Transactions Briefs

Design Migration From Peripheral ASIC Design to Area-I/O Flip-Chip Design by Chip I/O Planning and Legalization

Chia-Yi Chang and Hung-Ming Chen

***Abstract*—Due to higher input/output (I/O) count and power delivery problem in deep submicrometer (DSM) regime, flip-chip technology, especially for area-array architecture, has provided more opportunities for adoption than traditional peripheral bonding design style in high-performance application-specific integrated circuit and microprocessor designs. However, it is hard to tell which technique can provide better design cost edge in usually concerned perspectives. In this paper, we present a methodology to convert a previous peripheral bonding design to an area-I/O flip-chip design. It is based on an I/O buffer modeling and an I/O planning algorithm to legalize I/O buffer blocks with core placement without sacrificing much of the previous optimization in the original core placement. The experimental results have shown that we have achieved better area and I/O wirelength in area-IO flip-chip configuration (especially for pad-limit designs), compared with peripheral bonding configuration in packaging consideration.**

***Index Terms*—Area-array flip-chip, design migration, input/output (I/O) planning and legalization.**

I. Introduction

Nowadays more circuits are integrated onto one single chip [system-on-chip (SoC)], and this means more input/outputs (I/Os) are needed in modern designs. Many high-performance integrated circuits (ICs) and microprocessors are built with more I/O connections than in the past [1], meanwhile [2] showed the trend in the increase in I/O count and the reduction of die size when the typical peripheral wire-bond design was replaced by the flip-chip design. As a result, the flip-chip design is considered a better choice [3], [4]. There are some more advantages of flip-chip design, including minimizing size of electrostatic discharge (ESD) structure for intra-package I/O and improved signal integrity due to power and ground pad structure.

Since flip-chip design allows I/O buffers to be placed anywhere on the die, we need to focus on the change to improve the design and the cost for placing I/O buffer blocks into the design. Many approaches and methodologies have been presented in the literature, dealing with I/O placement and electrical checking using flip-chip technology [5]–[10], [15], also with chip-package codesign [12], [11], [24] and pad assignment [13]. Recently, [14] further considered the building cost of I/O buffer blocks in flip-chip design, but planned I/O buffer blocks were not legalized.

It is hard to tell which technique can provide better design cost edge in usually concerned perspectives, such as area, performance, technology, and packaging cost. In this paper, we present a methodology to convert a previous peripheral bonding design to an area-IO flip-chip design. It is based on the I/O buffer modeling concept and the I/O planning algorithm to legalize I/O buffer blocks with core placement. The idea is to generate the positions of I/O buffer blocks (based on force-directed approaches [18], [19]), then use our legalizer (inspired by Mongrel [20], [23]) to minimally correct the placement, without sacrificing much of the previous optimization in the original core placement. The experimental results have shown that we have achieved smaller area and I/O wirelength in area-IO flip-chip configuration, compared with the same design previously made for peripheral bonding configuration.

The rest of this paper is organized as follows. Section II describes the I/O buffer placement considerations, our model concept for I/O buffer block, and problem formulation. Section III presents our chip I/O planning algorithm. Section IV shows the experimental results and we discuss some extension, and conclude the paper in Section V.

Manuscript received January 25, 2006; revised August 13, 2006 and February 24, 2007. This work was supported by the National Science Council of Taiwan ROC under Grants NSC 95-2220-E-009-007, NSC 96-2220-E-009-013, and NSC 96-2220-E-009-016.

C.-Y. Chang is with Sunplus Technology, Hsinchu 300, Taiwan (e-mail: joe.chang@sunplus.com).

H.-M. Chen is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: hmchen@mail.nctu.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2007.912202

II. Area-Array I/O Buffer Block Placement

While performing area-array I/O buffer block placement, we can place those buffer blocks anywhere in the design. However, some issues including the forbidden minimal spacing between electrostatic discharge (ESD) structures and active devices should be considered. We describe the I/O buffer modeling concept to model the area taken by buffer itself, other protected circuits, and clearance region, at the same time considering the driving strength. Then, we formulate our problem in chip I/O planning for placement legalization and performance issues.

A. I/O Buffer Block Modeling

We use an example for flip-chip style design to justify our objective in I/O buffer block modeling. This chip [16, Fig. 7] adopted flip-chip design to reduce 20% of die area comparing to the peripheral pad design with 114 standard I/O pads along the perimeter. This design grouped the most part of I/O buffers at the center of the die to avoid the cost caused by the forbidden minimal spacing between ESD structures and active devices due to the foundry rules.

We consider our I/O buffer blocks as I/O macros which may contain several signals, ESD protection structure, latch-up ring, and some testing logics. We adopt the universal configuration in [16] for all our I/O buffer blocks, an example is shown in [16, Fig. 11], however those scaled constraint components are varied by design cases. We also apply the I/O regimes from [17] for our I/O buffer block model. Based on the number of the cells it connected, we build a lookup table for various types of buffer blocks. In order to fit with the style of standard cell design and simplify the problem, we build our I/O buffer blocks with the same height as the original row height of standard cell design, although real I/O buffer block may exceed the row height of standard cell design. The way we model our I/O buffer block is shown in Fig. 1.[1] The size of each buffer block comes with two parts. One is the minimal spacing between ESD structures and active devices. The other part is the area of the buffer block itself with ESD structure, latch-up ring, testing logic, and driver circuit. It is obvious that more buffers in a buffer block can reduce the area for minimal spacing between ESD and active devices, and the area for shared ESD and testing logic. We proportionally scale those spaces to reflect the benefits of buffer block generation.

[1]In real designs, the output buffer usually has bigger area than input buffer because of the requirement for driven ability. In our buffer model, we simply treat those two kinds of I/O buffer blocks as the same type.

1063-8210/$25.00 © 2008 IEEE

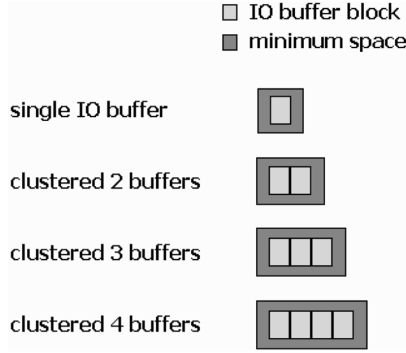

Fig. 1. Our I/O buffer block model. For example, the I/O buffer block which clustered four buffers means a signal buffer block which is able to drive four cells. The size of each buffer block comes with two parts. One is the minimal spacing between ESD structures and active devices. The other is the area of the buffer block itself with ESD structure, latch-up ring, testing logic, and driver circuit.

B. Problem Formulation

Our experiment focuses on I/O buffer placement in flip-chip design and we perform our placement in row-based design. We assume that all signal bumps can be assigned to any bump balls which are placed at predefined location. All the I/O signals are connected to cells through the I/O buffer blocks.

There are various constraints which should be concerned, including power/signal integrity, timing, I/O standard related, and floorplan induced region [24]. In this paper, we plan to reduce the timing related cost from the following objective functions Γ_1 and Γ_2:

$$\Gamma_1 = \sum_{j=1}^{n} d_j^{\mathrm{io}} + \sum_{j=1}^{m} d_j \quad (1)$$

$$\Gamma_2 = \left| \max_{1\le j\le n} d_j^{\mathrm{io}} - \min_{1\le j\le n} d_j^{\mathrm{io}} \right| \quad (2)$$

where Γ_1 gives the sum of wirelength of I/O nets and cell nets; d_j^{io} is the wirelength from signal bumps to cells via I/O buffer blocks in the I/O net; and d_j is the wirelength between all cells in the net. Γ_2 gives the I/O signal skew by the absolute value between longest path and shortest path in I/O nets. In our experiment, we use the linear delay model to determine the path delay.

The problem we are concerned about is described as follows. Given an initial standard cell placement based on peripheral I/O design, a set of I/O buffers (which has corresponding set of signal bumps) $\mathrm{IO} = \{\mathrm{io}_1, \ldots, \mathrm{io}_n\}$, a set of signal bumps $\mathrm{S} = \{s_1, \ldots, s_n\}$, a certain models for I/O buffer blocks, and a set of nets $\mathrm{N} = \{n_1, \ldots, n_m\}$, find an area-IO solution to reduce the I/O wirelength from signal bumps to cells via buffer blocks, with minimal placement correction. The signal skew is constrained given a user-specified skew range (USSR).

III. Algorithm for Chip I/O Planning in Design Migration

Our approach provides a novel methodology for migrating nonarea-IO designs into area-IO solutions, which can fully take advantage of flip-chip architecture. We take a given initial standard cell placement, then model the size of each I/O buffer block (follow the I/O macro modeling in [16, Sect. II-A]), finally place I/O buffer blocks into the design and assign signal bump to every I/O buffer block to achieve the requirements of flip-chip design. The flow of our methodology is shown in Fig. 2. The following sections describe the I/O buffer block planner and global legalization process for the die dimensions constraint.

A. I/O Buffer Block Planner

Our I/O buffer block planner is based on the approach in force-directed placement [18], [19]. This placement method applies an iterative placement procedure. The process starts with an initial placement and then selects a cell at a time to place the cell at its zero-force position which is computed numerically according to the connection with other cells. If the zero-force location is occupied by other cell, the placer will move the cell to another ideal location which is free to move in, or move the cell which occupies the zero-force location to another ideal location. In this paper, we use the geometry center as zero-force location of target cell.

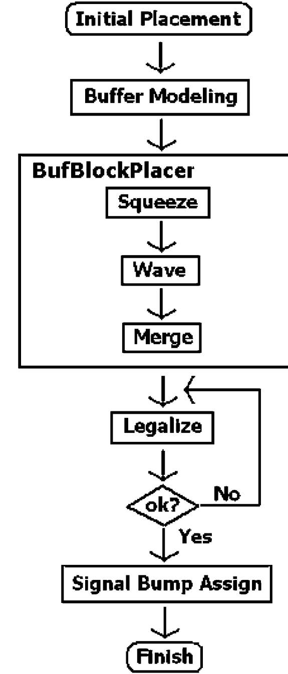

Fig. 2. Our design migration methodology flow. The flow includes I/O buffer block modeling, chip I/O planning and global legalization, and signal bump assignment steps.

The I/O buffer block planner/placer is implemented as follows. We first compute the geometry center of each net, then we order the nets by the position of their geometry center from bottom to top then left to right. Second, we determine the size of the buffer block of each net by the table we made in buffer modeling. We place buffer blocks at the geometry center of every net to minimize the longest path from cells to the buffer block and also reduce the interconnection length on the side. We have the following three operations to place those I/O buffer blocks into the core:

- *Cell Squeeze*: squeeze away the cell which occupied the location;
- *Buffer Block Merge*: merge two nearby buffer blocks into a single buffer block;
- *Local Legalization*: legalize the row length of the local rows.

We use *Cell Squeeze* to squeeze the cell away from its location and place the buffer in that location. *Buffer Block Merge* is to merge two buffer blocks into one buffer block if they are physically neighbored. Merging two different buffer blocks together can reduce the total area from our lookup table. After some operations of *Cell Squeeze* or *Buffer Block Merge*, the local rows may exceed the constraint of the length of row, we can use *Local Legalization* to adjust them to maintain the rectangular shape of the chip.

1) Cell Squeeze Operation: Once we determine the location of the buffer block, we have to move out the cell which occupied the target location. We focus on the movement of the cell while it has been squeezing away. Our *Cell Squeeze* operation has three directions to squeeze cells: right, up, or down. If we squeeze the cell to the right, all the cells on the right side of it including the cell itself will shift

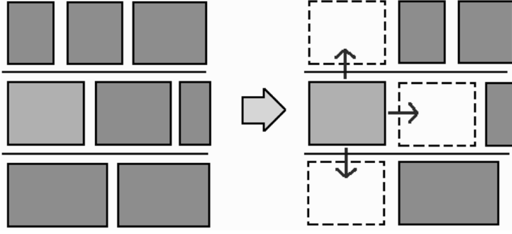

Fig. 3. *Cell Squeeze* operation has three directions to squeeze cells in $BufBlockPlacer$: right, up, and down.

right. If we squeeze the cell to the up or down, the cell will move to the target row and take the same operation (*Cell Squeeze*) to the right. The offset distance of shift right is the width of the cell which is squeezed. The operation of *Cell Squeeze* is illustrated in Fig. 3.

We calculate the costs of *Cell Squeeze* in all three directions by evaluating the weights of each cell move and consequent moves, then we choose the least cost as target movement. The cost function (weight) we use in this operation is

$$\text{cost} = \sum_{\text{all nets}} (\text{HP}_{\text{newBBox}} - \text{HP}_{\text{orgBBox}}) - \lambda W_{\text{sqCell}} \quad (3)$$

where HP means half-perimeter of the bounding box of the net for that squeezed cell, W means the width of the squeezed cell, and λ is to tradeoff the runtime of planner and I/O wirelength reduction. There are three situations for cell weight evaluation. First, when we move out one cell from current row, the cells on the right side of it should be drawn left. At the same time, when this cell move into the next row, the cells on the right side of the target slot should be pushed right. Second, the width of the cell which will be moved to the other row will affect the efficiency of local and global legalization. Moving bigger cell out of the longest row means less legalization effort will be needed. As a result, bigger cell can be evaluated as smaller weight. Third, if the movement of the cell has changed the bounding box of the net it connected, the weight is updated. *Cell Squeeze* will choose the least cost direction to squeeze the cell.

After the operation of *Cell Squeeze*, the position of cell and free space will be updated and the free space needed for the buffer block will be accounted for. If the free space needed is less than the width of the cell we plan to squeeze, we simply shift the cells right for the distance of free space needed instead of choosing a direction to squeeze. Once we get enough free space for the buffer block, we place it into the free space.

2) Buffer Blocks Merging Operation: After some operations performed in *BufBlockPlacer* (see Fig. 2), it is possible that two buffer blocks are physically neighbored. In order to reduce as much area as possible, we use *Buffer Block Merge* to merge two buffer blocks into one. Due to sharing the clearance region of minimum spacing between ESD structures and active devices, the area of the merged single buffer block is less than the sum of two individual buffer blocks. We use lookup table to determine the size of the merged buffer block, as illustrated in Fig. 1.

3) Local Legalization Operation: The length of some certain rows may exceed the constraint of the length of row after some operations of *Cell Squeeze* and *Buffer Block Merge*. We develop a local legalization process which is inspired from *Mongrel* [20], [23] to fix this problem. The idea of ripple moves in [20] and [23] is suitable in our framework and we modify the idea to fit in. Once we squeeze some cells away from the positions they belong, we use *Wave* operation to move cells in the nearby area to reduce the impact on the change of length of the row caused by the *Cell Squeeze* operation. In our local legalization procedure, we start with the initial placement (after *Cell Squeeze*) and then

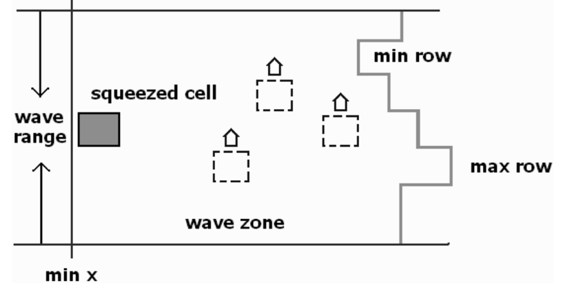

Fig. 4. Operation *Wave* sequentially moves cells from min row to max row in one wave zone during local legalization. Min and max rows mean that the minimum and maximum lengths of rows in wave zone.

sequentially move each least cost cell to its relaxed target location. The cost evaluated is the same as in *Cell Squeeze*. We can then produce a feasible placement with free space for our buffer block after each move.

To be more exact, in *Wave* operation, a wave zone will be created by the x and y dimensions: for x dimension, starting from the x-coordinate of the cell which has been squeezed in; for y dimension, user-specified parameter $WaveRange$ is used. If there is any buffer block in the wave zone, we redefine the range of the wave zone to exclude buffer blocks. As shown in Fig. 4, we sequentially move each least connectivity degree cell from the longest row to the shortest row. Every selected cell will move up/down to the next row and squeeze the cells to the right which are on the right side of that selected cells. The order we squeeze the cell is from bottom to top, then from left to right. As a result, the move in *Wave* operation will not affect the position of the buffer blocks which have been placed. The complexity of this operation is $O(M(N \log N + T(\text{CS})))$, where M is the range of $WaveRange$ (in rows), N is number of cells in a row, and $T(\text{CS})$ is the runtime of *Cell Squeeze*.

B. Global Legalization and Signal Bump Assignment

After performing *BufBlockPlacer*, the placement may still violate the constraint of the length of row. We can use the same heuristic in local legalization to solve the problem. Instead of local wave zone for *Wave* operation, this procedure deals with the whole chip. We sequentially move least cost cell to the next row, from the longest row to the shortest row. This iterative process is terminated when there is no violations on the constraint of the length of row or the number of iteration exceeding a user-specified count.

Once we finish the placement of buffer blocks, we apply a heuristic to reduce signal skew by assigning new signal bumps to those buffer blocks. Since we adopt the flip-chip design, the locations of the signal bumps are uniformly over the chip. In the beginning of this process, a set of locations in a gird are generated for signal bumps to select. We apply two-stage approach to determining the critical signal path. In the first step, we define the longest path from buffer block to the cell it connected in the design as a maximum delay path. In the second step, we select a closest signal bump location to that maximum delay path to minimize the delay of the maximum delay path. Here we get a maximum signal delay *MaxDelay* for all other I/O nets. We employ a parameter USSR (user-specified skew range) to control the skew for all input/output nets. After we finish the signal bump assignment for the net with longest path, we continue the next assignment for the net with the longest path in the rest of nets until all nets are processed. Fig. 5 illustrates an example for signal bump assignment.

IV. Experimental Results

We implemented our algorithm in C++ programming language and our experimental platform is Intel Pentium IV 2.4-GHz CPU with

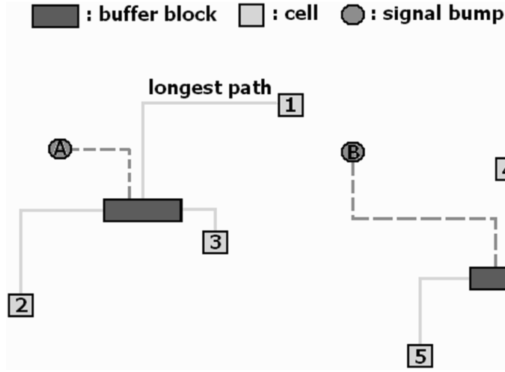

Fig. 5. Example for signal bump assignment after global legalization. Cells 1–3 are in the same net, so they all connect to the same buffer block. Cells 4 and 5 are in another net, so they connect to another buffer block. Cell 1 to its buffer block is the longest path, so we set it as *MaxDelay*. We assign a signal bump A for it. Cell 5 is the longest path in another net, so we assign signal bump B to it without exceeding *USSR* compared with *MaxDelay*.

TABLE I
NUMBER OF CELLS, NETS, AND I/O TERMINALS IN SOME MCNC STANDARD CELL PLACEMENT BENCHMARKS

Benchmark	Cells	Nets	I/Os
struct	1952	1920	64
biomed	6514	7052	97
industry2	12637	13419	495

TABLE II
AREA AND I/O WIRELENGTH COMPARISONS BETWEEN OUR APPROACH AND THE PERIPHERAL DESIGN FOR MCNC BENCHMARKS SHOWN IN TABLE I. THIS SHOWS OUR DESIGN MIGRATION ACHIEVES LESS AREA AND I/O WIRELENGTH IN FLIP-CHIP CONFIGURATION. SKEW RESULTS ARE SHOWN AS WELL AND THEY ARE WITHIN USSRS

Circuit	Initial area (um^2)	Peripheral design Area (um^2)	Peripheral design IO WL (um)	BufBlockPlacer Area (um^2)	BufBlockPlacer IO WL (um)	BufBlockPlacer Skew (um)	Improvement in Area (%)	Improvement in IO WL (%)
struct	11.47E+6	15.91E+6	656856	13.22E+6	613726	5250	16.91	6.57
biomed	52.48E+6	61.57E+6	2.992E+6	55.33E+6	2.605E+6	13094	10.13	12.94
industry2	10.45E+7	16.96E+7	1.416E+7	10.90E+7	9.074E+6	19564	35.73	35.92

TABLE III
RUNTIME COMPARISONS FOR OUR I/O BUFFER PLACER IN ONE MCNC BENCHMARK $industry2$. WE CAN ADJUST A TRADEOFF PARAMETER TO LET LEGALIZATION PROCESS CONSIDER ONLY WIRELENGTH OPTIMIZATION, OR OPTIMIZATION IN BOTH WIRELENGTH AND RUNTIME

industry2 with legalization consideration	Area (um^2)	IO WL (um)	Runtime (sec)
Wirelength only	11.08E+7	8.942E+6	2156
Wirelength and runtime	10.90E+7	9.074E+6	427

1.5 GB memory. The initial placements based on some MCNC benchmarks (see Table I) are obtained from the placer *FENG SHUI*[21], with aspect ratio 1.0. The number of signal bumps of the flip-chip design are scaled from IBM SA-27E area-array copper technology [5]. We compare our results with the peripheral design for area and I/O wirelength. The pad size of the peripheral design is 100×100 um^2 and the pad pitch is 100 um [10]. The minimum space between I/O pads and the core in peripheral design is set the same as the row height of the standard cell.

Table II shows the experimental results in area and I/O wirelentgh (IO WL)[2] on MCNC benchmarks summarized in Table I. The width of the area in flip-chip design is equal to the length of longest row. Since

[2]IO WL is the wirelength estimation from I/O buffers to other logic cells. We apply half-perimeter I/O net wirelength (HPWL) estimation.

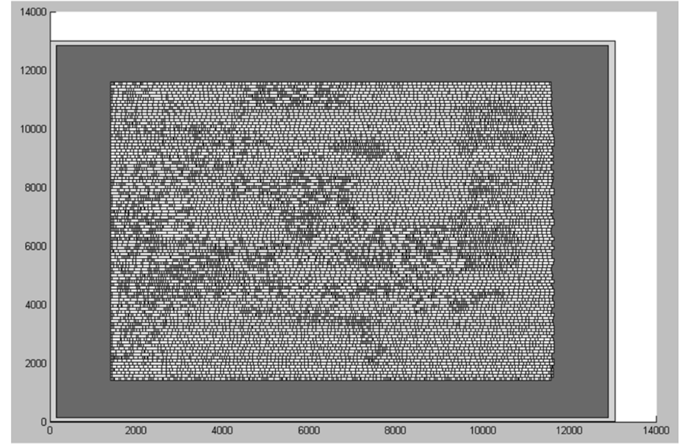

Fig. 6. Placement of $industry2$ in peripheral design.

Fig. 7. Final placement result of $industry2$ in Area-IO flip-chip design by our approach. The black dots represent the signal bumps.

$industry2$ is a big design with more than 400 I/Os, the size of initial peripheral design is not compatible with such amount of I/Os, we increase the space between I/O pads and the core for $industry2$ in peripheral design to fit the amount of the I/O pad. We obtain better I/O timing performance under skew constraints by smaller I/O wirelength. The wirelength from I/O nets to pads in peripheral design are estimated by the average distance from the net to the boundary of I/O pads. We can also use a parameter to tradeoff the runtime and I/O wirelength reduction, the result is shown in Table III. The final placement results for circuit $industry2$ for both peripheral and flip-chip design are shown in Figs. 6 and 7.

V. CONCLUSION

We have presented our chip I/O planning and legalization algorithm in design migration from peripheral style to area-IO flip-chip design. Experimental results have shown that our algorithm has better area and I/O wirelength under skew constraints, compared with peripheral design in high I/O count circuits.

For future works, we plan to consider signal integrity and other important concerns in [14] and [24]. Reference [14] mentions the I/O clustering for design cost optimization and power supply noise constraint preservation, and [24] presents the concerns in signal integrity, timing, I/O standard related, and floorplan induced region. Simultaneous ESD and predefined signal to bump balls assignment will be our target in the next implementation level as well.

ACKNOWLEDGMENT

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

REFERENCES

[1] A. Chandrakasan, W. J. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*. Piscataway, NJ: IEEE Press, 2001.

[2] P. Dehkordi and D. Bouldin, "Design for packageability: The impact of bonding technology on the size and layout of VLSI dies," in *Proc. Multi-Chip Module Conf.*, 1993, pp. 153–159.

[3] 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 Symp. Field Program. Gate Arrays*, 1995, pp. 17–23.

[4] P. A. Sandborn, M. S. Abadir, and C. F. Murphy, "The tradeoff between peripheral and area array bonding of components in multichip modules," *IEEE Trans. Compon., Packag., Manuf. Technol.—A*, pp. 249–256, 1994.

[5] P. H. Buffet, J. Natonio, R. A. Proctor, Y. H. Sun, and G. Yasar, "Methodology for I/O cell placement and checking in ASIC designs using area-array power grid," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2000, pp. 125–128.

[6] G. Yasar, C. Chiu, R. A. Proctor, and J. P. Libous, "I/O cell placement and electrical checking methodology for ASICs with peripheral I/Os," in *Proc. IEEE Int. Symp. Quality Electron. Des.*, 2001, pp. 71–75.

[7] R. Farbarik, X. Liu, M. Rossman, P. Parakh, T. Basso, and R. Brown, "CAD tools for area-distributed I/O pad packaging," in *Proc. IEEE Multi-Chip Module Conf.*, 1997, pp. 125–129.

[8] P. S. Zuchowski, J. H. Panner, D. W. Stout, J. M. Adams, F. Chan, P. E. Dunn, A. D. Huber, and J. J. Oler, "I/O impedance matching algorithm for high-performance ASICs," in *Proc. IEEE Int. ASIC Conf. Exhibit*, 1997, pp. 270–273.

[9] R. J. Lomax, R. B. Brown, M. Nanua, and T. D. Strong, "Area I/O flip-chip packaging to minimize interconnect length," in *Proc. IEEE Multi-Chip Module Conf.*, 1997, pp. 2–7.

[10] C. Tan, D. Bouldin, and P. Dehkordi, "Design implementation of intrinsic area array ICs," in *Proc. 17th Conf. Adv. Res. VLSI*, 1997, pp. 82–93.

[11] J. Mcgrath, "Chip/package co-design: The bridge between chips and systems," *Adv. Packag. Mag.*, Jun. 2001 [Online]. Available: http://ap.pennet.com/Articles/Article_Display.cfm?Section= Archives&Subsection=Display&ARTICLE_ID=103319& KEYWORD=joel%20mcgrath

[12] J. C. Parker, R. J. Sergi, D. Hawk, and M. Diberardino, "IC-package co-design supports flip-chips," *EE Times*, (2003, Nov.) [Online]. Available: http://www.eedesign.com/story/OEG20031113S0055

[13] K.-Y. Chao and D. F. Wong, "Signal integrity optimization on the pad assignment for high-speed VLSI design," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, 1995, pp. 720–725.

[14] H.-M. Chen, I.-M. Liu, D. F. Wong, M. Shao, and L.-D. Huang, "I/O clustering in design cost and performance optimization for flip-chip design," in *Proc. IEEE Int. Conf. Comput. Des.*, 2004, pp. 562–567.

[15] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "I/O buffer placement methodology for ASICs," in *Proc. IEEE Int. Conf. Electron., Circuits Syst.*, 2001, pp. 245–248.

[16] T. Schaffer, A. Glaser, and P. D. Franzon, "Chip-package co-implementation of a triple DES processor," *IEEE Trans. Adv. Packag.*, pp. 194–202, Feb. 2004.

[17] 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. IC/Package Des. Integr.*, 1998, pp. 93–98.

[18] M. Hanan and J. M. Kurtzberg, , M. A. Breuer, Ed., "Placement techniques," in *Design Automation of Digital Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1972, pp. 213–282.

[19] N. Quinn and M. Breuer, "A force-directed component placement procedure for printed circuit boards," *IEEE Trans. Circuits Syst.*, vol. CAS-26, no. 6, pp. 377–388, Jun. 1979.

[20] S.-W. Hur and J. Lillis, "Mongrel: Hybrid techniques for standard cell placement," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, 2000, pp. 165–170.

[21] P. H. Madden, "Reporting of standard cell placement results," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 2, pp. 240–247, Feb. 2002.

[22] C. Tan, D. Bouldin, and P. Dehkordi, "An intrinsic area-array pad router for ICs," in *Proc. 10th Ann. IEEE Int. ASIC Conf.*, 1997, pp. 265–269.

[23] S.-W. Hur, T. Cao, K. Rajagopal, Y. Parasuram, and B. Halpin, "Force directed mongrel with physical net constraints," in *Proc. ACM Des. Autom. Conf.*, 2003, pp. 214–219.

[24] J. Xiong, Y.-C. Wong, E. Sarto, and L. He, "Constraint driven I/O planning and placement for chip-package co-design," in *Proc. IEEE Asia-South Pacific Des. Autom. Conf.*, 2007, pp. 207–212.