行政院國家科學委員會補助專題研究計畫期中報告

一個全新的晶片-封裝-印刷電路板共同設計與共同最佳化方法（第二年）

***Abstract*—During the early design stages, I/O and bump layout should be evaluated to optimize design cost and to avoid product failures. In this report, the concurrent chip-package codesign flow has been proposed for solving pressing I/O and bump planning problem. On the basis of planning area-array ICs, we firstly design the I/O-bump tile which integrates I/O and bump into a hard macro with the considerations of I/O power connection and electrostatic discharge (ESD) protection. Then we propose an I/O-row based scheme to place I/O-bump tiles with existed metal layers. By such a scheme, it not only reduces efforts at redistribution layer (RDL) routing and package design rule check (DRC), but also insures that the proposed codesign flow can be applied thus theoretically speeding up the design convergence from weeks to days. In our methodology, three intuitive attempts are proposed for package-aware I/O-bump planning. Such planning methods provide the preliminary study of performance metrics in designing the interface between chip and package, including net crossing, total wirelength and length difference/deviation. The experimental results show that our methodologies reduce the die size of I/O-pad limited ICs without sacrificing the utilization rate required for core cell placement.**

I. 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 I/Os are connected with package balls by using wire-bonding process. Whereas, this planning method always requires larger die size to accommodate I/Os and pads, and degrades the signal integrity and power integrity for off-chip signaling due to parasitics and coupling effects [1] [2]. On the other hand, the area-array I/O planning using the flip-chip technique overcomes those drawbacks in the peripheral style. 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 ICs in deep-submicrometer (DSM) environment [3] [4].

A. Previous Works

Regarding the current flip-chip area-array ICs, two different area-array I/O regimes are widely utilized in industry: the extrinsic area-array I/O and intrinsic area-array I/O [5]. In [6], Maheshwari *et al.* 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 uses a dedicated redistribution layer (RDL) to redistribute nets from peripheral I/Os to area-array bumps located in the center of core area. It migrates package design from wire-bonding 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 distributed 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. As for 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 giving the flexibility in core-I/O placement.

Several works [7] [8] [9] [10] [11] proposed methodologies to deal with flip-chip area-array ICs. Fang *et al.* applied the network-flow-based and the integer-linear-programming-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 [7] [8]. On the other hand, two recent works focused on planning and placing intrinsic area-array I/Os. [9] 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 designer. In [10], based on integer linear programming (ILP), Xiong *et al.* 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 previous work proposed a network-flow based multi-RDL router for the intrinsic area-array flip-chip ICs [11]. 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.

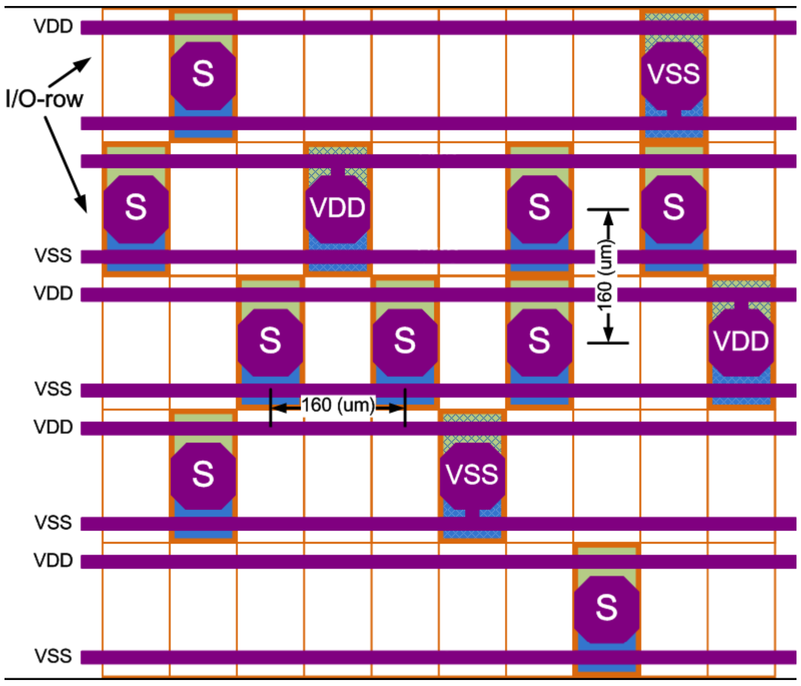

Fig. 1. The I/O-row based I/O-bump planning scheme. It shows the width/height of tile and I/O-row are designed for satisfying the bump size/pitch. This scheme therefore simplifies the placement legalization of I/O-bump tiles.

All the aforementioned researches achieve some notable results. However, more considerations need to be taken to apply their methods in the hostile chip-package codesign environment:

- These approaches assume that bumps are arranged in fixed array locations with unique spacing, they then design the area-array ICs by connecting bumps and I/Os with RDL routing or planning the I/O placement with cost functions and constraints. Unfortunately, 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 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 on the package ballplan given and optimized for PCB design. Without considering package ballplan, the I/Os and bumps will possibly lead to complicated package substrate routing, even failed package design [12] [13].
- The conventional design flow, which is IC-driven [14], has an inevitable shortcoming. As shown in Fig. 2, the initial I/O placement is usually assigned by chip designer according to the core cell floorplanning results. After that, it iteratively optimizes the locations of core cells and I/Os until the final placement meets design requirement such as the minimum total wirelength. Finally, this bottom-up design flow starts to process the bump planning and RDL routing, then finishes the bump placement and provides for package 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.

B. Our Contributions

In order to overcome these drawbacks, here we propose a feasible area-I/O design methodology. The contributions presented in this report are as follows:

- We propose a concurrent codesign flow as shown in Fig. 3. Comparing with the sequential design flow (Fig. 2), the concurrent one completes the core cell and I/O placement and package routing in parallel, thus reducing the turn around time when designing chip and package.
- Through designing the specific I/O-bump tiles shown in Fig. 4, complicated RDL routing efforts can be avoided. In addition, with our innovative I/O-row based scheme shown in Fig. 1, 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.
- In this study, we propose the package-aware I/O-bump planning in our concurrent codesign flow, which consider the package ball locations and the individual objectives in chip-package codesign such as non-crossing routing, the shortest net length and the minimum length deviation among all nets.
- For I/O-pad limited design, the experimental results show that our I/O-row based scheme can effectively reduce the die size. The proposed package-aware I/O-bump planning also provides initial planning to drive concurrent codesign with design trade-offs.

The rest of the report is organized as follows. Section II introduces the row-based I/O-bump tile design. Section III defines the problem of package-aware I/O-bump planning, while Section IV describes three I/O-bump planning methods. Section V shows the experimental results, followed by the conclusion in Section VI.

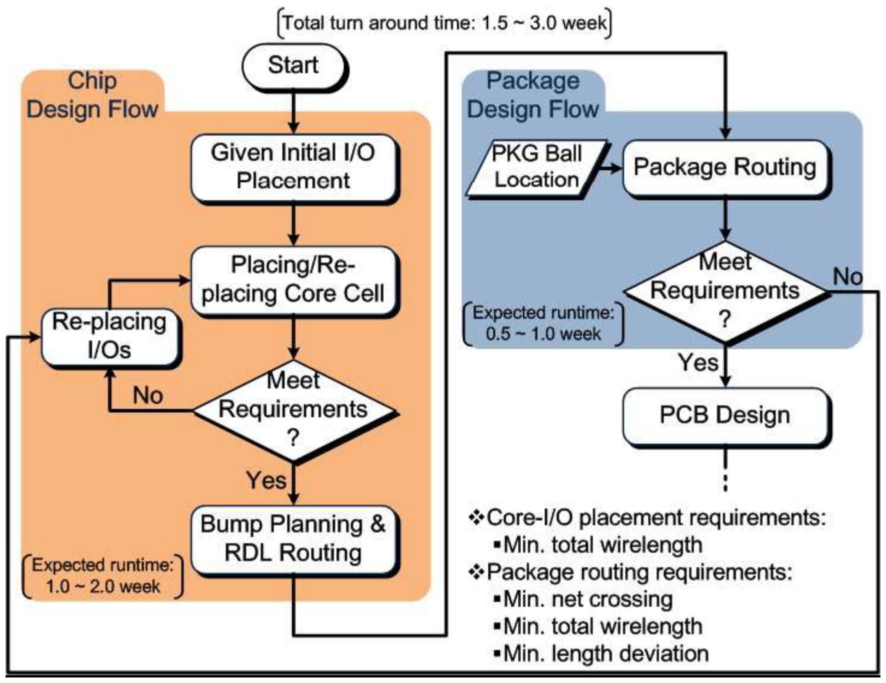
Total turn around time: 1.5 ~ 3.0 week
Start
Chip Design Flow
Given Initial I/O Placement
Placing/Re-placing Core Cell
Re-placing I/Os
No
Meet Requirements ?
Yes
Bump Planning & RDL Routing
Expected runtime: 1.0 ~ 2.0 week
Package Design Flow
PKG Ball Location
Package Routing
Meet Requirements ?
No
Yes
Expected runtime: 0.5 ~ 1.0 week
PCB Design
❖Core-I/O placement requirements:
▪Min. total wirelength
❖Package routing requirements:
▪Min. net crossing
▪Min. total wirelength
▪Min. length deviation

Fig. 2. 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.

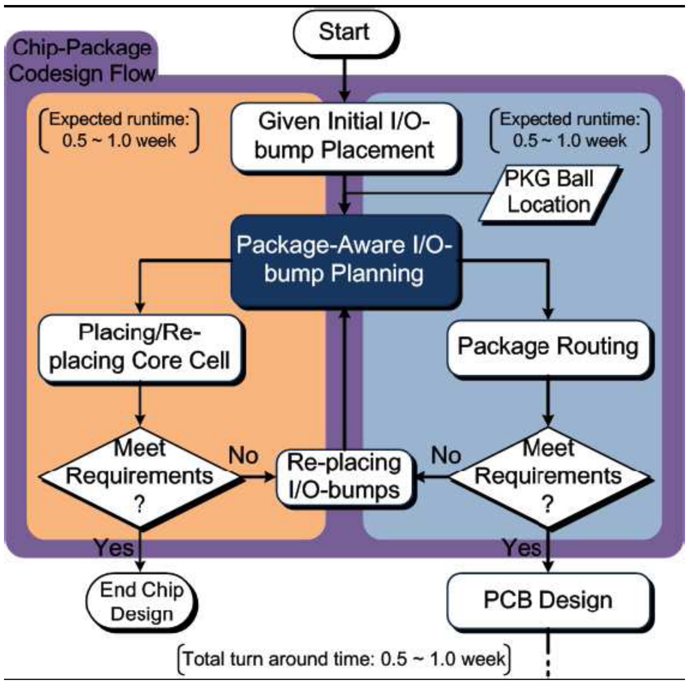
Chip-Package Codesign Flow
Start
Expected runtime: 0.5 ~ 1.0 week
Given Initial I/O-bump Placement
Expected runtime: 0.5 ~ 1.0 week
PKG Ball Location
Package-Aware I/O-bump Planning
Placing/Re-placing Core Cell
Package Routing
Meet Requirements ?
No
Re-placing I/O-bumps
No
Meet Requirements ?
Yes
Yes
End Chip Design
PCB Design
Total turn around time: 0.5 ~ 1.0 week

Fig. 3. The proposed concurrent chip-package codesign 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.

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

In order to implement our concurrent chip-package codesign flow, one of the major constructs is to integrate the I/O and bump into a specific tile. It consequently provides all information needed in both chip-level core-I/O placement and package-level bump-ball routing, while accomplishing the I/O-bump tiles planning. As shown in Fig. 4, each I/O-bump tile is designed as a hard macro which contains I/O driver, bump pad and power/ground trunk. All necessary interconnections are made using RDL layer 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. They both include the indispensable electrostatic discharge (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. Fig. 1 shows that each row is constructed with RDL layer. The width/height of tile and I/O-row are designed to follow the design rules of bump size/pitch (ex. 80 *um*/160 *um*). Once the tile is placed on the I/O-row, based on this design, only the rules within a row should be checked. It simplifies and facilitates the task for resolving the placement legalization issue. Although the I/O-row based scheme is not flexible for alternating core and I/O placement methodologies [1], this 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-package codesign.

III. I/O-BUMP PLANNING PROBLEM IN CONCURRENT CODESIGN

With I/O-bump tile design, such I/O-bump planning can provide a fairly good starting point for both chip-level and package-level design in proposed codesign flow. We define the package-aware I/O-bump planning problem as the assignment problem which assigns I/Os and bumps according to the distribution of package balls. Here are the detailed problem definitions:

Input:

- The given net names and locations for *n* package balls.
- The *p* I/Os and *p* bumps unassigned net names and locations (*p* = *n*).
- The design rules for chip and package.

Output:

- The assigned net names and locations for I/Os and bumps.
- 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):

- The minimized net crossing number.
- The minimized total wirelength.
- The minimized sum of length difference/deviation on each net.

In the next section, we show how we arrange I/Os and bumps simultaneously with the specific I/O-bump tiles invention.

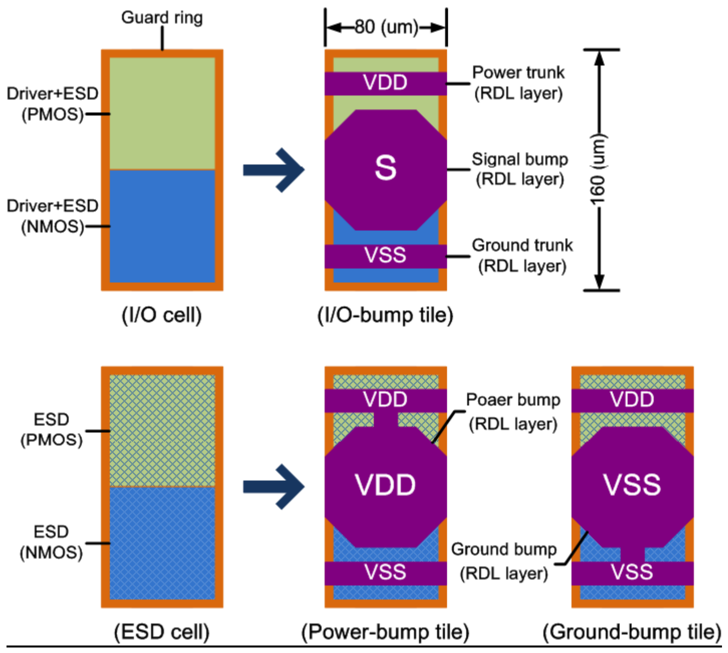

Fig. 4. The design of proposed I/O-bump tile which integrates the I/O and bump into the single and unique interface between chip and package.

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

To plan I/O-bump tiles along the package ball locations (given pin-out/ballplan), here we have performed two heuristic methods and applied one assignment algorithm. 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 wirelength deviation. As for the assignment algorithm, it balances those design requirements simultaneously. Like the works in [7] and [8], we partition the whole package into four sectors: north, west, south and east. As shown in Fig. 5, the initial placement of corresponding I/O-bump tiles will be randomly generated in each sector. After that, each I/O-bump tile planning method mentioned above starts at the east sector. While the I/O-bump tiles are planned within this sector, the package will iteratively be rotated counterclockwise and employed methodologies until all sectors' tiles are assigned.

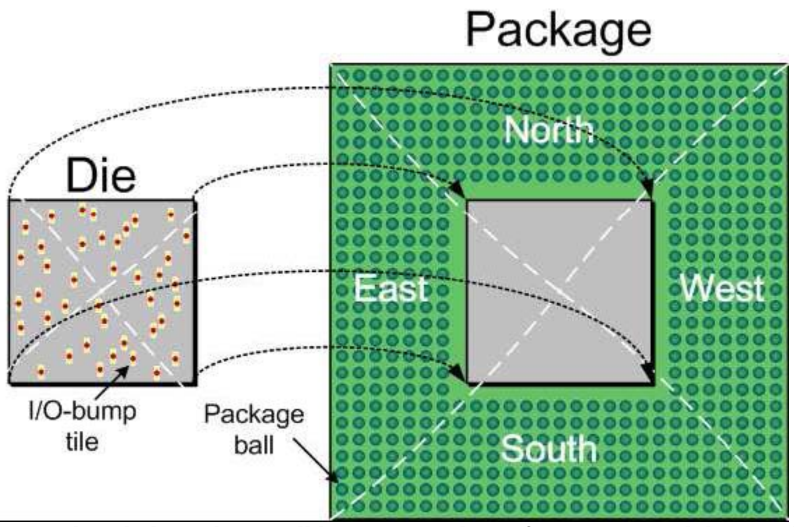

Fig. 5. The I/O-bump tiles placement regions. The whole package will be partitioned into four sectors and the initial placement of corresponding I/Obump tiles will be randomly generated within each sector.

A. Heuristic SORT: Sorting I/O-Bump Tiles

In order to obtain the non-crossing (planar) routing from die bumps to package balls, we propose a heuristic method called *SORT*. Referring to the order of balls, this method sorts the I/O-bump tiles and produces their proper order, thus resulting in zero net crossing by monotonic package routing.
The detailed steps are as follows:

```
Sort package balls:
1) $Order_{ball} \leftarrow 0$
2) **Repeat:**
3)        sorting ball rows (*top* ➔*bottom*)
4)        **Repeat:**
5)                sorting balls (*outer* ➔*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 (*top* ➔ *bottom*, *inner* ➔*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
```

$Order_{ball}$ and $Order_{bump}$ are used to assign the serial number for balls and bumps, the same numbers of ball and bump are paired for connection. Fig. 6 shows an example, the sorted order of package balls and I/O-bump tiles will lead to non-crossing package routing while applying the monotonic routing. For the tiles and balls located on other package sectors, we can also sort them as long as we rotate the package counterclockwise and implement this two-stage *SORT* heuristic.

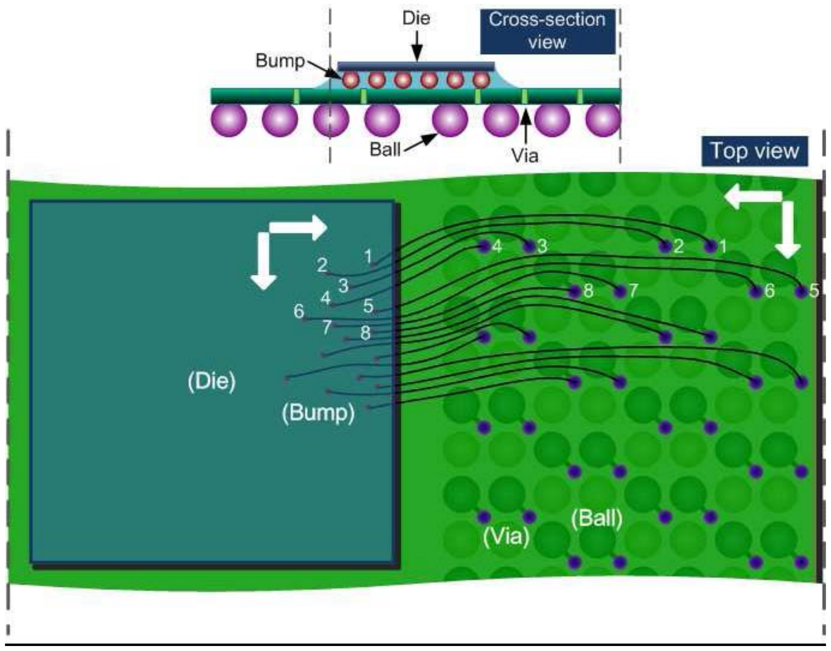

Fig. 6. The first heuristic method: *SORT*. It sorts the I/O-bump tiles and produces the proper order by referring to the order of balls, thus resulting in zero net crossing when using the monotonic package routing. Such routing method routes nets from die bumps to package vias on package top layer without U-turn path.

B. Heuristic GREEDY : Greedily Choosing the Shortest Flyline

The *SORT* method intuitively succeeds in producing a zero-crossing package routing. However, regarding the package routing task, the net length is another critical factor influencing 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 (the shortest nets and equi-length), we propose another heuristic to simultaneously shorten the total wirelength and the length deviation called *GREEDY*. The main idea of this method is to choose the shortest flyline between bumps and balls greedily. The *GREEDY* method also consists of two stages: sorting balls and greedily find shorter flylines. The first stage of process is same as that in *SORT* method (ball sorting), and the detailed steps in the second stage are listed below:

```
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 which 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
```

The flyline length is calculated with 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 *SORT* method. Fig. 7 shows the results achieved by the *GREEDY* method. It shows that each ball is greedily paired with the closed I/O-bump tile at the moment. As a result, the *GREEDY* method reduces both total wirelength and length deviation. Comparing with *SORT* method which has zero-crossing routing, the *GREEDY* method establishes the fairly different bump order. Therefore, it will inevitably cause the net crossing in package routing and increase the package design cost. The number of net crossing in *GREEDY* is calculated on swapping bump order as that in *SORT*.

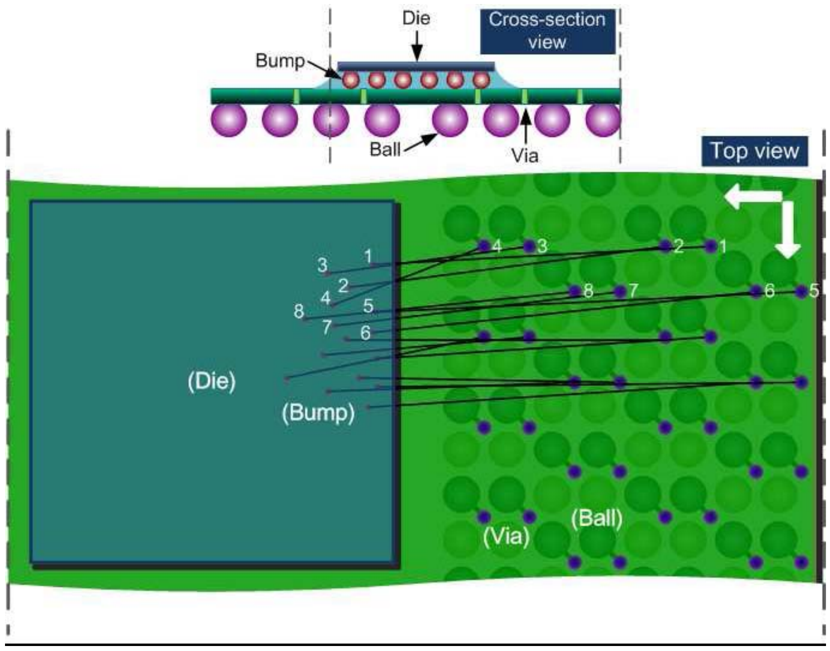

Fig. 7. The second heuristic method: *GREEDY*. By greedily choosing the shortest flyline between bumps and balls, this method shortens the total wirelength and the length deviation.

C. Algorithm WBIPT: Weighted Bipartite Matching

For optimizing the requirements in chip-package codesign, designer must minimize the net crossing, total wirelength and length deviation at the same time. We use the results obtained from the *SORT* method as the initial solution, and model the package-aware I/O-bump planning into a weighted bipartite matching problem as shown in Fig. 8. The assignment problem then becomes a matching problem to match the pre-ordered ball set (*Ball*$_i$) 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

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

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

$$x_{ij} \in \{0, 1\} \qquad (3)$$

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:

$$w_{ij} = \alpha \cdot Diff_{ij} + \beta \cdot |l_{ij} - AvgLength| \qquad (4)$$

where $Diff_{ij}$ (= $|Order_{balli}$ - $Order_{bumpj}|$) is obtained through directly subtracting the order of $Bump_j$ from that of $Ball_i$, and therefore calculating the upper bound of crossing number [15]. *AvgLength* is the average length obtained from *SORT* method and l_{ij} is the flyline length (mentioned in *GREEDY* method). It can be seen that the edge weight consists of the net crossing (first term) and the wirelength deviation (second term). In addition, the user-defined parameters α and β are used to adjust the importance of net crossing and wirelength deviation. Through implementing the weighted bipartite matching (*WBIPT*) algorithm and carefully specifying these user-defined parameters, this method reassigns the order of I/O-bump tiles. Because such reassignment has more balanced net crossing and wirelength deviation, the *WBIPT* algorithm optimizes the I/O-bump planning comparing with the previous heuristics.

Considering the given package ball locations, the I/O-bump locations planned by applying *WBIPT* algorithm will be provided in the proposed chip-package codesign flow (Fig. 3). In this codesign flow, the methodology of alternating the core cells and I/O-bump tiles placement/replacement and package routing will be iterated until the convincing design requirements are fulfilled.

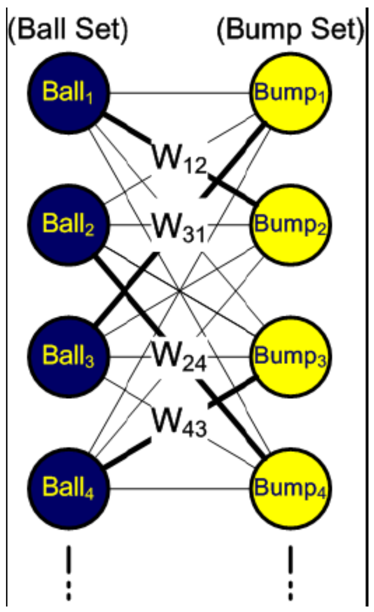

Fig. 8. The weighted bipartite matching algorithm: *WBIPT*. This method models the package-aware I/O-bump planning into a weighted bipartite matching problem.

V. EXPERIMENTAL RESULTS

We have implemented our methodologies in C++, and the platform is on Intel Pentium 4 3.20*GHz* processor with 1.5*GB* memory. Firstly, we use 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 that the die size can be reduced for those I/O-pad limited designs (*d*1 and *d*4 to *d*6). On the contrary, the core limited designs (*d*2 and *d*3) will not have this advantage due to the larger I/O-bump tiles.

To test our I/O-bump tile planning methods, we use *d*5 as the test case and summarize all algorithms in Table II. The first item *INIT* is the given initial I/O-bump locations without considering the package balls. The methods *SORT* and *GREEDY* are two heuristic methods described in Section 4.4, and the last three algorithms are applying *WBIPT* with specific user-defined parameters. As we have described in Section 4.4, those methods have different characteristics, such as lower package design cost, less parasitic effect and better length matching.

All these characteristics are evaluated with three terms: Net crossing, Total wirelength and Length deviation in Table III. The experimental results shown in Table III are fairly reassuring and encouraging. When ignoring the package design in planning I/Os and bumps, the *INIT* (#1) definitely produces the worst results comparing with all proposed package-aware I/O-bump planning algorithms. The *SORT* (#2) heuristic works toward obtaining the zero net-crossing in monotonic package routing through ordering I/O-bump tiles. Comparing with the other methods, the *GREEDY* (#3) heuristic succeeds in shortening the total wirelength and length deviation for flylines. Furthermore, the assignment algorithms *WBIPT* (#4 to #6) balance the net crossing and length deviation. Fig. 9 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 net crossing and length deviation according to the design requirements.

VI. CONCLUSION

To develop the concurrent chip-package codesign flow, in this report we have proposed a novel I/O-row based scheme to place I/O-bump tiles. Two heuristics and one assignment algorithm are also provided for package-aware I/O-bump planning. The drawbacks of previous works are therefore mitigated or eliminated. Comparing with the I/O placement produced without considering the package balls, our methodologies are realizable and provide a preliminary study of chip-package codesign requirements thus helping to improve design cost and to avoid product failures.

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 PROPOSED I/O-BUMP TILES 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 size (um^2)	Die size alteration
$d1$	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)						

TABLE II
THE SUMMARY OF SIX I/O-BUMP PLANNING METHODS.

	I/O-Bump Planning Method
$\sharp 1$	$INIT$
$\sharp 2$	$SORT$
$\sharp 3$	$GREEDY$
$\sharp 4$	$WBIPT$ ($\alpha = 1000$, $\beta = 1.0$)
$\sharp 5$	$WBIPT$ ($\alpha = 500$, $\beta = 1.0$)
$\sharp 6$	$WBIPT$ ($\alpha = 100$, $\beta = 1.0$)

TABLE III

THE EXPERIMENTAL RESULTS OF I/O-BUMP PLANNING ON TEST CASE $d5$.

	Flyline criteria					Total runtime
	Net crossing	Wirelength		Length deviation		
		Total (um)	Increase	Total (um)	Increase	(sec)
♯1	21596	6487720	1.219x	1790460	2.459x	< 1.0
♯2	0	5369640	1.009x	1297756	1.782x	< 2.0
♯3	964	5324000	–	728060	–	< 2.0
♯4	28	5358600	1.006x	1106244	1.519x	< 5.5
♯5	124	5356760	1.006x	1002125	1.376x	< 5.5
♯6	500	5353480	1.006x	931676	1.280x	< 5.5
	("–" stands for the baseline)					

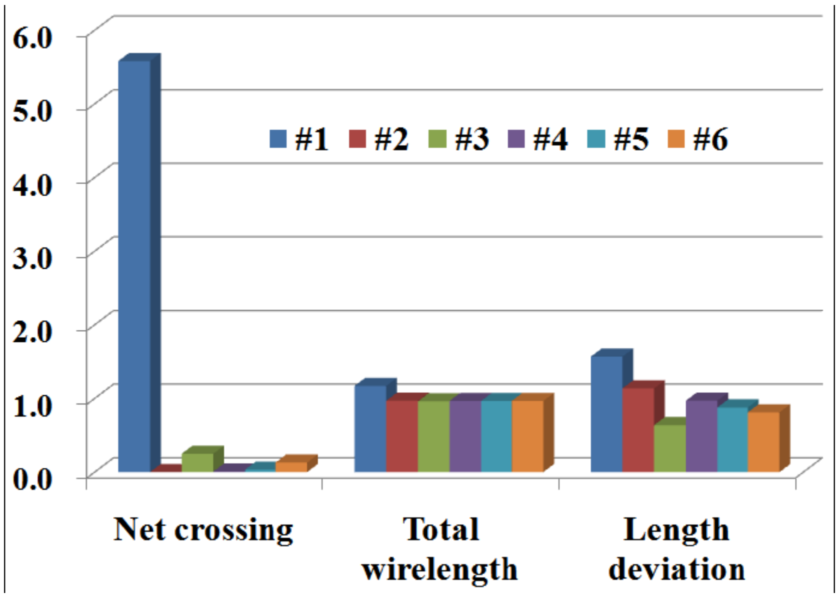

Fig. 9. The results of normalized performance metrics. It shows that the proposed methods (#2 to #6) achieve the individual objectives. The assignment algorithms (#4 to #6) determine the priority of net crossing and length deviation by specifying the suitable user-defined parameters (α and β).

REFERENCES

[1] A. E. Caldwell, A. B. Kahng, S. Mantik and I. L. Markov, "Implications of Area-Array I/O for Row-Based Placement Methodology," In *Proc. IEEE Symp. on IC/Package Design Integration*, pp. 93-98, 1998.

[2] J. Wang, K. K. Muchherla and J. G. Kumar, "A Clustering Based Area I/O Planning for Flip-Chip Technology," In *Proc. Intl. Symp. on Quality Electronic Design*, pp. 196-201, 2004.

[3] G. Pascariu, P. Cronin and D. Crowley, "Next Generation Electronics Packaging Utilizing Flip Chip Technology," In *IEEE Intl. Electronics Manufacturing Technology Symp.*, pp. 423-426, 2003.

[4] C.-Y. Chang and H.-M. Chen, "Design Migration From Peripheral ASIC Design to Area-I/O Flip-Chip Design by Chip I/O Planning and Legalization," In *IEEE Trans. on Very Large Scale Integration Systems*, vol. 16, no. 1, pp. 108-112, Jan. 2008.

[5] C. Tan, D. Bouldin and P. Dehkordi, "Design Implementation of Intrinsic Area Array ICs," In *Proc. Seventeenth Conf. on Advanced Research in VLSI*, pp. 82-93, 1997.

[6] V. Maheshwari, J. Darnauer, J. Ramirez and W. W.-M. Dai, "Design of FPGAs with Area I/O for Field Programmable MCM," In *Proc. ACM Intl. Symp. on Field-programmable gate arrays*, pp. 17-23, 1995.

[7] J.-W. Fang, I.-J. Lin, Y.-W. Chang and J.-H. Wang, "A Network-Flow Based RDL Routing Algorithms for Flip-Chip Design," In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26,

no. 8, pp. 1417-1429, Aug. 2007.

[8] J.-W. Fang, C.-H. Hsu and Y.-W. Chang, “An Integer-Linear-Programming-Based Routing Algorithm for Flip-Chip Designs,” In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 98-110, Jan. 2009.

[9] H.-M. Chen, I-M. Liu and D.-F. Wong, “I/O Clustering in Design Cost and Performance Optimization for Flip-Chip Design,” In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2552-2556, Nov. 2006.

[10] J. Xiong, Y.-C. Wong, E. Sarto and L. He, “Constraint Driven I/O Planning and Placement for Chip-package Co-design,” In *Proc. Asia and South Pacific Design Automation Conf.*, pp. 207-212, 2006.

[11] J.-W. Fang and Y.-W. Chang, “Area-I/O Flip-Chip Routing for Chip-Package Co-Design,” In *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pp. 518-522, 2008.

[12] K. Sheth, E. Sarto and J. McGrath, “The Importance of Adopting a Package-Aware Chip Design Flow,” In *Proc. Design Automation Conf.*, pp. 853-856, 2006.

[13] J. Desai and Y. Rao, “Avoid Design Issues with Package-Aware I/O Planning,” In *EE Times India*, Oct. 2006.

[14] A. Fontanelli, S. Arrigoni, D. Raccagni and M. Rosin, “System-on-Chip (SoC) Requires IC and Package Co-Design and Co-Verification,” In *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 319-322, 2002.

[15] T. Meister, J. Lienig and G. Thomke, “Novel Pin Assignment Algorithms for Components with Very High Pin Counts,” In *Proc. Design, Automation and Test in Europe*, pp. 837-842, 2008.