# 行政院國家科學委員會補助專題研究計畫期末報告

## 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（第三年）

***Abstract*—**Slow turn-around between design, package and system houses has been one of the primary concerns in semiconductor business. This is seriously lagging the development time of the system due to time-consuming interface design between chip, package and board. Due to this reason, we proposed this 3-year project, trying to study on this difficult problem and hoping to contribute more on the integration of the design flow. We have good results: we published **2 IEEE journal papers [11,12], 8 IEEE conference papers [13-20]**, some other submitted works, and **filed one patent**. Furthermore, we have attracted **one Japan well-known researcher to collaborate** with us, also co-published some works. This report mainly shows our integration in 3-year efforts. **We greatly appreciate NSC supports in this research.**

*In order to enable chip-package-board codesign to speedup the design process, we propose an approach to address this issue by efficiently planning wires for board and chip designs awareness, which includes the package pinout designation and corresponding wire planning in package and board. We model the problem as an interval intersection problem. Due to the special need in pin-out rules, we have developed an algorithm to resolve the problem. We then use some optimization techniques to further improve the objectives such as global wire congestion and length deviation. Our results show that we can perform a very efficient estimation considering those important objectives, even outperforming one recent related work in package congestion perspective.*

## I. INTRODUCTION

Nowadays larger gaps are emerging between chip, package, and board designs. More resources are spent on reaching a consensus between these three interfaces. Chip-Package-Board codesign targets at better system performance and shorter design cycles. It efficiently facilitates achieving a convergent solution. Fig. 1 shows the example of the whole platform: signals starting from I/O pads travel through many interfaces including RDL bumps, package balls, and printed circuit board (PCB). In modern VLSI designs, more than a thousand I/O pins are usually required to communicate with each other. Due to the demand for more I/Os, ball grid array (BGA) packaging has become a major interface between chip and PCB. Trade-offs between system performance and cost are therefore determined by BGA pin-out designation (also called ballout).

[1] proposed an efficient approach to automate pin-out designation for package-board codesign. Their frameworks consider signal integrity (SI), power delivery integrity (PI) and routability (RA) in pin-out block design, and achieve close-to-minimum package size while providing good signal quality. However, more requirements should be further fulfilled in pin-out designation, in addition to the performance metrics mentioned above, to facilitate the routing works between chip, package and PCB. Moreover, the design on the chip side should also be accounted for, in order to reflect important design constraints to impact package wire planning.

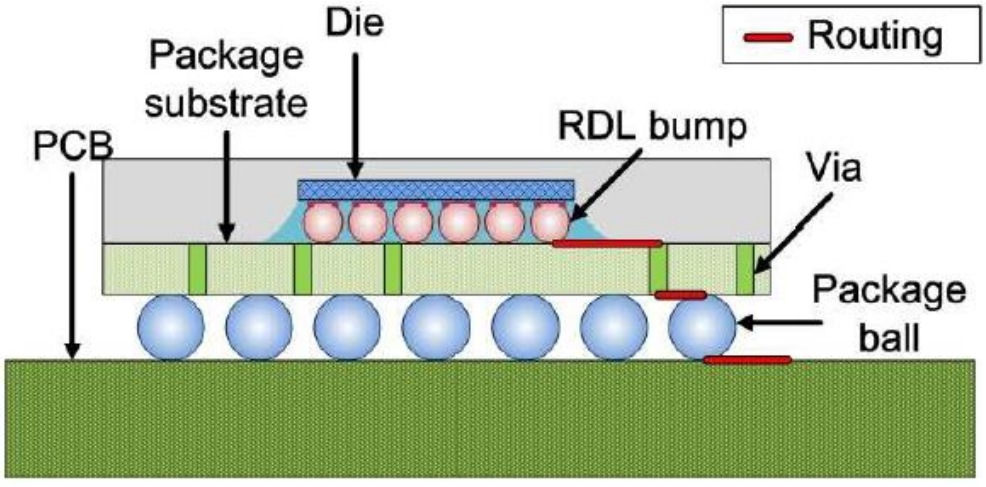


Fig. 1. The cross section of the platform: signal trace traveling through three interfaces including RDL bumps, package balls, and PCB.

*A. Previous Works*

Regarding the flip-chip designs, it is generally classified into two regimes. One is called peripheral-array I/O (PIO) where bumps are placed along the chip boundary. The other one is called area-array I/O (AIO) where bumps are placed in central area of the chip [2]. Since AIO accommodates much more number of bumps than PIO, it is more suitable for modern VLSI designs.

For AIO flip-chip designs, some sophisticated redistribution layer (RDL) routing methods are developed to connect the peripheral I/O pads with area-array bump pads. According to the pre-assigned order of I/O pads, Fang et al. applied the network-flow-based [2] and the integer-linear-programming-based [3] RDL routing algorithms for designing area-array ICs. Each of these two-stage techniques not only completes 100% routability but also reduces the total RDL wirelength and the signal skews compared with an industrial heuristic algorithm. Consequently, in order to preserve the optimized results in RDL routing, the pin-out designation must follow the ordered I/O pin sequence while designing package.

On the other hand, considering the PCB routing problem, it can also be divided into two categories. One is escape routing, which routes nets from pin terminal (ball) to component boundaries. The other one is area routing, which routes nets between component boundaries [4]. For area routing, the planar-fashioned bus routing is always preferred to control and match impedance for each high-speed signal. One approach regarding automatic bus planner for PCB was published very recently in [5]. On testing a state-of-the-art industrial circuit board, their bus planner achieves 98.5% routing completion and simultaneously assigns routing layers and nets.

However, the basic requirement of this bus planner is ordered escape routing which routes nets from balls to component boundaries with a given order. Without ordered escape routing, it is not guaranteed that the planar bus routing between components can be done [6]. To achieve the ordered escape routing, the given I/O pin sequence must be carefully considered when designating the package pin-out.

*B. Our Contributions*

The common approach usually takes weeks to rearrange pin-out, rework package substrate and PCB layout, as shown in Fig. 2, each modification of interfaces can bring costly iterations. For chip core designers, the iterations of modifying I/O pads and RDL bumps with system designers take at least one month. We hope to have a fast estimation on the resources we can use in package and board, to skip long turn-around time and

iterations between design house, package house and system house. This work proposes a feasible pinout designation which considers the ordered pin sequence in both die side and package side. These ordered pin sequences are passed to die RDL routing and PCB area routing, which are optimized by using previous works [2], [3], [4], [5], [6]. In other words, core designers can specify the preferred I/O pad ordering; system designers can specify the preferred bump pin-out designation. Our method can efficiently analyze if the preferences from both sides accommodate each other, before performing RDL routing and substrate routing. Thus the flow can be replaced by the proposed methodology, as shown in Fig. 3.

The rest of the report is organized as follows. Section II defines the problem of wire planning for 2-layer package design and PCB escape routing considering the ordered pin sequence. Section III describes the package ballout and wire planning approach; Section IV shows the optimization for various objectives to further strengthen our methodology. Section V shows the experimental results, followed by conclusion in Section VI.

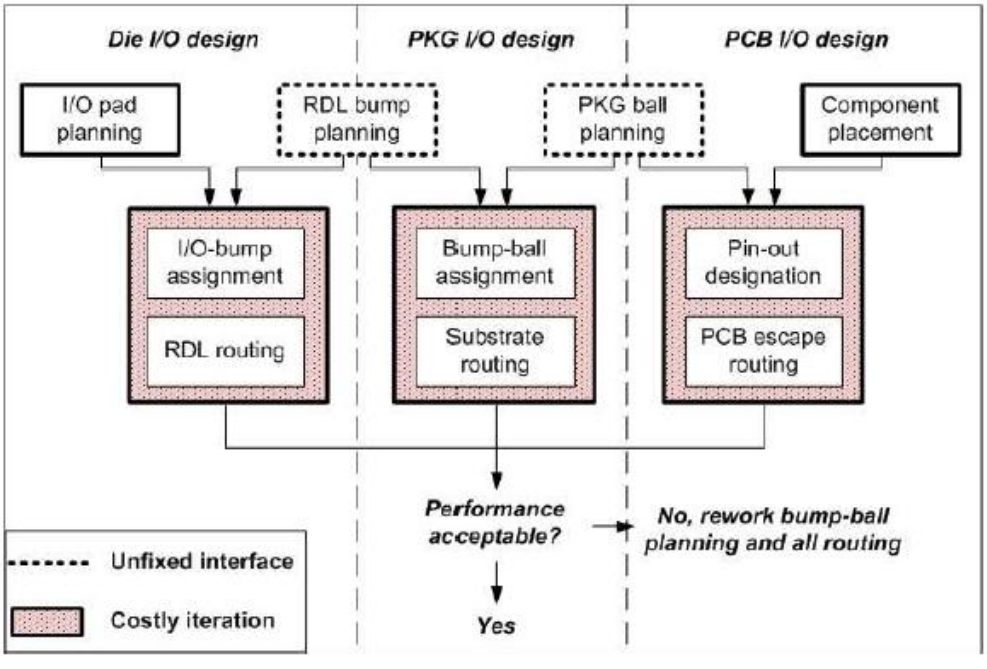


Fig. 2. Conventional flow suffers from costly rework and slow turn-around.

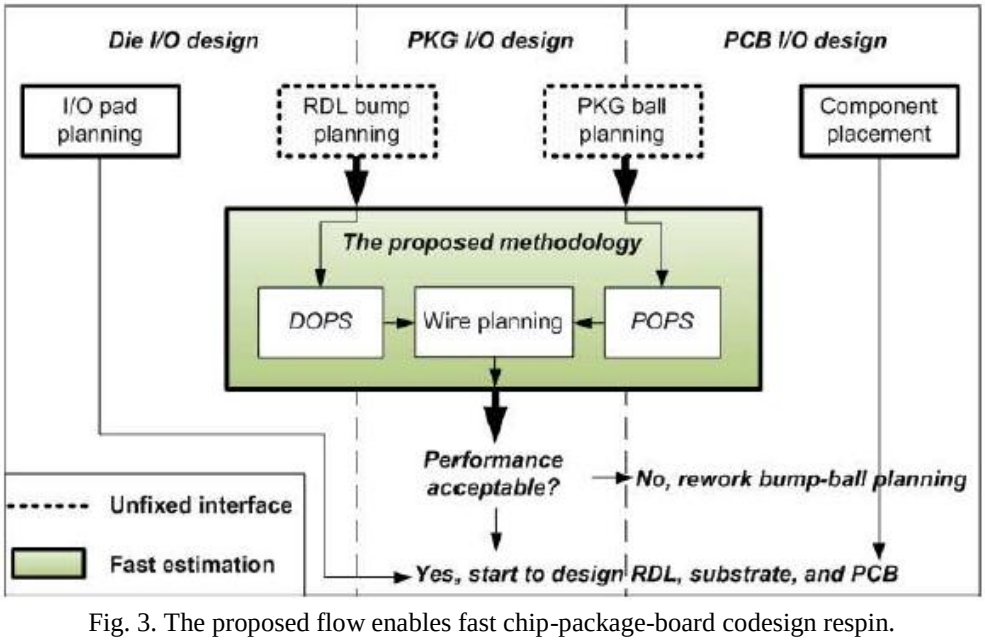


Fig. 3. The proposed flow enables fast chip-package-board codesign respin.

## II. PROBLEM DEFINITION

Fig. 4 shows our 2-layer BGA package model. Die-side ordered pin sequence (DOPS) and package-side ordered pin sequence (POPS) are the orders of I/O pins in both sides. DOPS serves as input of RDL bump assignment. The corresponding RDL routing can be optimized by applying the network-flow-based [2] or the integer-linear-programming-based algorithms [3]. POPS is regarded as input of PCB bus planner. The corresponding PCB area routing with planar bus can be well solved by [5].

In this 2-layer package model, each net (denoted as $n_i$), starting from DOPS, is connected to a via (denoted as vi) on layer-1. Each via connects to exactly one ball (denoted as $b_i$) on layer-2. Each ball then connects to POPS using PCB escape routing. In our initial assignment, $v_i$ and $b_i$ are tied up as one pin (denoted with $p_i$) and will be loosened in the post-optimization.

The ballout/pin-out designation problem is to assign $n_i$ to $v_i$ and $b_i$ and to generate the corresponding wire planning from given DOPS and POPS. The work [10] takes wire-length and wire-congestion as their objectives. Our objectives further consider package size minimization because it is important to know the trade-offs between wires and die-size. The formal definition is defined as follows:

*Input:*

- Given two sequences:
  - Die-side ordered pin sequence (DOPS).
  - Package-side ordered pin sequence (POPS).

*Output:*

- Ballout/Pin-out designation for 2-layer BGA package.
- The corresponding wire planning (monotonic global routing) for package design and PCB escape routing.

*Objectives:*

- Minimize package size (can be seen as the total number of columns used, referred to Fig. 5).
- Minimize wire congestion.
- Minimize wirelength variation/deviation for each routing layer.

There are 6 rows and 5 columns in the example shown in Fig. 5. The row number counts from top to bottom and the column number counts from left to right. For example, the locations of *p1* and *p4* are (row 3, col 5) and (row 1, col 1) respectively. Note that each route on package layer-1 is composed of two segments: 1st layer routing and 1st layer plating lead. A plating lead is redundant for operation and is usually used to reduce fabrication cost [8]. Plating leads are not plotted in the following figures; however, they are evaluated as normal wires in cost evaluation (shown in Section IV.C).

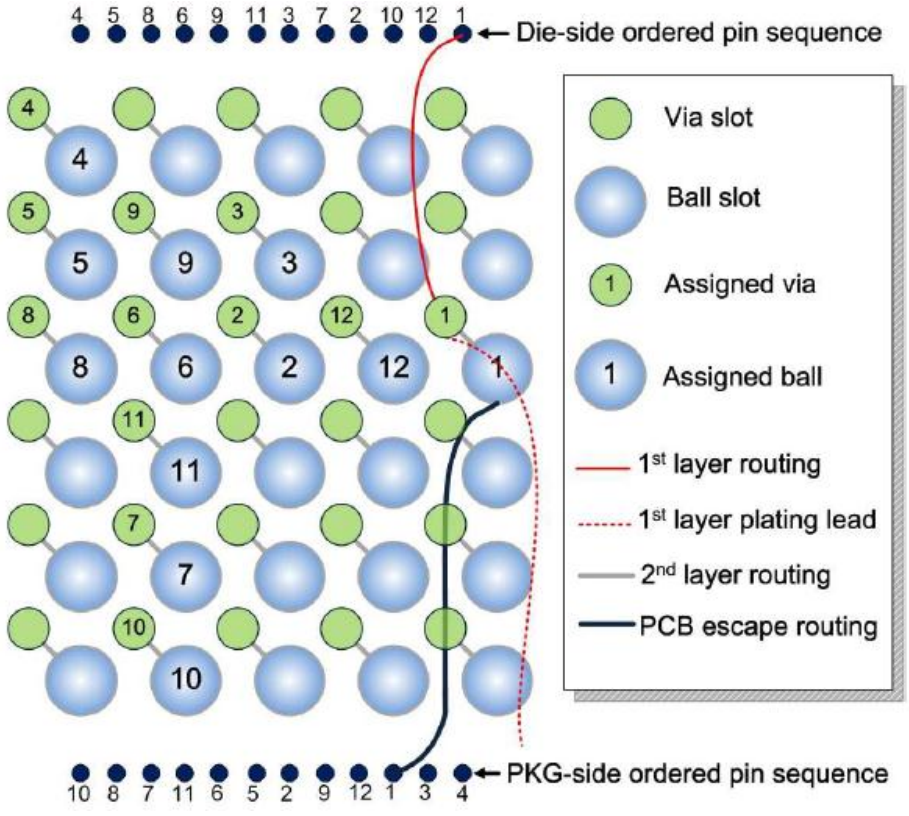


Fig. 5. Problem Illustration. Each net is designated to a via and a ball; the corresponding wire planning of n1 is plotted. vi and bi are tied up to find initial solution. The numbers inside via and ball slots are initial solution for these two ordered pin sequence.

## III. PACKAGE PIN-OUT AND CORRESPONDING WIRE PLANNING

### *A. Monotonic Global Routing in Wire Planning*

A route, from die-side pin to via and from ball to package-side pin, is called monotonic if it only intersects any straight line running parallel with the ordered pin sequence at most once. This definition is similar to the condition for monotonicity introduced in [7]. Fig. 6 shows eight routing scenarios of 2-layer package routing and PCB escape routing, only (a) and (e) are not monotonically assigned. [8] [10] have shown that the routing on the package layer-1 are monotonic when vias are monotonicly assigned. Following the same idea, when balls are designated to be monotonic, the PCB escape routing is monotonic. In Fig. 6, three nets (*n1*, *n2*, and *n3*) are assigned in eight different patterns. DOPS connects via on the first layer of package and POPS connects balls on PCB. DOPS is given as 1, 2, 3 and POPS is given as 2, 1, 3, in which the order of *n1* and *n2* are reverse. They are called *intersected nets* since their flylines intersects with each other.

Based on these scenarios, we can define the general rule of thumb for designating package pin-out and completing the monotonic global routing. Take Fig. 6(a) as an example, *p1* (pin/net 1; with via *v1* and ball *b1*) is on the left of *p2*, this order is consistent with DOPS but inconsistent with POPS. When the designated column number of n1 and n2 is not in the same order as in DOPS or POPS, that will cause the routing intersection during package layer-1 routing or PCB escape routing. To route without intersection, as shown in Fig. 6, different pin-out assignments will produce different routing results. These results are summarized as follows: ($row_i$ means the row number of $p_i$)

• ***Case 1*** (*row1 = row2, column1 ≠ column2*):

In this case, *p1* and *p2* are located at the same row. To solve the routing intersection, the package layer-1 routing or PCB escape routing must be non-monotonic (see Fig. 6(a) and (e)).

• ***Case 2*** (*row1 ≠ row2, column1 ≠ column2*):

In this case, the assignment of *p1* and *p2* can produce the monotonic routing. However, these pins will possibly be routed through more than one routing track. (see Fig. 6(b), (d), (f) and (h)).

• ***Case 3*** (*row1 ≠ row2, column1 = column2*):

In this case, *p1* and *p2* are located at the same column. For both the package layer-1 routing and PCB escape routing, the routing results not only are monotonic but also use only one routing track. (see Fig. 6(c) and (g)).

According to these scenarios, the pin-out designation rules are defined below:

- ***Rule 1: To achieve monotonic routing***
  1) *the pins corresponding to intersected nets must not be assigned at the same row, or*
  2) *the designated column number of pins corresponding to non-intersected nets must be in the same order as in both* DOPS *and* POPS.
- ***Rule 2: To minimize the routing space***
  1) *the pins corresponding to intersected nets must be assigned at the same column, and*
  2) *the designated row number of these pins corresponding to intersected nets must be adjacent.*

In order to designate package pin-out efficiently and to achieve the monotonic global routing for package design and PCB escape routing, the proposed methodology is to find the intersected relationship between nets by using an intersection graph. The pin-out assignment based on the aforementioned rules of thumb can be satisfied by applying the proposed intersecting relationship analysis.

*B. Pin-Out Designation Methods for Wire Planning*

In [9], they proposed a method using inversion table to analyze the orderings of two sequences. Among all pins of each side, the intersecting relationship must be figured out to know the topology and to get the intersection graph. We use interval diagram, which analyzes the intersection relationship of nets, to generate intersection graph. The interval diagram shown in Fig. 7 shows the intervals of nets. For each net $n_i$, its corresponding interval $I_i$ is composed of a start point $s_i$ and a destination point $d_i$. *si* and *di* are represented by a small solid circle and an arrow respectively and they are determined by the index of $n_i$ in DOPS and POPS respectively.

Interval-Scan Algorithm, which transforms the interval diagram into the intersection graph, is described below. Intersection graph is defined as $G_I = (V,E_I)$ and plotted in Fig. 8. V = $\{vt_i \mid vt_i$ represents interval $I_i\}$. Two vertices are connected by an edge if and only if their corresponding nets intersect each other.

**Interval-Scan Algorithm**:

1) $i \leftarrow 1, j \leftarrow 2$
2) **Repeat:**
3) select net $n_i$ from DOPS and scan $I_i$
4) **Repeat:**
5) select net $n_j$ from DOPS and scan $I_j$
6) **IF** ($I_i$ and $I_j$ go same direction and $I_i$ partially overlaps $I_j$ )
7) **Then** build an edge between $vt_i$ and $vt_j$
8) **IF** ($I_i$ and $I_j$ go opposite direction and $I_i$ fully overlaps $I_j$ )
9) **Then** build an edge between $vt_i$ and $vt_j$
10) increment *j*

11) **Until** Ii has scanned every $I_j$
12) increment $i$
13) **Until** all nets are selected

We have the following lemma:

*Lemma 1:*
If there exists an edge between two vertices, then the child should be placed at the parent's row + 1.

Once we have the intersection graph, the initial pin-out designation can be produced by using a simple algorithm based on the pin-out designation rules. The detailed processes are shown below:

**Initial Pin-Out Designation Algorithm**:
1) $i \leftarrow 1, j \leftarrow 2$
2) select net $n_i$ in DOPS, assign $row_i = 1$, $column_i = 1$
3) **Repeat:**
4) select net $n_j$ in DOPS
5) **IF** an edge exists between $vt_j$ and $vt_i$, **Then**
6) assign $row_j$ based on ***Rule 1***
7) assign $column_j$ based on ***Rule 2***
8) **ELSE**
9) assign $row_j = row_i$
10) assign $column_j$ based on ***Rule 1***
11) $i \leftarrow j$
12) increment $j$
13) **Until** all pins in DOPS have been assigned

Fig. 5 shows an example of initial assignment. By using this designation algorithm, we can obtain the monotonic global routing for package design and PCB escape routing. In the next section, we propose the pin-out optimization methods considering the ways to minimize the package size, routing congestion and wirelength difference on each routing layer, which are critical concerns in chip-package-board codesign.

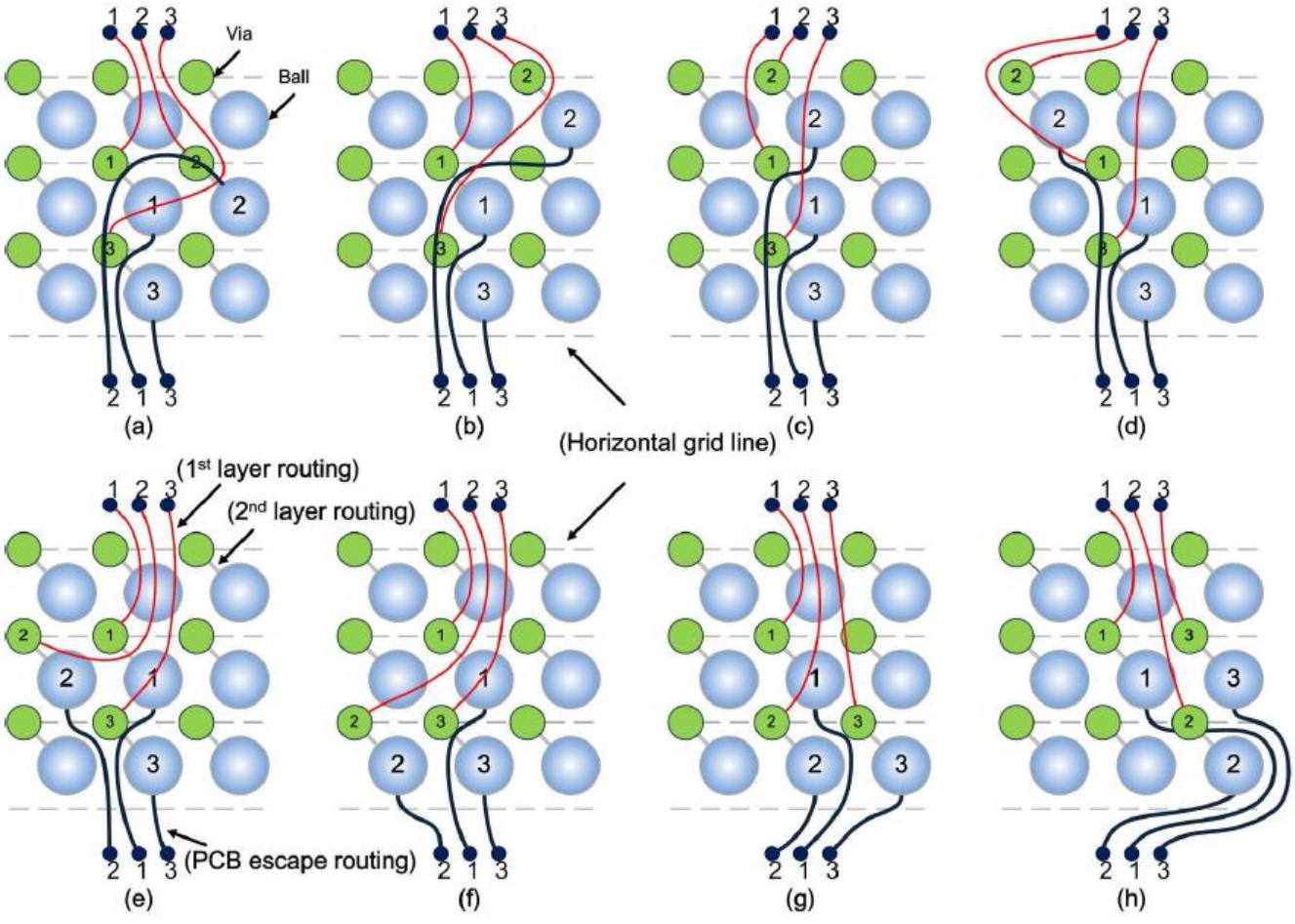


Fig. 6. Routing scenarios which are produced by the pins corresponding to intersected nets designated with different ways.

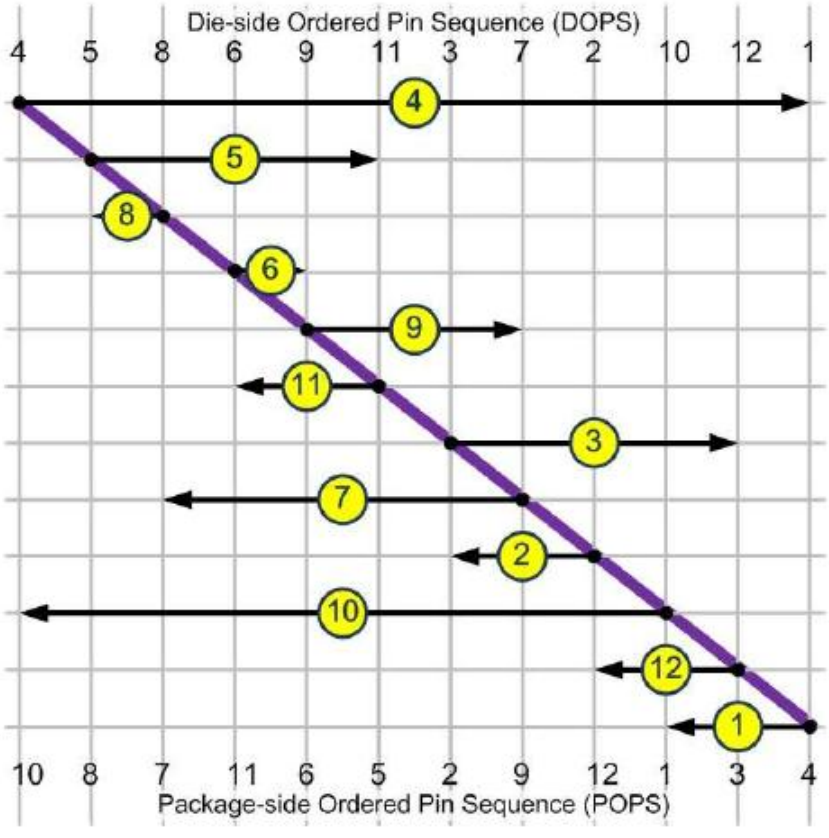


Fig. 7. Interval diagram shows intervals of nets. The start and end points of arrows represent pin locations in die and package sides.

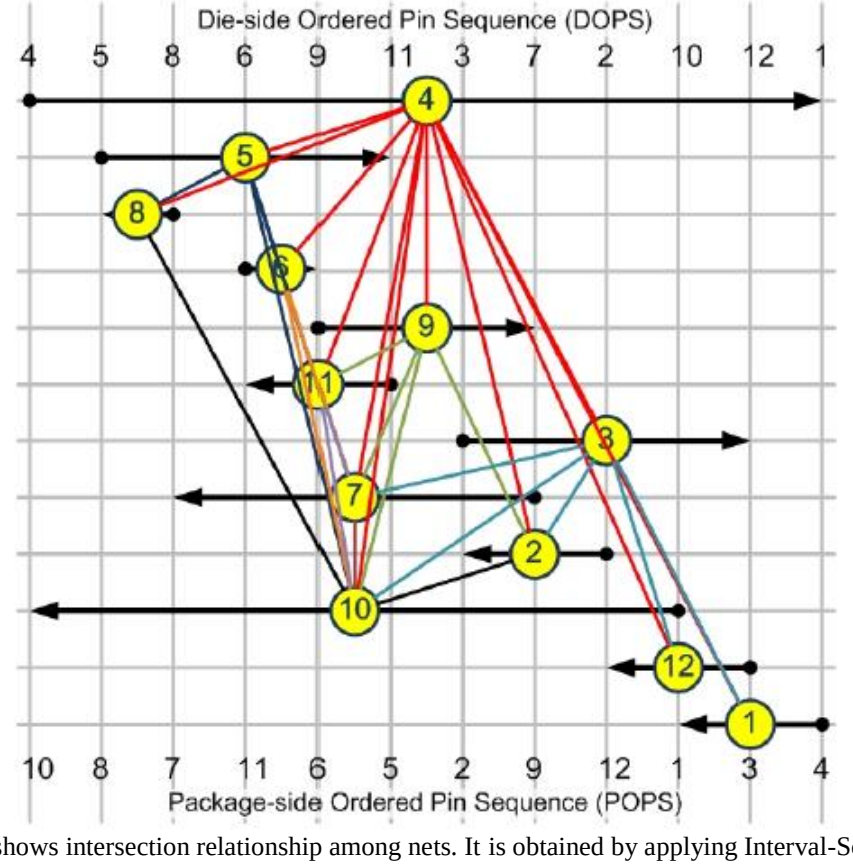


Fig. 8. Intersection graph shows intersection relationship among nets. It is obtained by applying Interval-Scan Algorithm on interval diagram. In intersection graph, if two intervals (an interval in interval diagram is a node in intersection graph) intersect with each other, an edge exists.

## IV. PIN-OUT OPTIMIZATION

### *A. Optimization for Individual Objectives*

Optimization scheme targeting at three individual objectives is discussed in this section: package size (*PS*), wire congestion (*Cong*), and sum of length difference on each routing layer (*Diff*).

*Cong* minimization is achieved by two intuitive methods. One is to equally distribute routing channels: when pins are constrained to be at the same row, intuitively, averaged routing channel distribution minimizes wire congestion. For example, when there are 4 columns and 2 pins, it is best to assign *p1* and *p2* to *column2* and *column4* respectively. The other is to change the column number of pins. As shown in Fig. 9, moving p3 out of the original row relieves *Cong* a lot. It is obvious that focusing on *Cong* minimization possibly enlarges pin-block size; however, the proposed general optimization scheme can still find the assignments which decrease *Cong* while preserving *PS*.

*Diff* minimization is to minimize the variation in wirelength for each layer. Length differences for each layer should be considered separately because the electrical characteristics on package are different from that on PCB. Note that the sum of routes is longer on the package layer-1 because plating leads are always required in packaging process. Rather than minimizing all wires altogether, to minimize the longest wires is sufficient because they often dominate *Diff* term.

*PS* minimization can be obtained if the pins are moved in a certain order iteratively. The order is generated by post-order traversal of intersection graph. For instance, it is 8, 10, 7, 11, 6, 5, 2, 9, 12, 1, 3, 4, for the tree in Fig. 8 and Fig. 10(a). The pins move sequentially in this order and obey the direction priority (go left > go bottom-left > go down). Fig. 10 (b)-(f) illustrates each step of the procedure in minimizing PS. For each step, only one pin moves once at a time as long as there exists an empty via/ball slot, and all pins move in order. Each pin stops moving if it touches the boundary thus the number of rows cannot be increased. In the initial solution, the number of rows is minimum, which is determined by the depth of intersection tree shown in Fig. 10(a). In order to preserve monotonic assignment, pins which cross each other cannot be placed in the same row. Thus, to minimize package size is to try to minimize the number of columns.

### *B. Unified Cost Optimization*

The proposed optimization scheme is to: 1)select one pin/via/ball which costs most, 2)search its legal neighbors, 3)perform operations between the pin/via/ball and its legal neighbors. It is important to honor the legality in order to keep the assignment monotonic. The costs of via grid array (cVGA) and ball grid array (cBGA) are summed up. So an optimization step which merits cVGA may demerit cBGA. We use the following heuristics to find better solutions in moving pin/via/ball. Note that *Cong*, *Diff*, and *PS* are normalized to fairly compare with each other.

- Greedy Method: Starting from initial solution, only downhill searches are accepted. The method keeps moving the most expensive pin/via/ball to its less expensive neighbors. It is useful when there is one pin/via/ball which contributes a lot in cost. However, it is not suitable when there is a group of pins/vias/balls which should be optimized simultaneously.

- Lowest partial cost (LPC) Method: A serial of moves are firstly accepted to escape local optimal. Moves are performed whose accumulated sum of costs is minimum. The first move is relatively important because the quality of all following moves depends on it.

The optimization scheme is conducted in two stages: tie-up optimization followed by loose optimization.

Each pair of $v_i$ and $b_i$ are tie-up as $p_i$ to search for a global optima in the first stage, and they are loosened to search for local optima in the second stage. Fig. 11 shows an optimization step for *n3*. *p3* attempts to decrease cost by exploring its neighbors in (a), while *v3* and *b3* search to decrease their own cost separately.

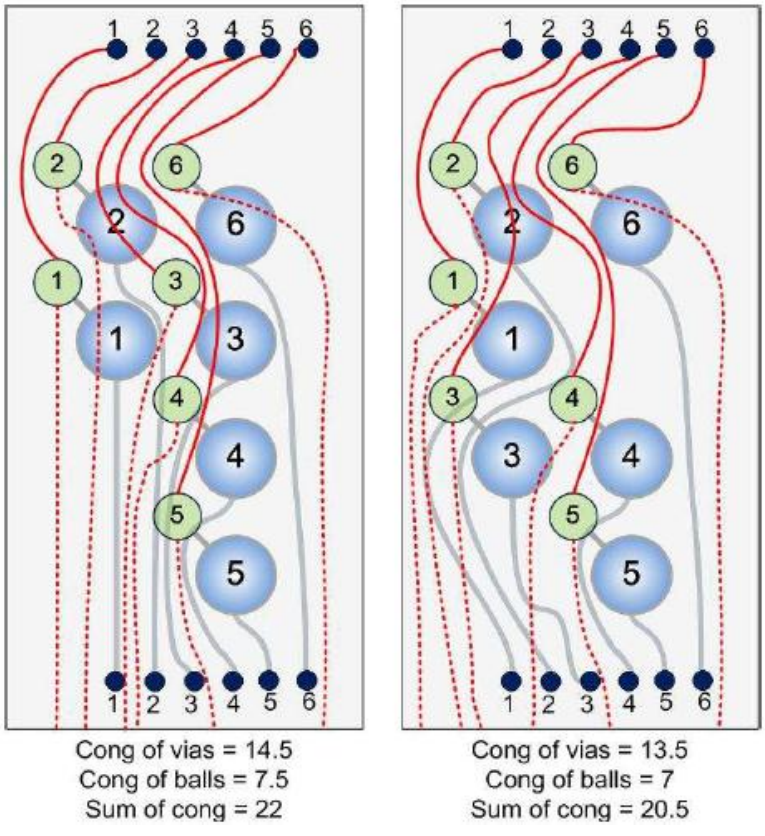


Fig. 9. Optimization for wire congestion. Originally, the cong cost is 22 for the assignment on the left. Moving via3 and ball3, like the assignment on the right, can reduce cong cost to 20.5.

*C. Cost Evaluation*

In the unified cost optimization, the cost function is defined as follows:

$$Cost_{vi/bi} = \alpha \times Cong_{vi/bi} + \beta \times Diff_{vi/bi} + \gamma \times PS_{vi/bi} \quad (1)$$

where $Cost_{vi/bi}$ indicates the cost of a via $v_i$ or a ball $b_i$, and $\alpha$, $\beta$, and $\gamma$ are user-defined parameters. Each via/ball has a cost composed of *Cong*, *Diff*, and *PS*. And total cost is the sum of all vias/balls. Here we define the cost of three objectives separately.

The cost of wire congestion of two via/balls is defined as:

$$Cong_{vi/bi} = \frac{\#wire_l}{\#channel_l} + \frac{\#wire_r}{\#channel_r} \quad (2)$$

where $\#wire_{l/r}$ denotes the number of wires which lie on the left/right, $\#channel_{l/r}$ denotes the number of routing channels on the left/right, shown in Fig. 12. Note that plating leads are metal wires so they also contribute to *Cong*.

The cost of length difference is defined as:

$$Diff_{vi/bi} = dist(v_i/b_i) - dist(avg) \qquad (3)$$

where *dist*($v_i/b_i$) denotes the Manhattan distance of the via/ball and *dist(avg)* denotes the averaged Manhattan distance of all vias/balls. Since the monotonic assignment can guarantee monotonic routing, using Manhattan distance to estimate wirelength is sufficient.

The cost of PS is defined as:

$$PS_{vi/bi} = \begin{cases} \frac{1}{\#v/b_V} + \frac{1}{\#v/b_H} \times W \times L & \text{if } \#v/b_{V/H} > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

where $\#v/b_V$ and $\#v/b_H$ denote the number of vias/balls which lie on the vertical and horizontal boundary respectively, W and L denote number of columns and rows of package size respectively, shown in Fig. 12.

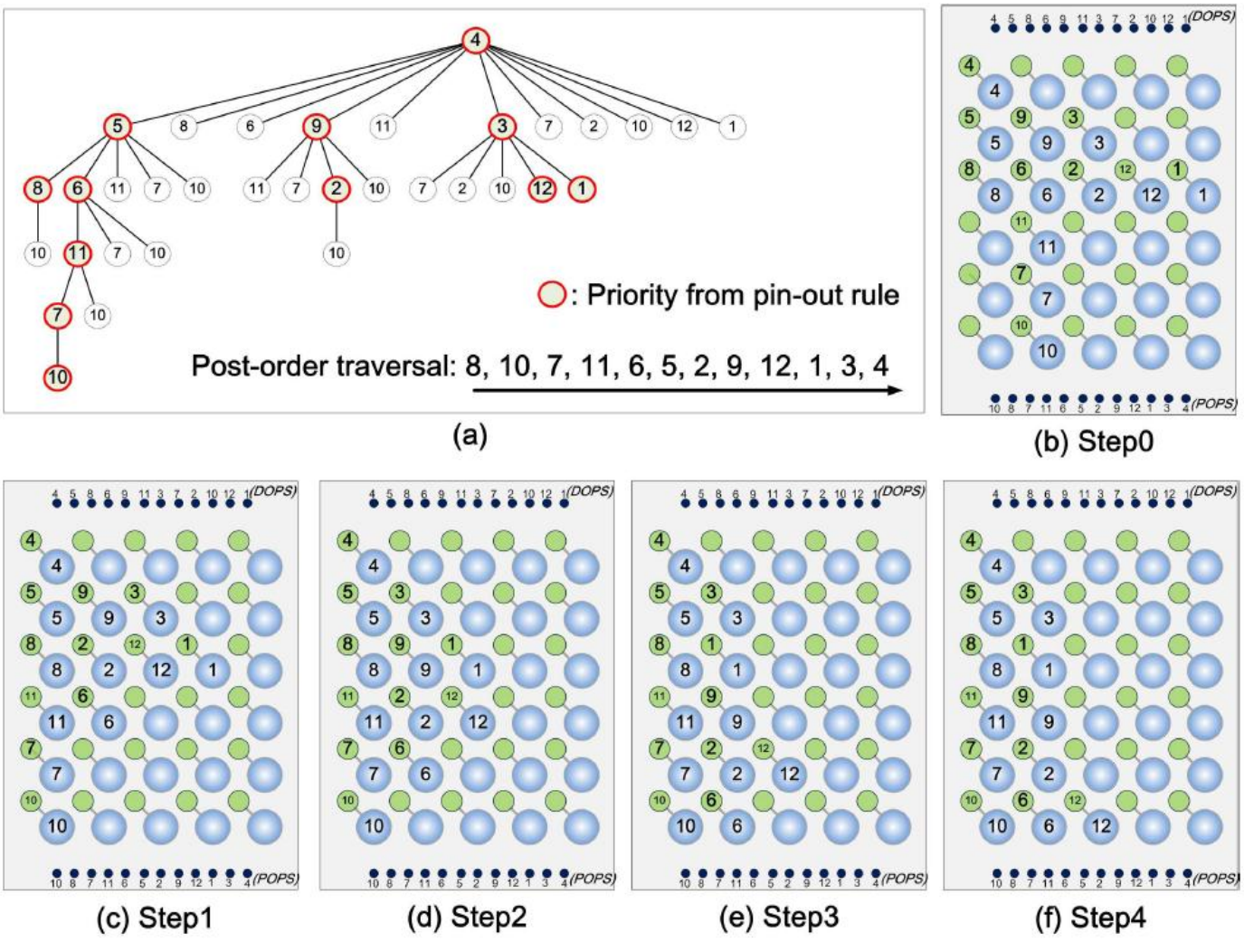


Fig. 10. Optimization for package size. (b)-(f) show the status of movement. The number of columns used is decreased from 5 to 3.

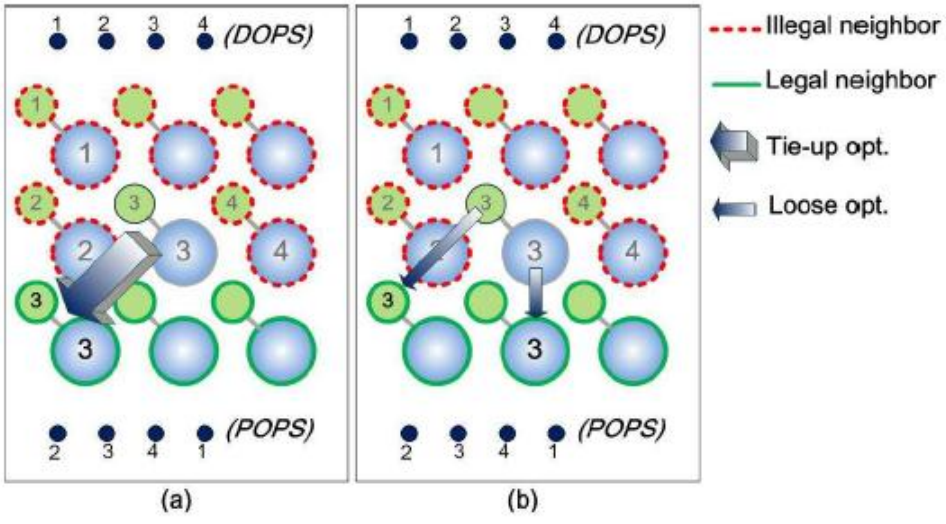


Fig. 11. Post Optimization. We first tie up v3 and b3 for global search, then we loosen this constraint so that they can separately find other local solutions to reduce the cost.

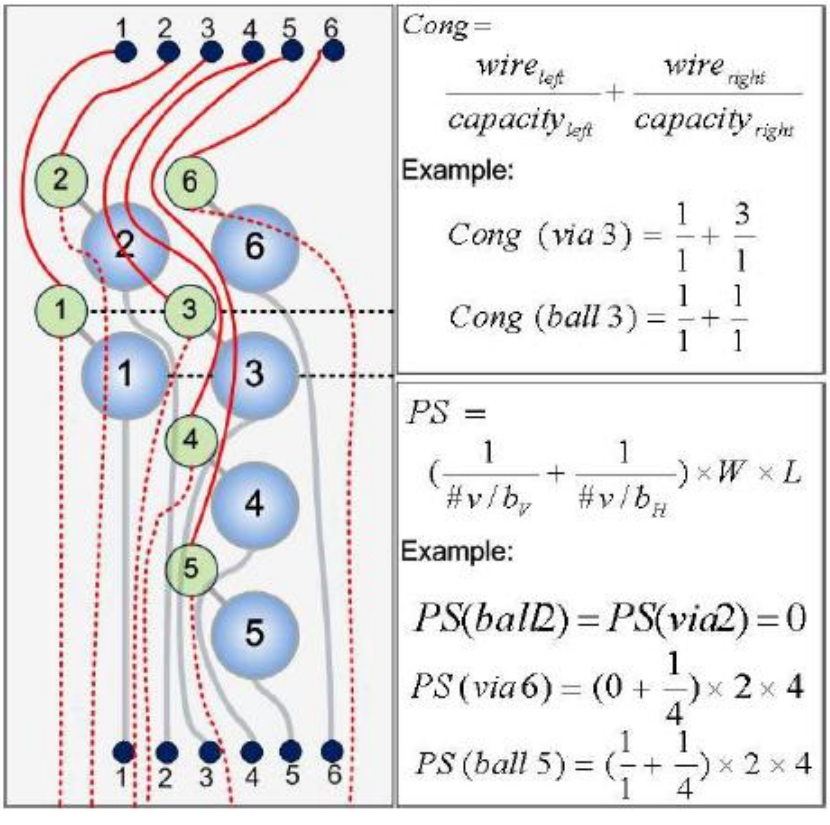


Fig. 12. Cost evaluation (*Cong* and *PS*) for via and ball. For Cong(via3), there is one wire (dotted red) on the left of via3; 3 wires (2 solid red and 1 dotted red) on the right of via3, so Cong(via3) is calculated as 1/1+3/1. For PS(via6), column2 is the right-most column which contributes to package size, there are 4 vias on column2, so PS(via6) is (0+1/4)*2*4.

## V. EXPERIMENTAL RESULTS

Our algorithm is implemented using C++ on a 3.0GHz Intel Xeon Quad Core Processor 5160 PC under the Linux operation system. In the following tables, cV GA denotes the total cost of Via Grid Array, cBGA denotes the total cost of Ball Grid Array, and *Sum* is the sum of cV GA and cBGA.

$$cVGA = \sum_{i=0}^{n} Cost_{vi} \quad (5)$$

$$cBGA = \sum_{i=0}^{n} Cost_{bi} \quad (6)$$

$$Sum = cVGA + cBGA \quad (7)$$

Two optimization schemes, *Greedy* and *LPC*, are implemented and tested in two modes: *tie-up* and *full*. tie-up indicates that, for *ni*, *vi* and *bi* are tied-up as *pi* to optimize simultaneously. *full* indicates that, after having conducted *tie-up*, *pi* is loosened to perform optimization for vias and balls separately. The proposed initial solution is generated by Initial Pin-Out Designation Algorithm proposed in Section III. Recall equation (1), the cost of *vi/bi* is composed of $PS_{vi/bi}$, $Diff_{vi/bi}$, and $Cong_{vi/bi.}$ In the experiments, the coefficients, $\alpha$, $\beta$, and $\gamma$, are set normalizing to the initial solution and defined as:

$$\alpha \times \sum_{i=0}^{n} Cong_{vi/bi} = 1 \quad (8)$$

$$\beta \times \sum_{i=0}^{n} Diff_{vi/bi} = 1 \quad (9)$$

$$\gamma \times \sum_{i=0}^{n} PS_{vi/bi} = 1 \quad (10)$$

Therefore, $cV\,GA_{initial\_sol} = cBGA_{initial\_sol} = 3.00$ and $Sum_{initial\_sol} = 6.00$.

In order to show our effectiveness in package wire planning, we compare with [10] in congestion perspective. Both [10] and the proposed methodology adopt the same 2-layer package model. However, [10] assigns its initial solution randomly and performs optimization for cV GA only. Their cost evaluation considers total wire-lengh minimization while our work focuses on length-variation minimization. Besides, their package size is given as input while ours can decide the trade-off between package size and routability.

In Table I, we test the proposed methodology in 4 industrial cases in which the largest pin count is 40, the results of bench1 for [10] are not available. Note that most costs of [10] are greater than 1.00, which means that they are more expensive than our initial solution. The proposed initial solution improves 54% in average, compared to [10]. The initial solutions can be further improved by 22% in average after applying *Greedy* method in *full* mode.

**TABLE I**

COMPARE [10] AND OUR POST-OPTIMIZATION SCHEME WITH OUR PROPOSED INITIAL SOLUTION. IN OUR INITIAL SOLUTIONS, CONG, DIFF, AND PS ARE ALL EVALUATED AS 1.00. THE INITIAL SOLUTIONS OUTPERFORM [10] BECAUSE MOST COSTS OF [10] ARE GREATER THAN 1.00. GREEDY METHOD CAN FURTHER IMPROVE THE INITIAL SOLUTIONS BY 22% IN AVERAGE. THE RESULTS OF BENCH1 FOR [10] ARE NOT AVAILABLE

| | [10] (opt. for cVGA only) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | VGA | | | BGA | | | Imp.% | | | |
| | Cong | Diff | PS | Cong | Diff | PS | Cong | Diff | PS | Sum |
| bench2 | 2.09 | 1.41 | 0.93 | 1.61 | 1.34 | 0.47 | -85% | -38% | 30% | -31% |
| bench3 | 1.70 | 4.21 | 1.30 | 1.50 | 1.12 | 1.30 | -60% | -166% | -30% | -85% |
| bench4 | 1.59 | 4.03 | 0.47 | 1.21 | 1.60 | 0.47 | -40% | -182% | -53% | -56% |
| bench5 | 1.09 | 3.63 | 1.08 | 0.98 | 1.21 | 1.62 | -4% | -142% | -35% | -42% |
| avg. | 1.62 | 3.32 | 0.94 | 1.33 | 1.32 | 0.96 | -47% | -132% | 5% | -54% |
| | Greedy Method full mode | | | | | | | | | |
| | VGA | | | BGA | | | Imp.% | | | |
| | Cong | Diff | PS | Cong | Diff | PS | Cong | Diff | PS | Sum |
| bench2 | 0.85 | 0.14 | 0.95 | 0.77 | 0.23 | 0.95 | 19% | 81% | 5% | 35% |
| bench3 | 0.75 | 0.30 | 1.06 | 0.80 | 0.31 | 1.00 | 22% | 69% | -3% | 30% |
| bench4 | 0.95 | 0.89 | 1.00 | 1.05 | 0.55 | 1.00 | 0% | 28% | 0% | 9% |
| bench5 | 0.87 | 0.84 | 1.00 | 0.94 | 0.58 | 1.00 | 9% | 29% | 0% | 13% |
| avg. | 0.86 | 0.54 | 1.00 | 0.89 | 0.42 | 0.99 | 13% | 52% | 1% | 22% |

Table II shows the results of two optimization schemes *Greedy* and *LPC*. Similar behaviors are observed for most of the results. They improve the initial solutions by 16% in tie-up mode. This can be further optimized in full mode to give a final improvement of 20-22% in average. Note that the initial assignment of bench1 is shown previously in Fig. 5. The execution time for all experiments is less than 1 second.

TABLE II

THE RESULTS IN DIFFERENT METHODS AND MODES, COMPARED WITH THE INITIAL SOLUTIONS. TAKE BENCH1 AS AN EXAMPLE, THE RESULTS OF GREEDY METHOD IN TIE-UP MODE IMPROVES Diff AND PS BY 53% AND 25% RESPECTIVELY, BUT IT WORSENS Cong BY 27%. OUR POST OPTIMIZATION SCHEMES SHOW THAT THE IMPROVEMENTS COMPARED WITH THE PROPOSED INITIAL SOLUTION ARE AVERAGELY GREATER THAN 20%.

| | Greedy Method | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | tie-up mode | | | | full mode | | | |
| | Cong | Diff | PS | Sum | Cong | Diff | PS | Sum |
| bench1 | -27% | 53% | 25% | 17% | -12% | 49% | 25% | 21% |
| bench2 | 7% | 73% | 5% | 30% | 19% | 81% | 5% | 35% |
| bench3 | 13% | 43% | 0% | 19% | 22% | 69% | -3% | 30% |
| bench4 | -1% | 11% | 0% | 3% | 0% | 28% | 0% | 9% |
| bench5 | 13% | 14% | 0% | 9% | 9% | 29% | 0% | 13% |
| avg. | 1% | 39% | 6% | 16% | 8% | 51% | 5% | 22% |
| | LPC Method | | | | | | | |
| | tie-up mode | | | | full mode | | | |
| | Cong | Diff | PS | Sum | Cong | Diff | PS | Sum |
| bench1 | -27% | 53% | 25% | 17% | -12% | 49% | 25% | 21% |
| bench2 | 12% | 73% | 5% | 30% | 18% | 78% | 5% | 33% |
| bench3 | 13% | 43% | 0% | 19% | 18% | 58% | -3% | 24% |
| bench4 | -1% | 11% | 0% | 3% | -1% | 26% | 0% | 8% |
| bench5 | 12% | 15% | 0% | 9% | 8% | 29% | 0% | 12% |
| avg. | 2% | 39% | 6% | 16% | 6% | 48% | 5% | 20% |

The results of bench3 are plotted in Fig. 13. Fig. 13(a) shows the proposed initial assignment in which all wires are planned monotonically; (b) and (c) shows the results of post-optimization using *Greedy* and *LPC* methods respectively. They have similar patterns with slightly different assignment. The cost of (b) is lower than (c) by 6% because greedy method can find better solution in loose mode. This shows that the cost of BGA benefits more than that of VGA from loose optimization. Fig. 13(d) is one of the experimental results in [10], which shows smaller in package size, however it is more congested and has larger variation in wirelength.

## VI. CONCLUSION

**In order to address the long-existing problem in slow turnaround between design, package and system houses, we define a new subproblem in helping the fast estimation of wire planning in chip-package-board codesign. Core designers can specify the preferred I/O pad ordering; system designers can specify the preferred bump pin-out designation. We can efficiently analyze if the preferences from both sides accommodate each other before performing RDL routing and substrate routing. We also consider the essential concerns in package design, such as routing congestion, package size and length deviation among all nets. The results show the effectiveness and efficiency.**

## Reference:


[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.-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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 8, pp. 1417-1429, 2007.

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

[4] M. M. Ozdal and D.-F. Wong, “Algorithms for Simultaneous Escape Routing and Layer Assignment of Dense PCBs,” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 8, pp. 1510-1522, 2006.

[5] H. Kong, T. Yan and D.-F. Wong, “Automatic Bus Planner for Dense PCBs,” in *Proceedings ACM/IEEE Design Automation Conference*, pp. 326-331, 2009.

[6] L. Luo and D.-F. Wong, “Ordered Escape Routing Based on Boolean Satisfiability,” in *Proceedings Asia and South Pacific Design Automation Conference*, pp. 244-249, 2008.

[7] Y. Tomioka and A. Takahashi, “Monotonic parallel and orthogonal routing for single-layer ball grid array packages,” in *Proceedings Asia and South Pacific Design Automation Conference*, pp., 24-27, 2006.

[8] Y. Kubo and A. Takahashi, “A Global Routing Method for 2-Layer Ball Grid Array Packages,” in *Proceedings International Symposium on Physical Design*, pp. 36-43, 2005.

[9] S.-S. Chen, J.-J. Chen, T.-Y. Lee, C.-C. Tsai, and S.-J. Chen, “A New Approach to the Ball Grid Array Package Routing,” in *Trans. on IEICE*, vol.E82-A, no.11, pp.2599-2608, 1999.

[10] Y. Tomioka and A. Takahashi, “Routability Driven Modification Method of Monotonic Via Assignment for 2-Layer Ball Grid Array Packages,” in *Proceedings Asia and South Pacific Design Automation Conference*, pp. 238-243, 2008.

[11] R.-J. Lee and H.-M. Chen, "Fast Flip-Chip Pin-Out Designation Respin for Package-Board Codesign," in IEEE Transactions on Very Large Scale Integration Systems (EI/SCI), vol. 17, no. 8, August 2009 (TVLSI-Aug-09)

[12] R.-J. Lee and H.-M. Chen, "Efficient Package Pin-Out Planning with System Interconnects Optimization for Package-Board Codesign," in IEEE Transactions on Very Large Scale Integration Systems (EI/SCI), vol. 19, no. 5, May 2011 (TVLSI-May-11)
[13] C.-H. Lu, H.-M. Chen, C.-N. Liu, and W.-Y. Shih, "Package Routability- and IR-Drop-Aware Finger/Pad Assignment in Chip-Package Co-Design," Proc. of IEEE Design, Automation and Test in Europe, pp. 845-850, April 2009 (DATE-09)
[14] R.-J. Lee and H.-M. Chen, "Novel I/O-Bump Design and Optimization for Chip-Package Codesign," Proc. of IEEE Electrical Design of Advanced Package & Systems Symposium, December 2009 (EDAPS-09)
[15] R.-J. Lee and H.-M. Chen, "Efficient Package Pin-Out Planning with Chip-Package Interconnects Optimization," Proc. of IEEE Electrical Design of Advanced Package & Systems Symposium, December 2009 (EDAPS-09)
[16] C.-Y. Lin, Y.-W. Chen, W.-J. Chen, and H.-M. Chen, "Fast Detection and Analysis Schemes for System-in-Package in the Presence of RAM," IEEE Workshop on RTL and High Level Testing, December 2010 (WRTLT-10)
[17] K.-S. Lin, H.-W. Hsu, R.-J. Lee, and H.-M. Chen, "Area-I/O RDL Routing for Chip-Package Codesign Considering Regional Assignment," Proc. of IEEE Electrical Design of Advanced Package & Systems Symposium, December 2010 (EDAPS-10)
[18] R.-J. Lee and H.-M. Chen, "Row-Based Area-Array I/O Design Planning in Concurrent Chip-Package Design Flow," Proc. of IEEE Asia and South Pacific Design Automation Conference, January 2011 (ASP-DAC-11)
[19] T.-Y. Tsai, R.-J. Lee, C.-Y. Chin, C.-Y. Kuan, H.-M. Chen, and Y. Kajitani, "On Routing Fixed Escaped Boundary Pins for High Speed Boards," Proc. of IEEE Design, Automation and Test in Europe, March 2011 (DATE-11)
[20] R.-J. Lee, H.-W. Hsu, and H.-M. Chen, "Implementing the Practical Chip-Package-Board Codesign Considering Package Design and Board Escape Routing," ACM/IEEE Design Automation Conference, Work in Progress, June 2011