 + w_{\mathrm{core}} + w_2)$, $1 \leq y_p \leq h_1$.
- *Region* C: $\mathrm{WireLength} = |x_p - (w_4 + w_{\mathrm{core}} + w_2 + 1)| + |y_p|$ when $(w_4 + w_{\mathrm{core}} + 1) \leq x_p \leq (w_4 + w_{\mathrm{core}} + w_2)$, $(h_1 + 1) \leq y_p \leq (h_1 + h_{\mathrm{core}}/2)$.

According to the definition of *RangeSide*, the reference lines used for calculating the wirelength in *RangeSide*2, *RangeSide*3, and *RangeSide*4 are individually established in the east, north, and west of package. The results of wirelength estimation are shown in Table III. Again, in most cases the SA planner has positive improvements over [3] by minimizing total cost. However, there is negative improvement produced by our planner in test case I. Because the pin-block size and group number in each *RangeSide* are varied, in our planner all pin-blocks are located near each center of *RangeSide* to optimize the package performance for high speed IPs. In this case the wirelength is increased slightly due to the compromise between penalty of each *RangeSide*.

As we mentioned in Section I, our method will try to avoid the bent pin-block to meet the objectives of shorter path length and equilength on package routing. However, in out experimental results some pin-blocks are still bent into two parts after minimizing total cost. That is because some interfaces possess enormous I/O pins and are grouped into large pin-blocks in our industrial test cases. Besides, power/ground pins will not be added penalties in proposed method when they are located outside the constrained region. As a result, the bent pin-blocks are inevitable, but the proposed method will mitigate the impacts. Finally, the results shown in Tables II and III indicate that in most cases our methodologies not only consider the package design but also minimize the wirelength in PCB escape routing.

V. CONCLUSION

We have proposed an improved pin-block planner with range constraints and a representation for automating pin-out designation. Based on the method of pin-block design in [3], our approach minimizes the package size and considers SI, PI, and RA as that in [3]. The experimental results show that the proposed methodologies provide significant improvement especially for large number of pin-block groups. Furthermore, we can use the range concept to restrict the pin-block location within the preferred region thus optimizing the package performance and board wire-planning.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for providing precious comments to greatly improve this paper.

REFERENCES

[1] A. Hasan and D. Sato, "BGA package ball field interaction with manufacturing and design," in *Proc. IEEE Electron. Components Technol. Conf.*, 2004, pp. 326–333.

[2] A.-H. Titus and B. Jaiswal, "A visualization-based approach for bump-pad/io-ball placement and routing in flip-chip/BGA technology," *IEEE Trans, Adv. Packag.*, vol. 29, no. 3, pp. 576–586, Aug. 2006.

[3] R.-J. Lee and H.-M. Chen, "Fast flip-chip pin-out designation respin for package-board codesign," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 8, pp. 1087–1098, Aug. 2009.

[4] F.-Y. Young and D.-F. Wong, "Slicing floorplans with pre-placed modules," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 1998, pp. 252–258.

[5] F.-Y. Young, D.-F. Wong, and H.-H. Yang, "Slicing floorplans with boundary constraints," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 9, pp. 1385–1389, Sep. 1999.

[6] F.-Y. Young, D.-F. Wong, and H.-H. Yang, "Slicing floorplans with range constraint," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 2, pp. 272–278, Feb. 2000.

[7] Y.-H. Jiang, J. Lai, and T.-C. Wang, "Module placement with preplaced modules using the B*-tree representation," in *Proc. Int. Symp. Circuits Syst.*, 2001, pp. 347–350.

[8] J.-M. Lin, H.-E. Yi, and Y.-W. Chang, "Module placement with boundary constraints using B*-trees," *IEE Proc.—Circuits, Devices Syst.*, pp. 251–256, 2002.

[9] H. Murata, K. Fujiyoushi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 1995, pp. 472–479.

[10] Altera Corp., San Jose, CA, "Designing with high-density BGA packages for Altera devices," Appl. Note AN-114-4.0, Feb. 2006.

[11] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplanss," in *Proc. IEEE/ACM Design Autom. Conf.*, 2000, pp. 458–463.

An Enhanced Canary-Based System With BIST for SRAM Standby Power Reduction

Jiajing Wang, Alexander Hoefler, and Benton H. Calhoun

***Abstract*—To achieve aggressive standby power reduction for static random access memory (SRAM), we have previously proposed a closed-loop $V_{\rm DD}$ scaling system with canary replicas that can track global variations. In this paper, we propose several techniques to enhance the efficiency of this system for more advanced technologies. Adding dummy cells around the canary cell improves the tracking of systematic variations. A new canary circuit avoids the possibility that a canary cell may never fail because it resets into its more stable data pattern. A built-in self-test (BIST) block incorporates self-calibration of SRAM minimum standby $V_{\rm DD}$ and the initial failure threshold due to intrinsic mismatch. Measurements from a new 45 nm test chip further demonstrate the function of the canary cells in smaller technology and show that adding dummy cells reduces the variation of the canary cell.**

***Index Terms*—Built-in self test (BIST), data retention voltage (DRV), standby power, static random access memory (SRAM), variation.**

I. Introduction

Since SRAM/Cache continues to be the largest and most dense component in many digital systems or system-on-chips (SoCs), its leakage power dominates the overall leakage power of the system. One of the most effective leakage reduction techniques is supply voltage $(V_{\rm DD})$ scaling. All the leakage current components, including sub-threshold leakage, gate leakage, and junction leakage current, decrease dramatically with a smaller $V_{\rm DD}$. Leakage power decreases even more rapidly due to the reduction of both $V_{\rm DD}$ and leakage current. Many designs have exploited $V_{\rm DD}$ scaling during standby and/or active operation for SRAM leakage power reduction [1]–[4]. However, the scaled $V_{\rm DD}$ not only reduces cell stability itself but also heightens the sensitivity of cell stability to mismatch. The data retention voltage (DRV) is the minimum $V_{\rm DD}$ for the cell to preserve its data [3]. Local variation spreads the DRV of the cells across the chip. To preserve all the data in an SRAM, $V_{\rm DD}$ must be above the DRV of the worst cell within the SRAM array, which we call standby Vmin in this paper. Standby Vmin varies with process variations, voltage fluctuations, and temperature changes (PVT variations). Thus we must address this Vmin variability when choosing standby $V_{\rm DD}$.

The most straightforward solution is the worst-case based open-loop approach, in which the standby voltage is picked based on the DRV for the worst scenario at design time and maintains unchanged for all the scenarios. Although it is robust, substantial power and energy are wasted because of two reasons. First, the worst PVT scenario only occurs in extreme conditions like extremely high temperature, which is rare for most applications. Second, the margin for the worst PVT protection can be quite large, and it even becomes larger as CMOS technology continuously scales.

Manuscript received May 14, 2009; revised October 13, 2009. First published March 11, 2010; current version published April 27, 2011. This work was supported in part by SRC & FCRP C2S2.

J. Wang and B. H. Calhoun are with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904 USA (e-mail: jjwang@virginia.edu; bcalhoun@virginia.edu).

A. Hoefler is with Freescale Semiconductor, Austin, TX 78729 USA (e-mail: alexander.hoefler@freescale.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2010.2042184