A Study of Row-Based Area-Array I/O Design Planning in Concurrent Chip-Package Design Flow

REN-JIE LEE, Novatek Microelectronics, Taiwan and National Chiao Tung University, Taiwan
HUNG-MING CHEN, National Chiao Tung University, Taiwan

IC-centric design flow has been a common paradigm when designing and optimizing a system. Package and board/system designs are usually followed by almost-ready chip designs, which causes long turn-around time communicating with package and system houses. In this article, the realizations of area-array I/O design methodologies are studied. Different from IC-centric flow, we propose a chip-package concurrent design flow to speed up the design time. Along with the flow, we design the I/O-bump (and P/G-bump) tile that combines I/O (and P/G) and bump into a hard macro with the considerations of I/O power connection and electrostatic discharge (ESD) protection. We then employ an I/O-row based scheme to place I/O-bump tiles with existed metal layers. By such a scheme, it reduces efforts in I/O placement legalization and the redistribution layer (RDL) routing. With the emphasis on package design awareness, the proposed methods map package balls onto chip I/Os, thus providing an opportunity to design chip and package in parallel. Due to this early study of I/O and bump planning, faster convergence can be expected with concurrent design flow. The results are encouraging and the merits of this flow are reassuring.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids

General Terms: Algorithms, Design

Additional Key Words and Phrases: Area-array IC design, I/O-bump planning, chip-package feasibility study, concurrent IC design flow

ACM Reference Format:
Lee, R.-J. and Chen, H.-M. 2013. A study of row-based area-array I/O design planning in concurrent chip-package design flow. ACM Trans. Des. Autom. Electron. Syst. 18, 2, Article 30 (March 2013), 19 pages.
DOI: http://dx.doi.org/10.1145/2442087.2442101

1. INTRODUCTION

Modern I/O planning is divided into two categories: peripheral I/O and area-array I/O. The peripheral I/O planning is to place I/Os along the sides of core boundary and the I/Os are connected with package balls by using wire-bonding process. Whereas, as shown in Figure 1(a) and Figure 2(a), this planning method always requires larger die size to accommodate I/Os and pads, and degrades the signal and power integrity in off-chip signaling due to parasitics and coupling effects [Caldwell et al. 1998; Wang et al. 2004]. On the other hand, the area-array I/O planning using the flip-chip technique overcomes those drawbacks in the peripheral style with higher cost. Figure 1(b) and Figure 2(b) and (c) show that the flip-chip area-array I/O technology offers the considerable flexibility in optimizing core-I/O placement and package routing. It also has the features of smaller die size, higher I/O density, lower parasitic effects and better heat dissipation, and therefore meets the requirements of designing advanced and high-performance ICs in deep-submicrometer (DSM) environment

Author's address: R.-J. Lee, email: rjlee@vda.ee.nctu.edu.tw.

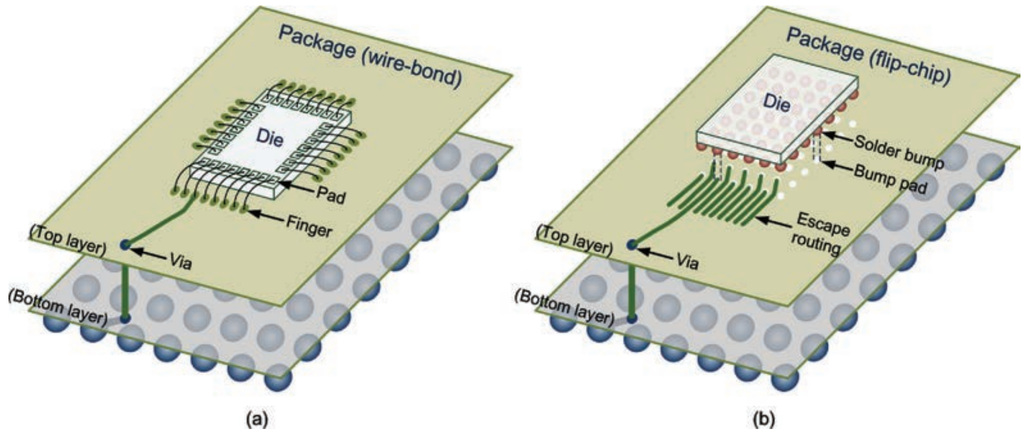

Fig. 1. The simplified illustration of two general package types. (a) is the wire-bond package, and (b) is the flip-chip package.

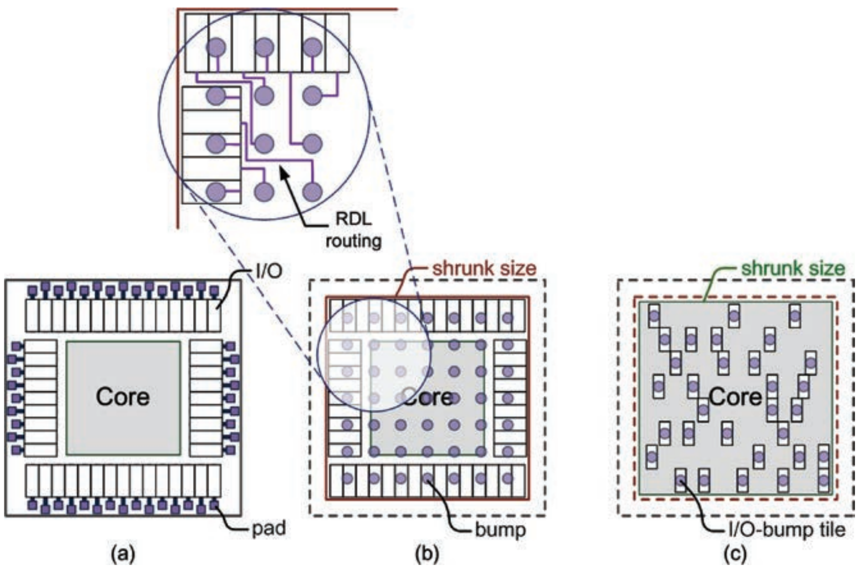

Fig. 2. The die size comparison of designs with different I/O planning. (a) is implemented with the peripheral I/Os and wire-bond package, (b) is a general flip-chip design planned with extrinsic area-array I/Os and connected to bumps by RDL routing and (c) is the proposed flip-chip design applied intrinsic area-array IO concept by using I/O-bump tiles.

[Pascariu et al. 2003; Chang and Chen 2008]. Table I reassurts the advantages over traditional peripheral I/O methodology, with the addition of our row-based I/O planning. A further explanation is provided in our experimental results.

1.1. Previous Works

Regarding the current flip-chip area-array ICs, two different area-array I/O regimes are well known in industry: the extrinsic area-array I/O and intrinsic area-array I/O [Tan et al. 1997]. In Maheshwari et al. [1995], authors distinguished area-array I/O as redistribution and true area-I/O. For the extrinsic area-array IC, I/Os are placed along the peripheral boundary of the core. It is similar to the peripheral I/O planning but

Table I.
The industrial chipset designs. Results show that the die size of I/O-pad limited designs ($d2$ and $d3$ are core limited designs) can be reduced by using our row-based area-array I/Os instead of traditional peripheral I/Os.

	Peripheral I/O				Area-Array I/O		
	Tech. (um)	Die size (um^2)	I/O size (um^2)	I/O number	Die size (um^2)	I/O-bump tile size (um^2)	Die size difference
$d1$	0.18	2500^2	115×65	220	2327^2	160×80	−6.92%
$d2$	0.18	3250^2	200×60	188	3475^2	160×80	+6.93%
$d3$	0.18	2510^2	140×65	130	2742^2	160×80	+9.25%
$d4$	0.13	2580^2	120×75	200	2364^2	160×80	−8.39%
$d5$	0.13	4720^2	115×50	628	4600^2	160×80	−2.55%
$d6$	0.09	6800^2	175×65	390	6645^2	160×80	−2.29%

(The utilization rate of core cells is kept the same)

uses a dedicated redistribution layer (RDL) to redistribute nets from peripheral I/Os to area-array bumps located in the center of core area, as shown in Figure 2(b). It migrates package design from wire-bond to flip-chip technology by a re-design process, namely RDL routing task, thus gaining the advantages of smaller parasitic effects and less thermal issues. However, the die size of this redistributed design will still be enlarged while the number of I/Os being increased due to the same I/O planning with that of peripheral I/Os. Figure 2(c) shows the intrinsic area-array IC, on the contrary, I/Os and bumps are freely located in the center of core region thus shrinking the die size and improving the flexibility in core-I/O placement as well as package routing.

Several works proposed methodologies to deal with flip-chip area-array ICs. Fang et al. [2007] and Fang et al. [2009] applied the network-flow-based and the integer-linear-programming (ILP) based RDL routing algorithms for designing extrinsic area-array ICs. The two-stage technique not only completes 100% routability, but also reduces the total RDL wirelength and the signal skews compared with an industrial heuristic algorithm. On the other hand, two recent works focused on planning and placing intrinsic area-array I/Os. Chen et al. [2006] applied I/O clustering concept to place I/Os, and formulated the problem as a min-cost maximum flow problem. The encouraging results indicate that the method not only achieves better timing performance but also reduces the design cost when compared with the conventional method commonly used by the designers. In Xiong et al. [2006], based on ILP framework, authors formulated a constraint-driven I/O planning and placement problem, and solved it by a multi-step algorithm. The experimental results show that their algorithm can effectively deal with large scale I/O placement problem and satisfy all design constraints in real design. Another recent work proposed a network-flow based multi-RDL router for the intrinsic area-array flip-chip ICs [Fang and Chang 2008]. For chip-package codesign, their router completes both chip-level routing from block ports to I/O pads and package-level RDL routing from I/O pads to bump pads, thus improving the design convergence.

All the aforementioned researches achieve some notable results. However, as the chip-package design has become increasingly challenging, more considerations must be accounted for as noted here.

—These approaches assume that bumps are arranged in fixed array locations with unique spacing, they then design the area-array ICs by planning the I/O placement with cost functions and constraints or connecting bumps and I/Os with RDL routing. It is inevitable that area-array I/O planning and RDL routing need a lot of efforts to coordinate with the core cell placement and routing. Moreover, the presumed

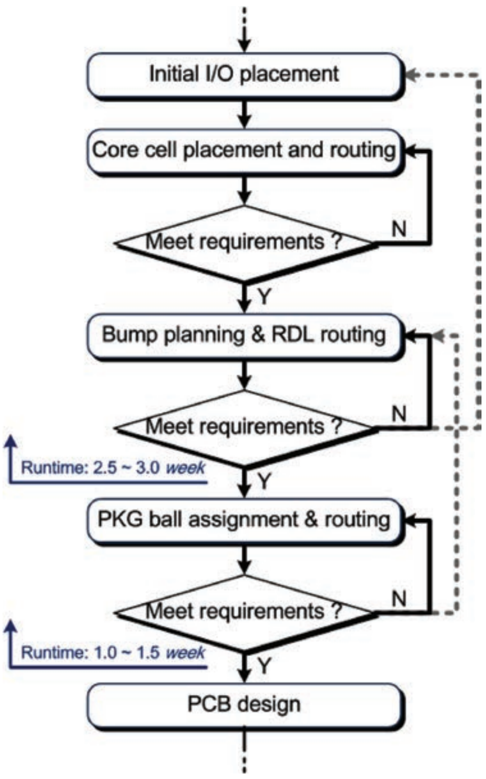

Fig. 3. The conventional design flow. It iteratively optimizes the locations of core cells and I/Os, then performs the bump planning, RDL routing and finishes the bump placement for package routing. This sequential design flow takes long turn-around time to meet all design requirements.

uniform and fixed bump location restricts the flexibility in optimizing both chip and package designs.

—Most of previous works focus on designing area-array ICs and do not emphasize the package ballplan, which is given and optimized for PCB design. Without considering the package ballplan, the I/Os and bumps will possibly lead to complicated package substrate routing, even failed package design [Sheth et al. 2006].

—The conventional design flow, which is IC-centric [Fontanelli et al. 2002], has an implicit issue in design turn-around. As shown in Figure 3, chip designer usually assigns the initial I/O placement according to the results of core cell floorplanning. Next, the core cell placement and routing are iteratively optimized until the final results meet the design requirement such as minimum total wirelength. After that, this bottom-up design flow processes the bump planning and RDL routing, then proceeds to the package-level design including package ball assignment and routing. The major disadvantage of this sequential design flow is evident: it will probably result in long and costly re-spin cycles on satisfying entire system's design constraints.

1.2. Our Contributions

In order to deal with the issues enumerated above, here we propose a realizable area-I/O design methodology. The contributions presented in this article are as follows.

—In this study, we propose a concurrent design flow, as shown in Figure 4. Since the necessary information is preliminarily provided by package-aware I/O-bump

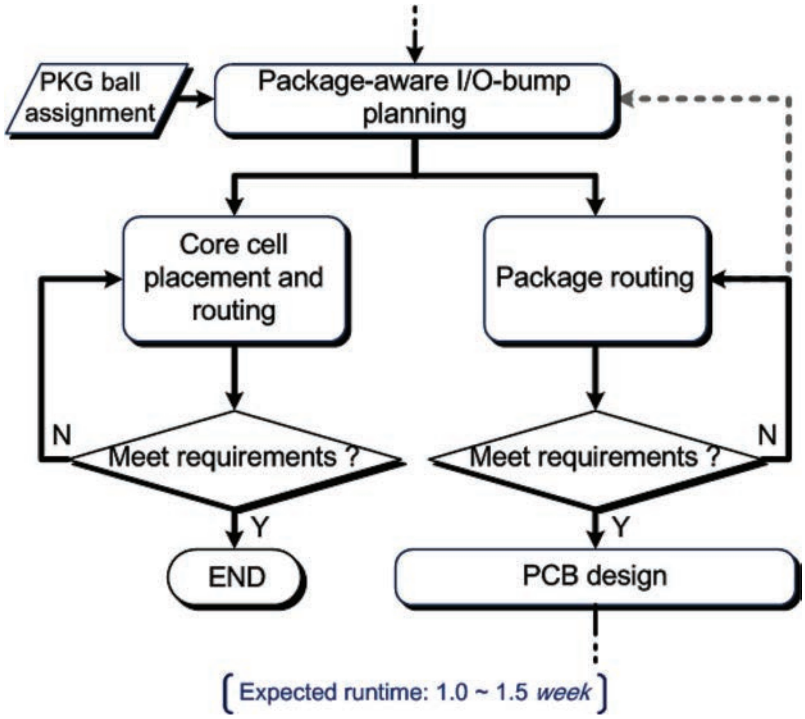
PKG ball assignment
Package-aware I/O-bump planning
Core cell placement and routing
Package routing
Meet requirements ?
N
Y
Meet requirements ?
N
Y
END
PCB design
Expected runtime: 1.0 ~ 1.5 week

Fig. 4. The proposed concurrent chip-package design flow. Through package-aware I/O-bump planning, it completes the core cell and I/O placement and package routing simultaneously, thus reducing the design cycles between chip and package.

planning, the concurrent design flow completes chip-level core cell placement and routing and package-level bump-ball routing in parallel. Comparing with the sequential design flow (Figure 3), the proposed one will expectedly reduce the turn-around time when designing chip and package.

—To achieve concurrent design flow, one of the major contributions is I/O-bump and P/G-bump tile designs. Through designing the specific I/O-bump tiles (shown in Figure 5), complicated RDL routing efforts can be avoided. Under such circumstance, not only is the signal skew caused by RDL routing eliminated, but also the RDL can be released for I/O power delivery. Therefore, with the I/O-row based scheme (Figure 6), I/O-bump tiles can be freely placed at core area without keeping the same spacing. As a result, we significantly improve the flexibility in arranging I/Os and bumps for chip-level and package-level design.

—Another contribution is the package-aware I/O-bump planning in our concurrent design flow, which considers package ball locations and the essential concerns of chip-package codesign, such as noncrossing routing, the shortest wirelength and the minimum length deviation among all nets. The quick evaluation of the configurations can be achieved by different planning strategies.

Table II shows the average runtime of a SoC system design.[1] That has approximately $5.0M$ gate count and 500 I/Os. While implementing this system with typical design

[1]This SoC system provided as an example design is a chipset IC packaged with a flip-chip ball grid array (BGA). The related design information about the average runtime and design solutions in typical design flow is derived from an experienced engineer who is working for a design service compony in Hsinchu Science Park, Taiwan.

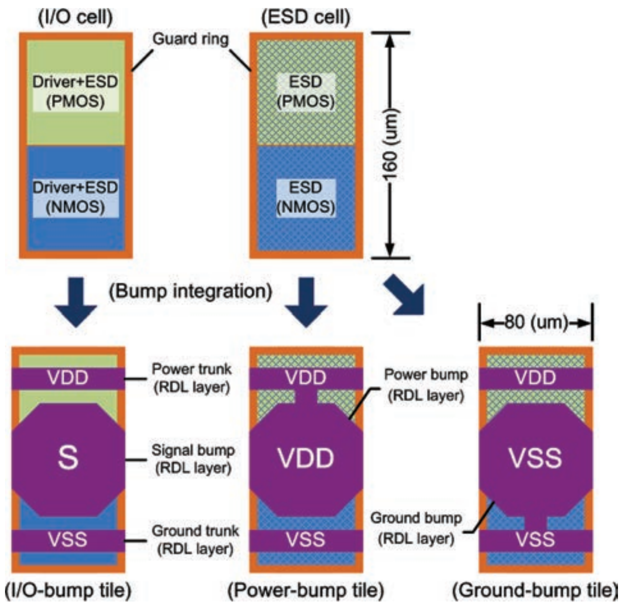

Fig. 5. The design of proposed I/O-bump tile that integrates I/O and bump into the single and unique interface between chip and package. Power-bump and ground-bump tiles are developed based on the same concept as well.

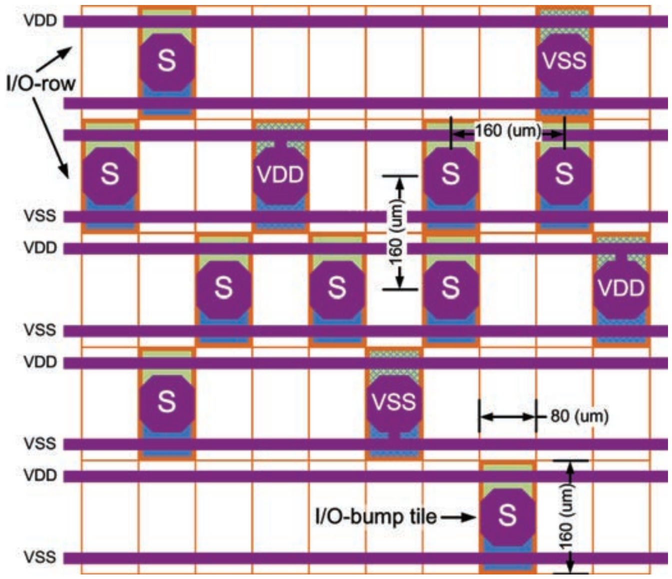

Fig. 6. The I/O-row based I/O-bump planning scheme. The width/height of tile and I/O-row are designed to satisfy the bump size/pitch. This scheme therefore simplifies the placement legalization of I/O-bump tiles.

Table II.
The runtime comparison of a SoC system design implemented with typical design flow and our proposed approaches. Through using automated methodologies and going with the concurrent design flow, the total runtime of this system can be significantly improved.

		t_1	t_2	t_3	t_4	t_5	t_6	t_7
	Design task	I/O placement	Core cell placement	Core cell routing	Bump placement	RDL routing	Ball assignment	Package routing
Typical design flow (sequential)	Solution	Manual design	Cadence SoC Encounter	Cadence SoC Encounter	Manual design	Cadence Allegro PKG Designer	Manual design	Cadence Allegro PKG Designer
	Average runtime	≈ 2.0 $(week)$			≈ 1.0 $(week)$		≈ 1.5 $(week)$	
	$Total\ runtime = \sum_{i=1}^{7} t_i = 4.5$ $(week)$							
Our design flow (concurrent)	Solution	Automated methodology	Cadence SoC Encounter	Cadence SoC Encounter	Automated methodology	No necessary	Manual design	Cadence Allegro PKG Designer
	Average runtime	< 1.5 $(week)$			< 0.1 $(week)$		≈ 1.5 $(week)$	
	$Total\ runtime = \max(t_i) \approx 1.5$ $(week)$							

flow, the total runtime obtained from industry is around 4.5 (*week*). However, through using the proposed methodologies and going with the concurrent design flow, the total runtime can be significantly reduced to 1.5 (*week*). As a result, our approaches shorten the turn-around in converging the solutions of chip and package, and tremendously reduce the time-to-market (TTM) for designs.

The rest of the article is organized as follows. Section 2 introduces the row-based I/O-bump tile design. Section 3 defines the problem of package-aware I/O-bump planning, while Section 4 describes the I/O-bump planning methods. Section 5 shows the experimental results, followed by the conclusion in Section 6.

2. NOVEL I/O-BUMP TILE DESIGN AND I/O-ROW BASED PLANNING

In order to ease the extra efforts in chip-package codesign and to achieve concurrent design flow, we integrate the I/O (and P/G) and area-array bump into one specific tile called I/O-bump[2]. It consequently provides all information needed in both chip-level core-I/O placement and package-level bump-ball routing. As shown in Figure 5, each I/O-bump tile is designed as a hard macro that contains I/O driver, bump pad and power/ground trunk. All necessary interconnections are made using RDL layer, which is usually dedicated for connecting I/Os and bumps in flip-chip design. In addition to the signal bumps, we also design the power/ground-bump tile for power supply and ground connection based on the same concept. All these tiles include the indispensable ESD protection circuit commonly used in modern ICs for preventing signal and power/ground bumps from ESD damage.

Moreover, to follow the package design rules, the bump size and pitch must be taken into consideration when planning I/O-bump tiles. In legalizing the I/O-bump tile placement, we propose an I/O-row scheme without adding extra routing layers. Figure 6 shows that each row is constructed with RDL layer. The minimum width/height of tile and I/O-row are designed to follow the basic flip-chip design rules of bump size/pitch. Once the tile is placed on the I/O-row, based on this design, only the bump spacing rules within a row should be checked. It simplifies and facilitates the task for resolving the placement legalization issue. In this article, the tile size of 160×80 (um^2) is provided as an example. It is not unified for all flip-chip designs and should be determined according to the minimum flip-chip design rules. Such scheme makes the RDL routing trivial and creates the single and unique interface between chip and package by combining I/O with bump, thus actually implementing the chip and package design in parallel.

3. I/O-BUMP PLANNING PROBLEM IN CONCURRENT DESIGN

With I/O-bump tile design, we believe that consequent I/O-bump planning can provide a fairly good starting point for both chip-level and package-level design in proposed design flow. We thus create the package-aware I/O-bump planning problem as the assignment problem that assigns the I/Os and bumps according to given package ballplan. Here are the detailed problem definitions.

Input:

—The given net names and locations for n package balls.
—The design rules for chip and package.

[2]In this work, we only give an example to have the size of I/O. In general case, as long as the I/O is well designed and its size is larger than bump pitch, such I/O-bump tile design would be no limitation on implementation. However this may affect the design flexibility when I/O size is smaller than bump pitch.

Output:

—The assigned net names and locations for p I/Os and p bumps ($p = n$).
—The preliminary assignment provided for chip-level core-I/O placement and package-level bump-ball routing.

Assignment criteria (considering the flyline between bumps and balls):

—Minimum possible routing layer (minimum net crossing number).
—Minimum timing delay (minimum total net length).
—Minimum signal skew (minimum sum of length difference/deviation on each net).

Instead of calculating the actual wirelength, we use the flyline (net) length, which we choose to use the Euclidean distance between the bump and ball as the expected routing length, 45 degree distance can be employed as well. This estimation can be obtained easily and evaluates the performance of I/O-bump planning methods efficiently, it thus facilitating and speeding up the optimization works. Based on the same intention, our approach uses the flyline (net) crossing number to estimate the true crossing number of routing. The consistency between the flyline estimation and results of performing detailed package routing has been ensured in Meister et al. [2008]. More accurate wirelength estimation for detailed package routing (e.g., 45 degree routing) can be found in Sarrafzadeh and Wong [1992]. In the next section, we show how we arrange I/Os and bumps simultaneously with the specific I/O-bump tiles invention.

4. PACKAGE-AWARE I/O-BUMP PLANNING METHODS

According to the package ball locations (given pin-out/ballplan), here we employ two intuitive methods and one matching-based heuristic to plan I/O-bump tiles. Those methods help us achieve very efficient evaluation for all possible chip-package configurations. Each of them is distinguished by different design goals. Aiming at minimizing package routing layer to reduce package cost, the first heuristic is used to obtain a zero net crossing design. To mitigate the parasitic effect on routing nets and facilitate the wirelength matching, the second heuristic focuses on shortening the net length and minimizing the length deviation. As for the matching-based assignment, it balances all the aforementioned design requirements simultaneously.

Before planning I/O-bump tiles, we partition the whole package into four sectors: north, west, south and east similar to the works in Fang et al. [2007] and Fang et al. [2009]. In general design flow, designer specifically arranges the initial I/O location based on some particular purposes before running chip-level placement. In this study, according to the given package ball plan, the initial placement of corresponding I/O-bump tiles is randomly generated in each sector, as shown in Figure 7. In real design flow, the initial placement can be also given from the design team, and we can use our approaches to refine it. After that, each planning method mentioned above starts at the east sector. While I/O-bump tiles are planned within this sector, the planning method will be iteratively applied until all sectors' tiles are assigned. Once the randomly generated tile locations have some issues with placing core cells or macros, these initial locations will be regenerated easily and the I/O-bump tiles will be re-planned speedily by our approach.

Aiming at different design goals, this work provides two intuitive methods and one matching-based heuristic to plan I/O-bump tiles. These methods efficiently evaluate all possible chip-package codesign configurations. Among these methods, the SORT minimizes the net crossing, and the GREEDY shortens the net length and reduces the length deviation. As for the matching-based assignment, it balances all the aforementioned design requirements simultaneously. Below we describe how these methods work in more details.

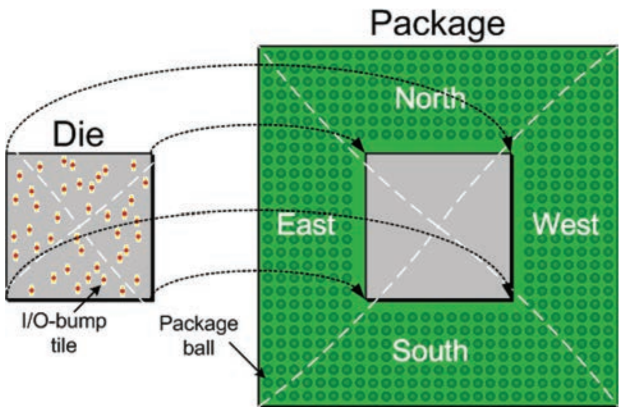

Fig. 7. The I/O-bump tiles placement regions. The whole package will be partitioned into four sectors and the initial placement of corresponding I/O-bump tiles will be given within each sector.

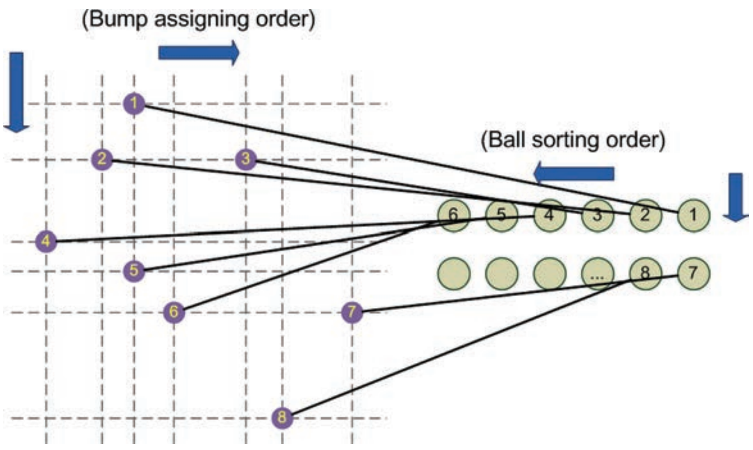

Fig. 8. Double sorting for planar planning. It sorts the I/O-bump tiles and produces the proper order by referring to the order of balls.

4.1. Double Sorting for Planar Planning

To achieve noncrossing (planar) routing results from die bumps to package balls, we propose a heuristic method called $SORT$. This method sorts the I/O-bump tiles and produces their proper order in accordance with the order of balls, thus resulting in zero net crossing by monotonic package routing.[3] The detailed steps are listed below and illustrated in Figure 8 ($Order_{ball}$ and $Order_{bump}$ are the assigned serial number for balls and bumps).

Sort package balls:

```
(1) $Order_{ball} \leftarrow 0$
(2) **Repeat:**
(3)     sorting ball rows ($upper \Rightarrow lower$)
```

[3]The monotonic package routing is a routing method that routes nets from die bumps to package vias on package top layer without U-turn path. It consumes less routing resource and results in higher routing completion compared with nonmonotonic routing method [Fang et al. 2009]. More detailed theory about monotonic routing can be found in Kubo and Takahashi [2005] and Yu and Dai [1995].

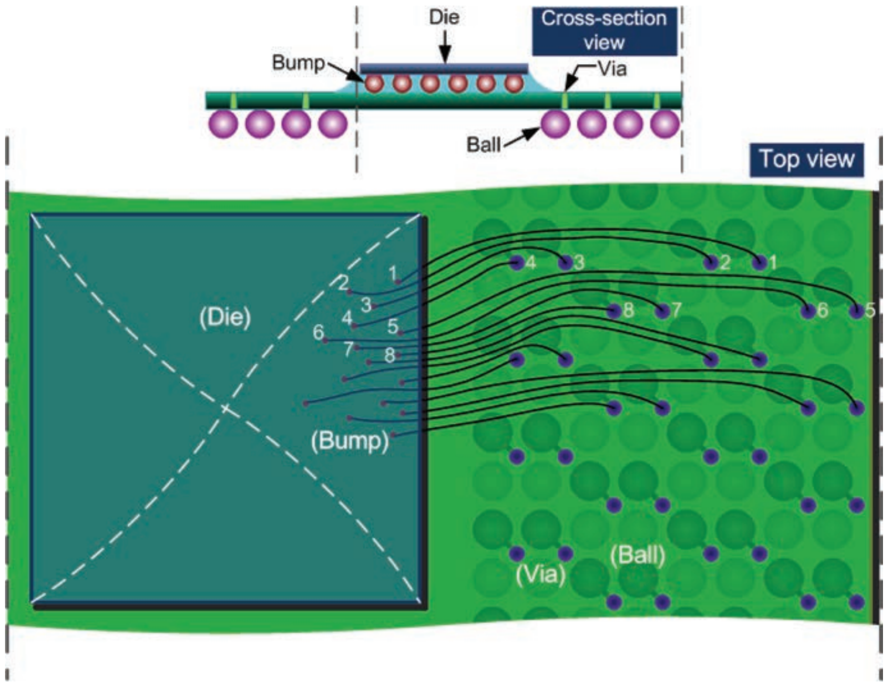

Fig. 9. Example of double sorting method. This method achieves zero net crossing when using the monotonic package routing. Such routing method routes nets from die bumps to package vias on package top layer without U-turn path.

```
(4)  Repeat:
(5)    sorting balls ($outer \Rightarrow inner$)
(6)    ordering the ball: $Order_{ball} \leftarrow Order_{ball} + 1$
(7)  Until all balls are sorted within a ball row
(8) Until all ball rows are sorted within a package sector
```

Sort I/O-bump tiles:

```
(1) $Order_{bump} \leftarrow 0$
(2) Repeat:
(3)   sorting I/O-bump tiles ($upper \Rightarrow lower$, $inner \Rightarrow outer$)
(4)   ordering the bump: $Order_{bump} \leftarrow Order_{bump} + 1$
(5) Until all I/O-bump tiles are given an order within a package sector
```

As the sorting steps shown above, the vertical sorting is to order balls from upper row to lower row. For balls located at the same row, the horizontal sorting is to order balls from outer ball to inner ball. At the second stage, the same sorting steps will be followed to order I/O-bump tiles. After that, balls and bumps that have the same numbers will be paired for connection. Figure 9 shows an example; the sorted order of package balls and I/O-bump tiles will lead to noncrossing package routing while applying the monotonic routing.

4.2. Shortening Flylines Between I/O-Bumps and Package Balls

The previous sorted method intuitively succeeds in producing a zero-crossing package routing. However, regarding the package routing task, the wirelength is another critical

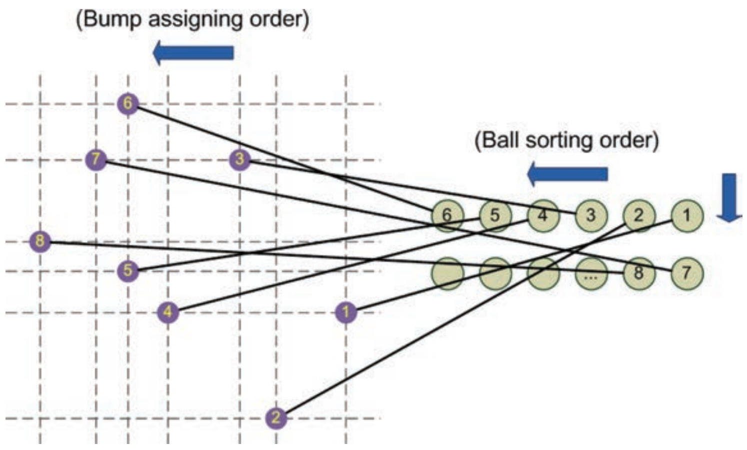

Fig. 10. Shortening flylines between I/O-bumps and package balls. It greedily chooses the shortest flyline between bumps and balls.

factor affecting its performance. Since the longer wirelength induces the larger parasitic effects, nets from bumps to balls should be routed as short as possible. Besides, to achieve impedance matching, each net should be kept in the similar wirelength. For these two objectives, we apply a greedy way to shorten the total net length and the length deviation called *GREEDY*. The main idea of this method is to choose the shortest flyline between bumps and balls greedily at the moment. This method also consists of two stages: sorting balls and greedily find shorter flylines. The first stage of process is same as that in the previous method (ball sorting), and the detailed steps in the second stage are as follows.

Choose the shortest flyline greedily:

```
(1) $Order_{bump} \leftarrow 0$
(2) given the ball order in $SORT$, starting from the first ball
(3) Repeat:
(4)   connecting the ball with all unchosen I/O-bump tiles
(5)   choosing one I/O-bump tile that can result in the
      shortest flyline
(6)   ordering the chosen bump: $Order_{bump} = Order_{ball}$,
      move to the next ball
(7) Until all balls are connected within a package sector
```

As we mentioned in Section 3, we use the flyline criteria to evaluate the performance of I/O-bump planning methods, and to facilitate the optimization works. The flyline length is the Euclidean distance between the package ball and assigned bump, and the length deviation is the difference between flyline length and the average length obtained from the *SORT* method. After applying the *SORT* heuristic, balls are ordered from outer ball to inner ball and from upper row to lower row as shown in Figure 10. According to this ball order, each ball is greedily paired with the closest I/O-bump tile thus shortening the flyline length at the moment. As a result, the *GREEDY* method can reduce the total net length, and at the same time help reduce the length deviation. Figure 11 shows the results achieved by the *GREEDY* method. Comparing with the *SORT* method, which has zero-crossing routing, the *GREEDY* method arranges the

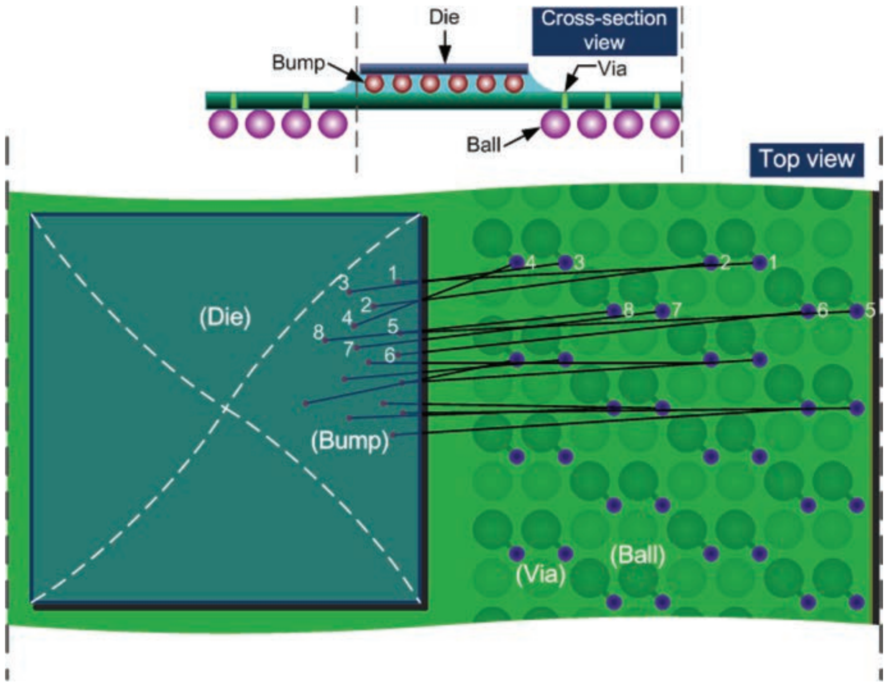

Fig. 11. Example of shortening flylines method. This method shortens the sum of length deviation on each net.

fairly different bump order. Therefore, it will inevitably cause the net crossing in package routing and increase the package design cost.

4.3. Matching-Based Assignment

For optimizing the requirements in chip-package codesign, designer must minimize the net crossing, the total net length, and the length deviation at the same time. We use the results obtained from the previous sorted method as the initial solution, and model the package-aware I/O-bump planning as a weighted bipartite matching problem, as shown in Figure 12. The assignment problem then becomes a matching problem to match the preordered ball set ($Ball_i$) (by SORT) and bump set ($Bump_j$) with the minimum edge weight (w_{ij}). The objective functions are as follows.

Minimize

$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} \cdot x_{ij}$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j = 1, \ldots, n \tag{1}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i = 1, \ldots, m \tag{2}$$

$$x_{ij} \in \{0, 1\}. \tag{3}$$

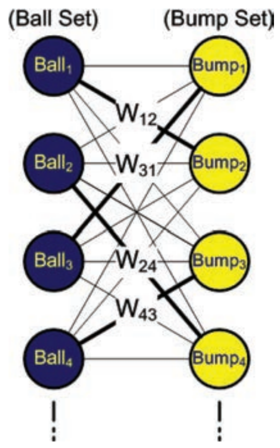

Fig. 12. Matching-based assignment $WBIPT$. This method models the package-aware I/O-bump planning as a weighted bipartite matching problem to balance severeal design requirements.

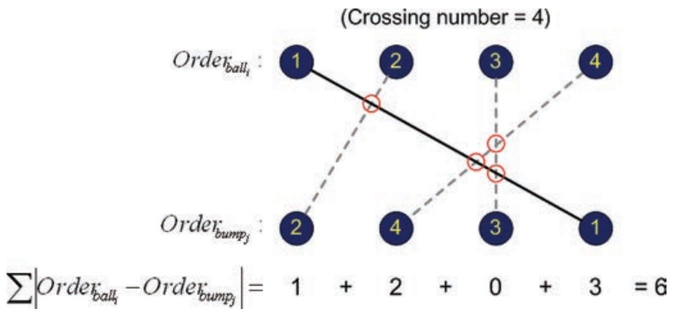
(Crossing number = 4)
$Order_{ball_i}$: 1 2 3 4
$Order_{bump_j}$: 2 4 3 1
$\sum|Order_{ball_i} - Order_{bump_j}|$ = 1 + 2 + 0 + 3 = 6

Fig. 13. Estimation of net crossing number. The estimated crossing number is larger than real one.

The element x_{ij}, a binary variable, is 1 if $Ball_i$ is assigned to $Bump_j$; otherwise, x_{ij} is 0. Variables m and n are the total number of balls and bumps respectively ($m = n$). The edge weight w_{ij} is formulated below:

$$w_{ij} = \alpha \cdot Diff_{ij} + \beta \cdot |l_{ij} - AvgLength|, \tag{4}$$

where $Diff_{ij}$ $(= |Order_{ball_i} - Order_{bump_j}|)$ is obtained through directly subtracting the order of $Bump_j$ from that of $Ball_i$, and therefore calculating the upper bound of flyline crossing number as shown in Figures 13 and 14. $AvgLength$ is the average length obtained from the $SORT$ method and l_{ij} is the Euclidean flyline length as mentioned in previous subsection. It can be seen that the edge weight consists of the net crossing (first term) and the length deviation (second term). As for the user-defined parameters α and β, they are used to adjust the importance of these two terms. Since the net length (l_{ij}) is also included in the second term of the edge weight, the weighted bipartite matching ($WBIPT$) method reassigns the order of I/O-bump tiles and fulfils all assignment criteria through carefully specifying these user-defined parameters, including the wirelength due to the summation of l_{ij}. Moreover, such reassignment has more balanced net crossing and net length deviation, $WBIPT$ method optimizes the I/O-bump planning comparing with the previous heuristics.

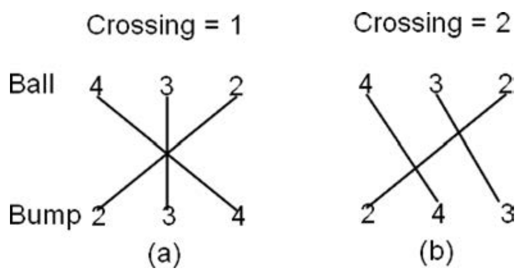

Fig. 14. Our crossing's definition to show that our estimation is the upper bound of crossing number.

Table III.
The Summary of Six I/O-Bump Planning Methods

	I/O-Bump Planning Method
♯1	*SORT*
♯2	*GREEDY*
♯3	*WBIPT* ($\alpha = 5000$, $\beta = 1.0$)
♯4	*WBIPT* ($\alpha = 2500$, $\beta = 1.0$)
♯5	*WBIPT* ($\alpha = 1000$, $\beta = 1.0$)
♯6	*WBIPT* ($\alpha = 500$, $\beta = 1.0$)

5. EXPERIMENTAL RESULTS

5.1. The Effectiveness of I/O-row Methodology

We have implemented our methodologies in C++, and the platform is on Intel Pentium IV 3.20GHz with 1.5GB memory. We have obtained six industrial chipset designs as our test cases to demonstrate the effectiveness of I/O-row based scheme on shrinking die size (or increasing I/O number). According to the core cell floorplanning results, the peripheral I/Os originally used in those designs are replaced with our I/O-bump tiles while keeping the utilization rate of core cells the same. Without sacrificing the wiring and placement resource of core cells, Table I shows the benefit of implementing I/O-bump tiles in I/O-pad limited designs ($d1$ and $d4$ to $d6$). For this kind of design, the number of its core cells is usually small, and its die size is mainly determined by the total occupied area of I/Os. Therefore, the flexibility of using area-array I/O-bump tile can help to reduce its die size compared with the style of locating peripheral I/Os at fixed site. On the contrary, the core limited designs ($d2$ and $d3$) that have an enormous number of core cells will not have such advantage while inserting I/O-bump tiles into their congested core area, and area increase is shown due to area-I/O application causing I/O-core integrated placement in such congested area.

5.2. I/O-Bump Planning Study

Our industrial cases listed in Table I are all chipset designs. Since they have the similar ball plan for the same I/O group, we choose two of these designs to measure the performance of planning methods. Design $d4$, which has the least I/O number among I/O-pad limited designs is tested and compared with the design $d5$ whose I/O number is the most. Table III summarizes all applied algorithms. The methods $SORT$ and $GREEDY$ are two heuristic methods described in Section 4, and the last four algorithms are applying $WBIPT$ with specific user-defined parameters. As we have described in Section 4, these methods have different characteristics, such as lower package design cost, less parasitic effect and better length matching. Like the intention stated in Section 3, to evaluate the performance of I/O-bump planning methods

Table IV.
The results on test case $d4$ (has 200 I/Os), when the initial I/O-bump tiles are randomly assigned ($RAND$) and uniformly arranged at fixed location ($UNIF$) in each sector

		Flyline criteria						Total
		Net crossing		Net length		Length deviation		runtime
		No.	Imp. (%)	Total (*um*)	Imp. (%)	Total (*um*)	Imp. (%)	(*sec*)
$RAND$	♯1a	0	100.00	2022540	–	485091	–	< 1.0
	♯2a	341	–	1998210	1.20	299349	38.29	< 1.0
	♯3a	0	100.00	2022540	0.00	485091	0.00	< 1.5
	♯4a	8	97.65	2016260	0.31	432443	10.85	< 1.5
	♯5a	30	91.20	2011210	0.56	391516	19.29	< 1.5
	♯6a	75	78.01	2006650	0.79	359033	25.99	< 1.5
$UNIF$	♯1b	0	100.00	2122570	–	424324	–	< 1.0
	♯2b	462	–	2107850	0.69	326473	23.06	< 1.0
	♯3b	1	99.78	2122590	0.00	414079	2.41	< 1.5
	♯4b	4	99.13	2121120	0.07	393317	7.31	< 1.5
	♯5b	11	97.62	2115890	0.31	372148	12.30	< 1.5
	♯6b	47	89.83	2112080	0.49	338260	20.28	< 1.5

("–" stands for the baseline)

Table V.
The results on test case $d5$ (has 628 I/Os), when the initial I/O-bump tiles are randomly assigned ($RAND$) and uniformly arranged at fixed location ($UNIF$) in each sector

		Flyline criteria						Total
		Net crossing		Net length		Length deviation		runtime
		No.	Imp. (%)	Total (*um*)	Imp. (%)	Total (*um*)	Imp. (%)	(*sec*)
$RAND$	♯1c	0	100.00	5473480	–	1432024	–	< 2.0
	♯2c	1056	–	5416680	1.04	720120	49.71	< 2.0
	♯3c	0	100.00	5473480	0.00	1432024	0.00	< 5.5
	♯4c	32	96.97	5461600	0.22	1209704	15.52	< 5.5
	♯5c	140	86.74	5447080	0.48	996644	30.40	< 5.5
	♯6c	376	64.39	5437040	0.67	920888	35.69	< 5.5
$UNIF$	♯1d	0	100.00	5813320	–	1645728	–	< 2.0
	♯2d	1432	–	5775240	0.66	1410436	14.30	< 2.0
	♯3d	0	100.00	5813320	0.00	1645728	0.00	< 5.5
	♯4d	24	98.32	5802480	0.19	1505192	8.54	< 5.5
	♯5d	32	97.77	5794920	0.32	1480016	10.07	< 5.5
	♯6d	148	89.66	5786320	0.46	1374196	16.50	< 5.5

("–" stands for the baseline)

efficiently and facilitate the optimization works, all these characteristics are identified with three terms: Net crossing, Total net length, and Length deviation in experimental results. To demonstrate the features of our I/O-row based scheme on improving chip-package codesign requirements, the experimental results are achieved by adopting two different methods to place initial I/O-bump tiles. In Table IV and Table V, a and c denote that initial locations of I/O-bump tiles are randomly generated in each sector as long as the bump spacing meet design rules. Whereas b and d denote that initial I/O-bump tiles are uniformly arranged with fixed bump spacing.

The experimental results of test cases $d4$ and $d5$ shown in Table IV and Table V are consistent, and are independent of the I/O number. They both are fairly reassuring and encouraging. The $SORT$ (♯1x) heuristic works toward obtaining the zero net-crossing in monotonic package routing through ordering I/O-bump tiles. Comparing with the

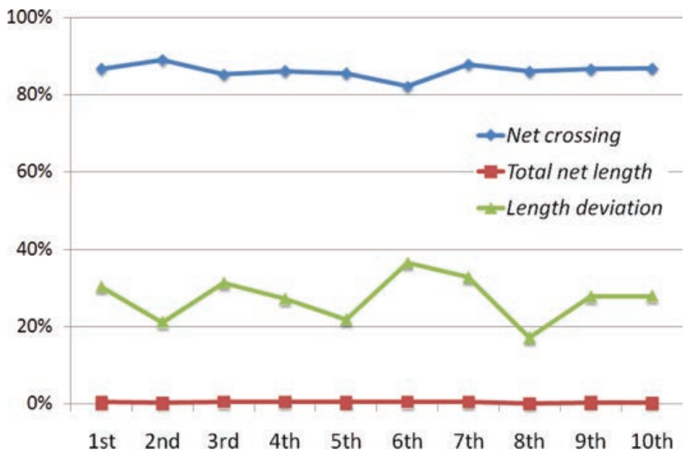

Fig. 15. Improvement curves of performance metrics plotted by randomly generating the initial I/O-bump tiles and applying method ♯5*c*. After running ten times, the average improvements on net crossing, total net length and total length deviation are 86.22%, 0.47% and 27.43% respectively.

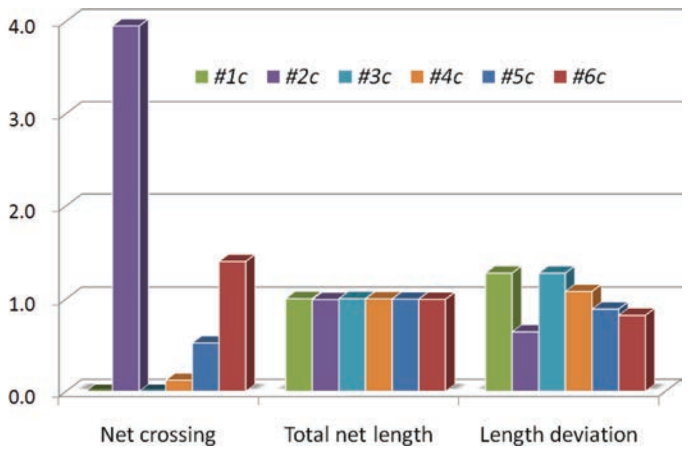

Fig. 16. Normalization of performance metrics obtained by adopting methods ♯1*c* to ♯6*c*. The assignment algorithms (♯3*c* to ♯6*c*) can perform the tradeoffs between the net crossing and the length deviation by specifying the suitable user-defined parameters (α and β).

other methods, the $GREEDY$ (♯2x) heuristic succeeds in shortening the total net length and the length deviation for flylines. Furthermore, the assignment algorithms $WBIPT$ (♯3x to ♯6x) balance the net crossing and the length deviation. Comparing the results in Table IV and Table V, the randomly assigned initial I/O-bump tiles will produce the shorter net length and the length deviation among all planning methods due to higher flexibility, and the uniformly arranged ones will cause smaller net crossing number thus reducing package routing layers.

To ensure that our random manner of generating the initial I/O-bump tiles will not produce the worse results, we apply method ♯5c ten times and plot the improvement curves of performance metrics. Figure 15 shows that the average improvements on net crossing, total net length and total length deviation are 86.22%, 0.47% and 27.43% respectively. As a result, the method $WBIPT$ with appropriate parameters can certainly optimize the I/O-bump tiles, even though their initial locations are not assigned by a deterministic algorithm.

Figure 16 is made by normalizing those performance metrics with their average value. The results show that we can specify the appropriate user-defined parameters (α and β) to determine the priority of the net crossing and the length deviation according to the design requirements, and to obtain a fairly good point to continue with the detailed planning/routing. In other words, the results can be utilized as preliminary feasibility studies for all possible chip-package configurations.

6. CONCLUSION AND FUTURE WORK

In order to avoid much longer turnaround time with package and system houses, we study the realization of area-array I/O design methodology, emphasizing on package design awareness. With our setup in the design of I/O-bump tile and I/O-row based scheme, we further develop a chip-package concurrent design flow. Due to the early phase of I/O-bump planning study, faster convergence can be expected from some encouraging results.

For area-array I/O planning, our future work will focus on dealing with the differential signals and taking account of the power/ground-to-signal ratio in planning methodology, which are two major issues of maintaining signal integrity and power integrity for high-speed system.

ACKNOWLEDGMENT

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

REFERENCES

Caldwell, A., Kahng, A. B., Mantik, S., and Markov, I. L. 1998. Implications of area- array i/o for row-based placement methodology. In *Proceedings of the IEEE Symposium on IC/Package Design Integration*. 93–98.

Chang, C.-Y. and Chen, H.-M. 2008. Design migration from peripheral asic design to area-i/o flip-chip design by chip i/o planning and legalization. *IEEE Trans. VLSI Syst. 16*, 1, 108–112.

Chen, H.-M., Liu, I.-M., and Wong, M. D. F. 2006. I/O clustering in design cost and performance optimization for flip-chip design. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 25*, 11, 2552–2556.

Fang, J.-W. and Chang, Y.-W. 2008. Area-i/o flip-chip routing for chip-package co-design. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 518–522.

Fang, J.-W., Hsu, C.-H., and Chang, Y.-W. 2009. An integer-linear-programming-based routing algorithm for flip-chip designs. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 28*, 1, 98–110.

Fang, J.-W., Lin, I.-J., Chang, Y.-W., and Wang, J.-H. 2007. A network-flow based rdl routing algorithms for flip-chip design. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 26*, 8, 1417–1429.

Fontanelli, A., Arrigoni, S., Raccagni, D., and Rosin, M. 2002. System-on-chip (SoC) requires ic and package co-design and co-verification. In *Proceedings of the IEEE Custom Integrated Circuits Conference*. 319–322.

Kubo, Y. and Takahashi, A. 2005. A global routing method for 2-layer ball grid array packages. In *Proceedings of the International Symposium on Physical Design*. 36–43.

Maheshwari, V., Darnauer, J., Ramirez, J., and Dai, W. W.-M. 1995. Design of fpgas with area i/o for field programmable mcm. In *Proceedings of the ACM International Symposium on Field-Programmable Gate Arrays*. 17–23.

Meister, T., Lienig, J., and Thomke, G. 2008. Novel pin assignment algorithms for components with very high pin counts. In *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe*. 837–842.

Pascariu, G., Cronin, P., and Crowley, C. 2003. Next generation electronics packaging utilizing flip chip technology. In *Proceedings of the IEEE International Electronics Manufacturing Technology Symposium*. 423–426.

Sarrafzadeh, M. and Wong, C.-K. 1992. Hierarchical steiner tree construction in uniform orientations. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 11*, 9, 1095–1103.

Sheth, K., Sarto, E., and McGrath, J. 2006. The importance of adopting a package-aware chip design flow. In *Proceedings of the ACM/IEEE Design Automation Conference*. 853–856.

Tan, C., Bouldin, D., and Dehkordi, P. 1997. Design implementation of intrinsic area array ics. In *Proceedings of the 17th Conference on Advanced Research in VLSI*. 82–93.

Wang, J., Muchherla, K. K., and Kumar, J. G. 2004. A clustering based area i/o planning for flip-chip technology. In *Proceedings of the 5th International Symposium on Quality Electronic Design*. 196–201.

Xiong, J., Wong, Y.-C., Sarto, E., and He, L. 2006. Constraint driven I/O planning and placement for chip-package co-design. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 207–212.

Yu, M.-F. and Dai, W. W.-M. 1995. Single-layer fanout routing and routability analysis for ball grid arrays. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 581–586.

Received July 2010; revised January 2011, August 2011, March 2012, October 2012; accepted November 2012