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

Ren-Jie Lee, *Hung-Ming Chen

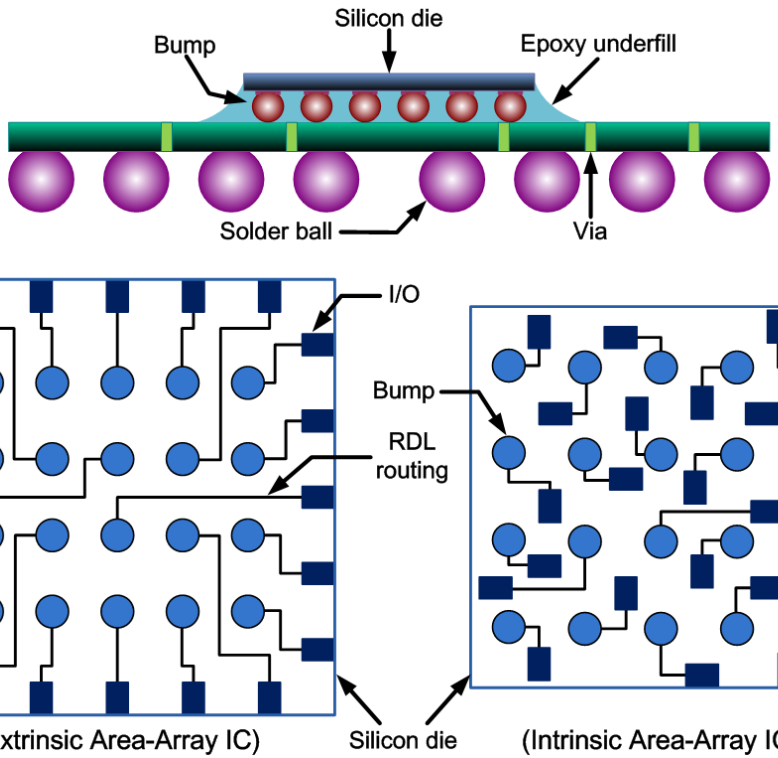
(EE Dept., National Chiao Tung Univ., Taiwan)

Outline

- Introduction
- Novel I/O-bump tile design and I/O-row based planning
- Package-aware I/O-bump planning methods
- Experimental results
- Conclusion

Previous Work

- **Wire-bonding package**
  - Peripheral I/O-pad
- **Flip-chip package**
  - Area-array I/O-bump
    - ✓ Extrinsic area-array I/O
      - network-flow-based [5]
      - ILP-based [6]
    - ✓ Intrinsic area-array I/O
      - I/O clustering method [7]
      - constraint-driven I/O planning [8]



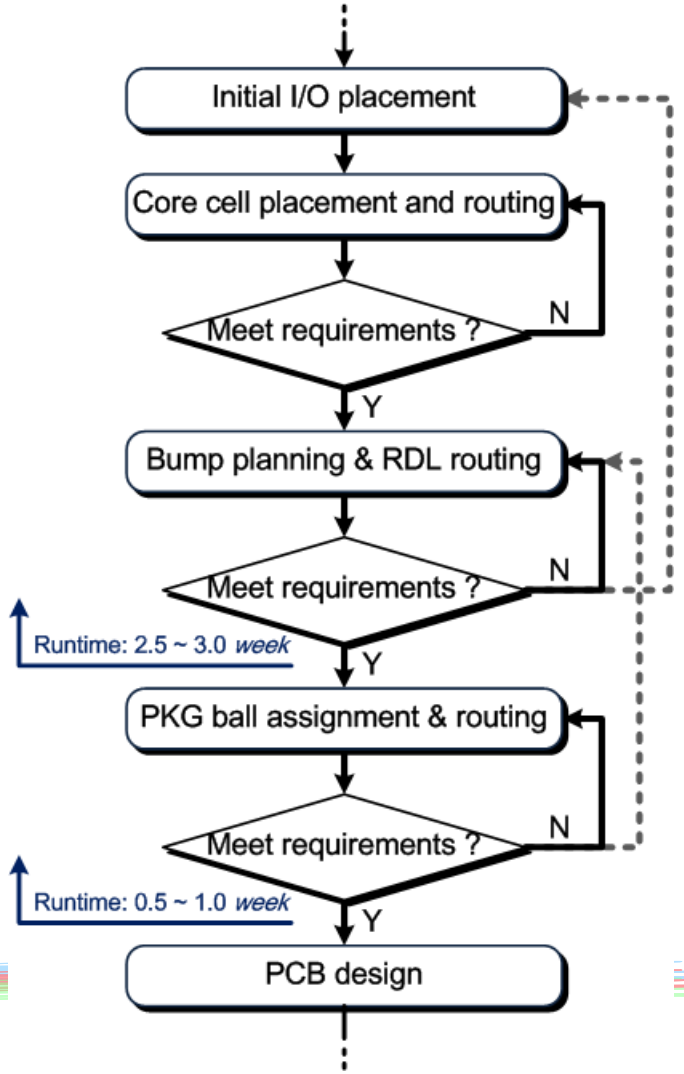
Motivation

Issues in previous works

- **Bumps are assumed to be arranged in fixed array location**
  - ✓ Flexibility in optimizing chip and package designs is restricted
  - ✓ Costly RDL routing or I/O planning is needed
- **Pin-out (ballplan) is ignored**
  - ✓ It possibly leads to complicated or failed package design
- **The conventional design flow is a sequential flow**
  - ✓ It will result in long and costly re-spin cycles on satisfying the entire system's design constraints (see next slide)

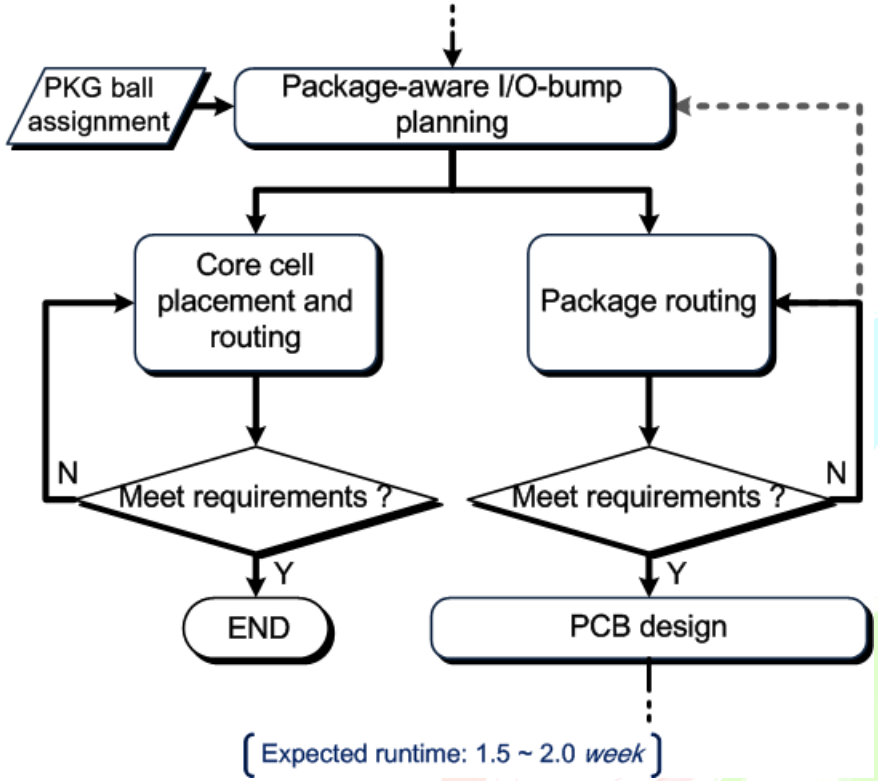
Motivation (cont.)

Conventional chip-package design flow (IC-driven)

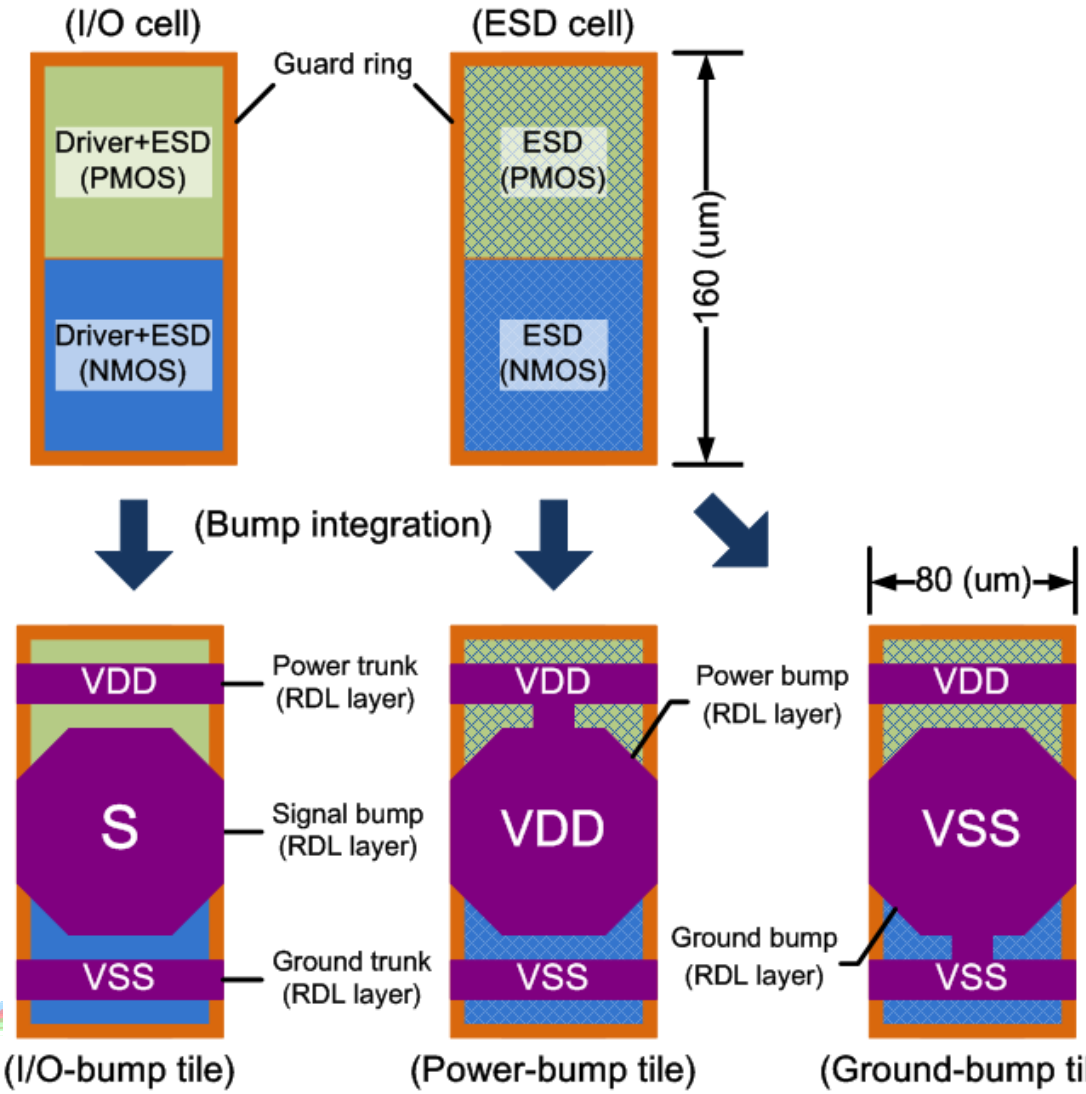


Our Contributions

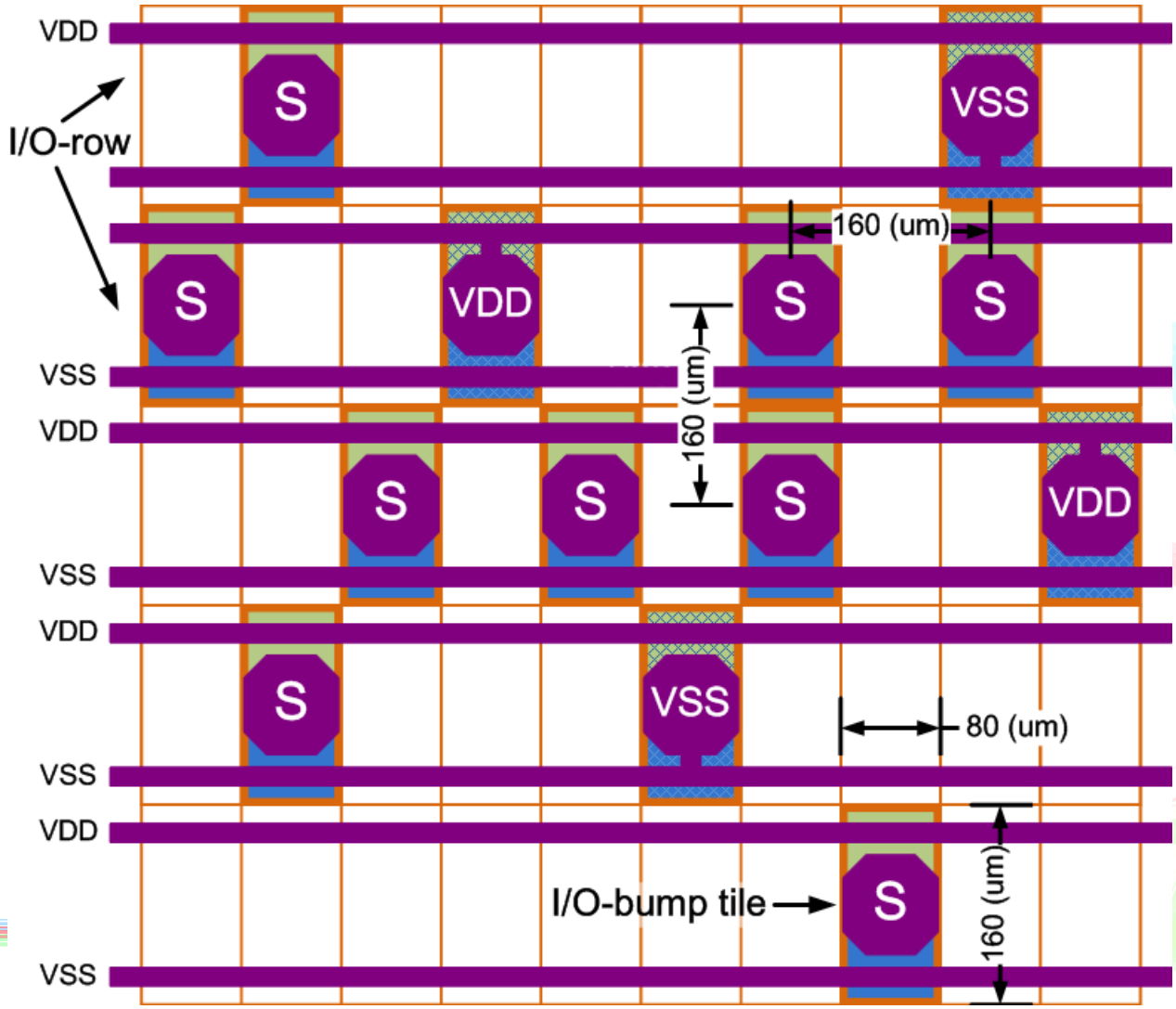
- We propose a concurrent design flow
- We design the specific I/O-bump tiles with I/O-row based scheme
- We develop two heuristics and one optimization algorithm to place I/O-bump



Novel I/O-bump tile design and I/O-row based planning



Novel I/O-bump tile design and I/O-row based planning (cont.)



Problem Description

Input:

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

—The design rules for chip and package.

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:

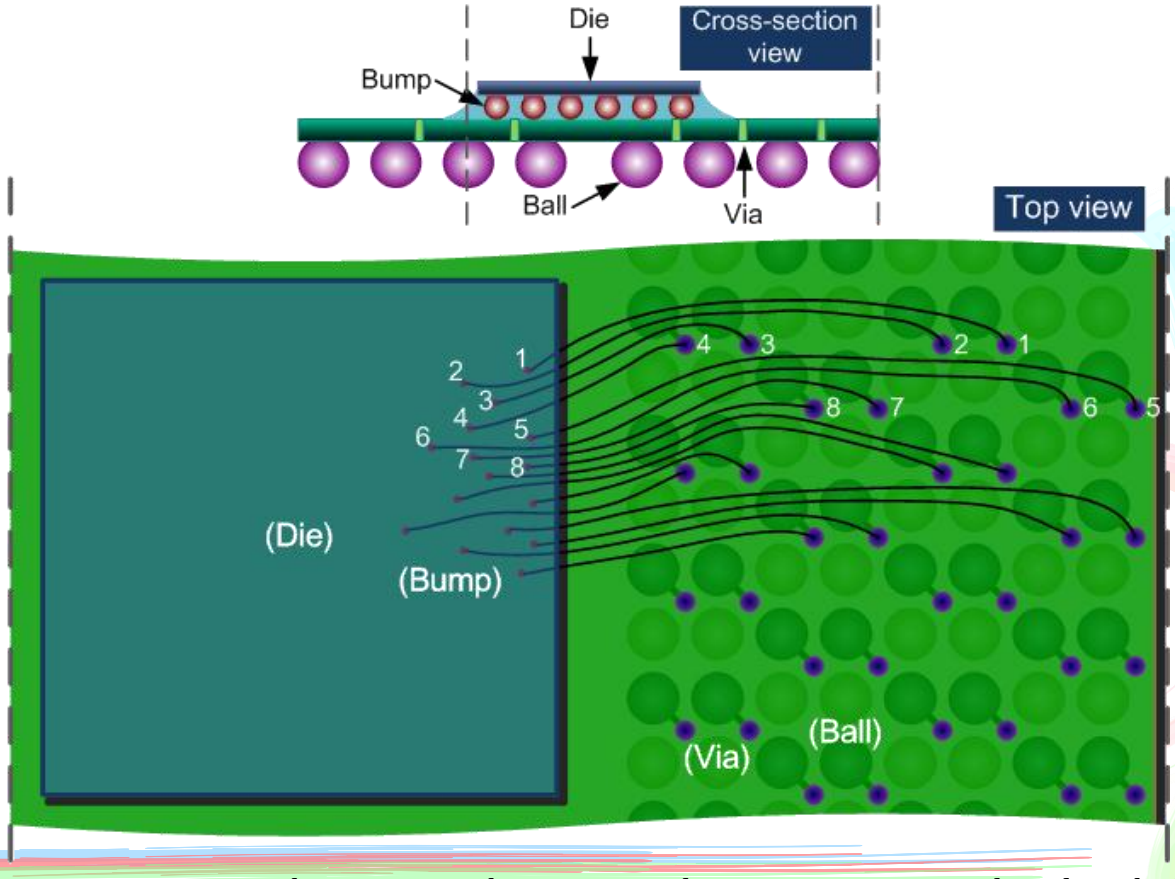
—Minimum possible routing layer (minimum net crossing number).

—Minimum timing delay (minimum total net length).

—Minimum signal skew (minimum sum of length difference/deviation on each net).

Package-aware I/O-bump planning methods

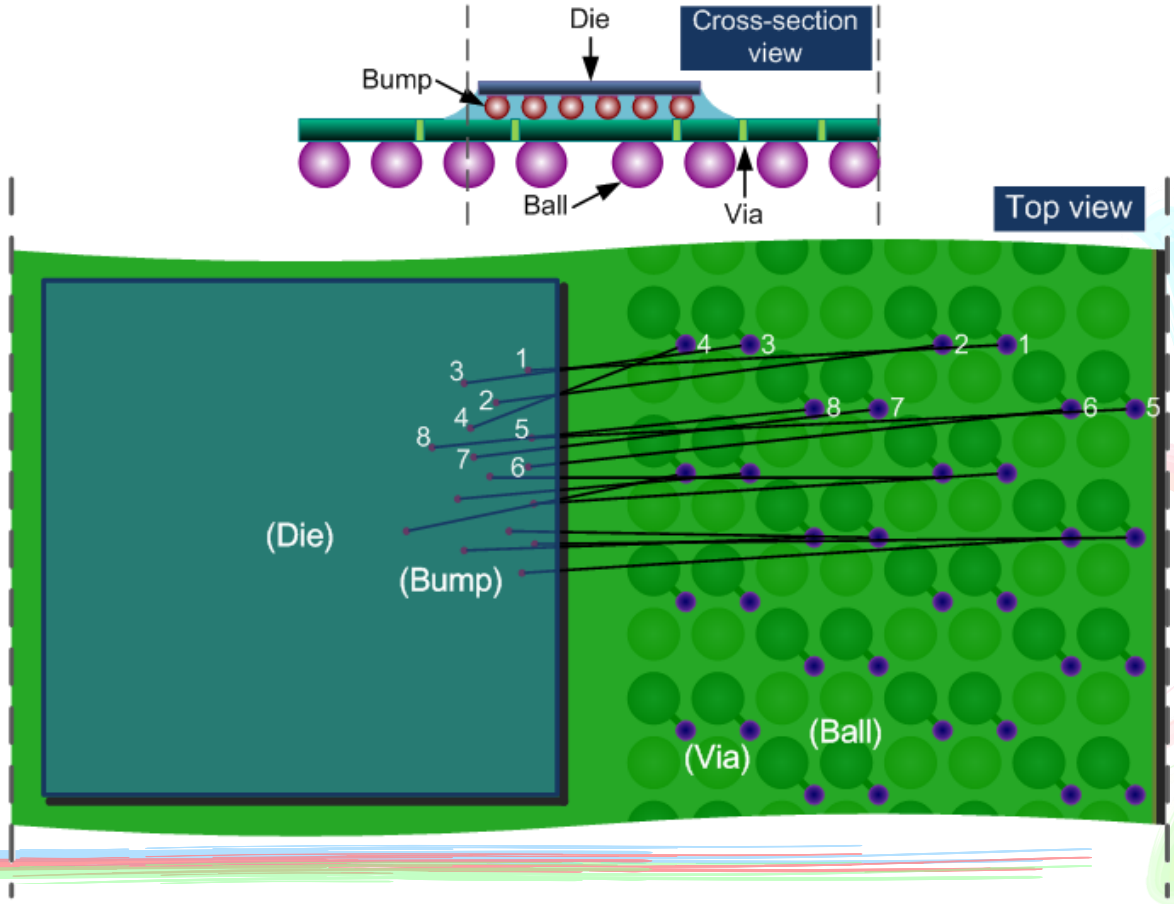
Heuristic *SORT*: Double Sorting for Planar Planning



*The monotonic routing is a route with no U-turn path. It consumes less routing resource and results in higher routing completion compared with nonmonotonic routing [6].

Package-aware I/O-bump planning methods (cont.)

- Heuristic *GREEDY*: Shortening Flylines Between I/O-Bumps and Package Balls



Package-aware I/O-bump planning methods (cont.)

Optimization *WBIPT*: Matching-Based Assignment

Minimize

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

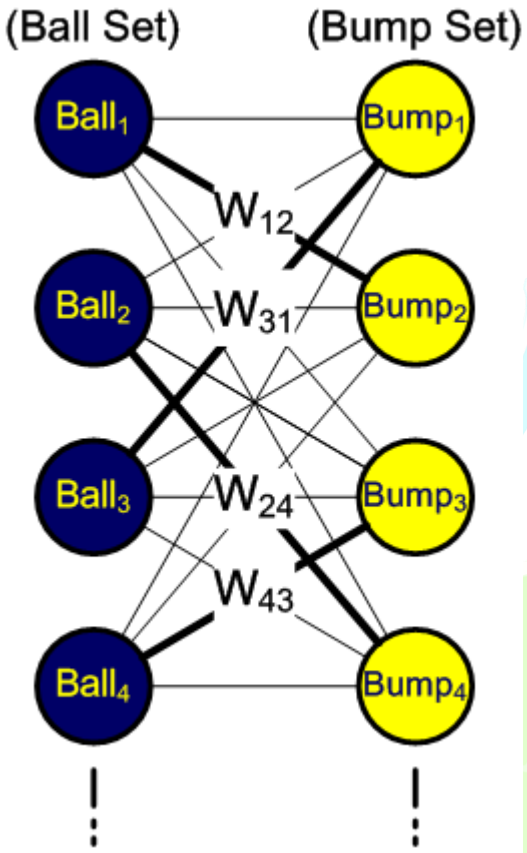
subject to

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

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

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

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



* where $Diff_{ij} = |Order_{ball_i} - Order_{bump_j}|$ is obtained through directly subtracting the order of $Bump_j$ from $Ball_i$, and therefore calculating the upper bound of crossing number [14].

Experimental Results

The industrial chip designs

	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)						

* *d2* and *d3* are core-limited designs.

Experimental Results (cont.)

- **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	$WBIPT$ ($\alpha = 500, \beta = 1.0$)

- **The I/O-bump planning on test case *d5* (random)**

	Flyline criteria					Total runtime (sec)
	Net crossing	Wirelength		Length deviation		
		Total (um)	Increase	Total (um)	Increase	
♯1a	0	5473480	1.010x	1432024	1.989x	< 2.0
♯2a	1056	5416680	—	720120	—	< 2.0
♯3a	0	5473480	1.010x	1432024	1.989x	< 5.5
♯4a	32	5461600	1.008x	1209704	1.680x	< 5.5
♯5a	140	5447080	1.006x	996644	1.384x	< 5.5
♯6a	376	5437040	1.004x	920888	1.279x	< 5.5
	("—" stands for the baseline)					

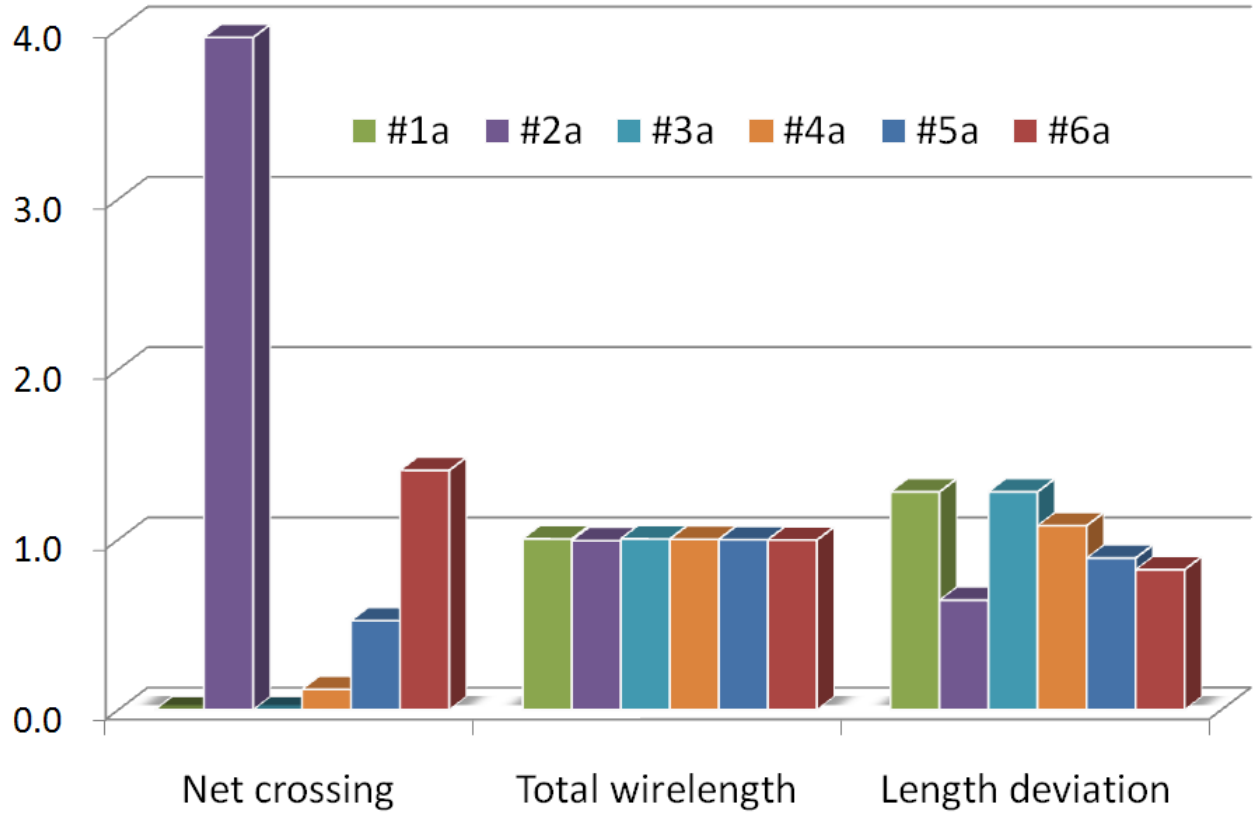
Experimental Results (cont.)

- The I/O-bump planning on test case *d5* (uniform)

	Flyline criteria					Total runtime (sec)
	Net crossing	Wirelength		Length deviation		
		Total (um)	Increase	Total (um)	Increase	
$\sharp 1b$	0	5813320	1.007x	1645728	1.167x	< 2.0
$\sharp 2b$	1432	5775240	–	1410436	–	< 2.0
$\sharp 3b$	0	5813320	1.007x	1645728	1.167x	< 5.5
$\sharp 4b$	24	5802480	1.005x	1505192	1.067x	< 5.5
$\sharp 5b$	32	5794920	1.003x	1480016	1.049x	< 5.5
$\sharp 6b$	148	5786320	1.002x	1374196	0.974x	< 5.5
	("–" stands for the baseline)					

Experimental Results (cont.)

- Results of normalized performance metrics



Conclusion

- We propose a concurrent design flow which completes the core-I/O placement and package routing in parallel.
- With our I/O-bump tile designs and I/O-row based scheme, we improve the flexibility in arranging I/Os and bumps.
- Two heuristics and one optimization algorithm are provided to implement the package-aware I/O-bump planning.

Thank You