Ant System-Corner Insertion Sequence: An Efficient VLSI Hard Module Placer

Chyi-Shiang HOO, Kanesan JEEVAN, Velappa GANAPATHY, Harikrishnan RAMIAH
Department of Electrical, Faculty of Engineering, University of Malaya, Lembah Pantai, 50603 Kuala Lumpur, Malaysia.
francioshoo@siswa.um.edu.my

***Abstract*—Placement is important in VLSI physical design as it determines the time-to-market and chip's reliability. In this paper, a new floorplan representation which couples with Ant System, namely Corner Insertion Sequence (CIS) is proposed. Though CIS's search complexity is smaller than the state-of-the-art representation Corner Sequence (CS), CIS adopts a preset boundary on the placement and hence, leading to search bound similar to CS. This enables the previous unutilized corner edges to become viable. Also, the redundancy of CS representation is eliminated in CIS leads to a lower search complexity of CIS. Experimental results on Microelectronics Center of North Carolina (MCNC) hard block benchmark circuits show that the proposed algorithm performs comparably in terms of area yet at least two times faster than CS.**

***Index Terms*—Design, system, aided, floorplanning, VLSI, representation, circuits, algorithm, scale, optimization.**

I. INTRODUCTION

In the modern IC design flow, a placement of an IC is a schematic representation of placement of its major functional blocks. As the Very Large Scale Integration (VLSI) chip keeps shrinking in size, many factors such as the total area [1-2], wirelength [3-5], power consumption [6-7] and congestion reduction [8-9] will affect the reliability and efficiency of the chip. Hence, to cope up with these issues, efficient placement plays a very crucial role as far as the quality of the VLSI design is concerned.

The VLSI placement design problem is well-known as the NP-hard problem and hence it is difficult to find exactly optimal solution in practical applications [10-11]. In order to solve this combinatorial optimization problem, placement layout is tackled mathematically in order to be optimized using the tools such as mathematical optimization or artificial intelligence (AI) technique. Many approaches [1], have been proposed in the literature with different modeling representations [12-15] and optimization methods [2], [5], [16-17] to enhance the quality of the placement design. To facilitate a good placement, it is necessary to develop an effective model for blocks placement to reduce the dead space area as well as minimizing the placement runtime.

II. PROBLEM DESCRIPTION

Let a set of n rectangular modules $B = \{b_1, b_2, \ldots, b_n\}$ and w_m and h_m are the width and height of the module respectively of the module b_m. with the constraints of $1 \leq m \leq n$. In the VLSI placement problem, it is an assignment where all the modules of b_m are arranged to form a layout without any overlap among the modules placed. Usually this problem is to minimize the area and/or wirelength induced by the assignment of b_ms' locations, without extending the runtime significantly. Area is calculated by the smallest rectangle that can be formed to enclose all the modules while the wirelength is the summation of all the modules center-to-center interconnections. In this paper, area optimization with reduced runtime is the main objective.

III. CORNER INSERTION SEQUENCE

Similar to Corner Sequence (CS) [1], Corner Insertion Sequence (CIS) is a very effective nonslicing floorplan implementation. CIS consists of two tuples that denote the packing sequence of blocks and the corresponding corners to which the blocks are placed. Inheriting the sequence properties similar to CS, the complexity of CIS is proven to be $O(n!2^{n-1})$, comparatively lower than $O((n!)^2)$ of CS. With the reduction in search complexity, CIS algorithm can perform much faster than CS. Even though the search complexity of CIS is reduced, the preset boundary imposed to the placement enables CIS to search for more compacted solutions by regaining the search bound similar to CS, which will be explained in Section III (C). The higher complexity of CS is due to the redundancy of the CS representation. The redundancy of CS is now reduced by using CIS method.

A. Matrix Representation

Different from CS, CIS involves only corner edges associated with the most recently placed blocks, being considered in the placement. This in turn will reduce search complexity and result in faster optimization of placement. In this representation, blocks are placed one at a time according to a predefined sequence. After performing numerous permutations of blocks placements, a CIS matrix is formed. Let us consider a (n x 3) matrix CIS, shown in Figure 1, where n is the number of blocks placed in the layout. There are 3 configurations determining the placement of a particular block. The first column is the block number, which is block 1 to block n. The second column is the left neighboring blocks, which are x_1 to x_n, while the third column is the bottom neighboring blocks, which are y_1 to y_n.

B. Matrix-layout Inter-transformation

Assume a CIS matrix as given in Figure 1. Initially, we have only two dummy modules, b_y and b_x as our corner modules (CM) and $\{b_y, b_x\}$ as our corner edges. According to the sequence, module b_3 is placed onto the corner edge $\{b_y, b_x\}$, creating two extra contour edges $\{b_y, b_3\}$ and $\{b_3, b_x\}$. Now, we have b_y, b_3 and b_x as our CMs. Next, module b_2 is inserted onto the contour edge

Digital Object Identifier 10.4316/AECE.2013.01002

$$\text{CIS} = \begin{pmatrix} b_3 & b_y & b_x \\ b_2 & b_3 & b_x \\ b_1 & b_y & b_3 \\ b_4 & b_1 & b_2 \\ b_5 & b_4 & b_2 \end{pmatrix}$$

Layout-to-representation ↑ ↓ Representation-to-layout

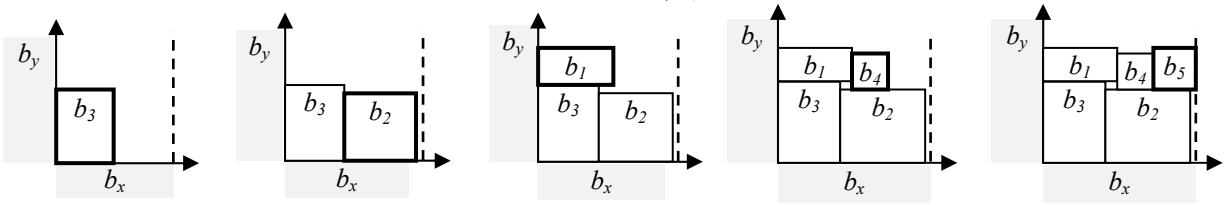

Figure 1: Transformation between matrix and layout (dotted line represents the boundary limit).

{b_3, b_x}, leading to four CMs, b_y, b_3, b_2 and b_x. However, in CIS representation, only selected CMs are used to derive corner edges. The corner edges must be associated with the most recently placed module; in this case, it is b_2. Hence, the arising corner edges should be {b_3, b_2} and {b_2, b_x}. In the third placement, module b_1 is placed onto the corner edge of {b_y, b_3} as the placement at corner edge {b_2, b_x} reaches the preset boundary of the placement. The CMs now consist of only b_y, b_1, b_2 and b_x. This process is continued until all the modules are placed and the process is shown in Figure 1.

C. Corner Update and Overlap Avoidance

For every placement, the coordinates of the corners have to be updated from time to time so that the algorithm can evaluate the exact location of the placement. The corner coordinates are so important since the algorithm will recognize the coordinates before the blocks are pushed onto its respective corners. If the coordinates were wrong, the placement might not be aligned with the desired locations, and hence causing overlapping and gaps between blocks.

Important advantage of corner coordinates updates is that no overlapping conditions occur in floorplan placement. Overlapping placement in a floorplan is defined as an extension of blocks over other blocks in the placement. To understand the significance of overlapping, the relationship between two blocks is defined in [18].

In order to avoid a long flat floorplan layout generation, a preset boundary is imposed. If the placement has reached the preset boundary, a new placement will commence from the most bottom-left corners available. This preset boundary not only prevents the floorplan from generating a long flat placement but also enable CIS to explore optimal solutions, with less search complexity. For example, the corner edge {b_y, b_3} in Figure 1 which is abandoned after the placement of module b_2, has been reconsidered after the modules placement meets the preset boundary. Consequently, the corner edges adjacent to this bottom-left corner edges will also be reconsidered. This indicates that even though CIS is bounded by the search complexity, previously neglected solutions are reconsidered and therefore, similar search bound as CS is obtained. By referring to Figure 1, the placement layout requires only one CIS matrix representation whereas CS generates more than one representation to model the same layout, as shown in Eq. (1). This example indicates that there is a redundancy in CS representation that leads to an increased search complexity.

$$\text{CS} = \begin{pmatrix} b_3 & b_y & b_x \\ b_2 & b_3 & b_x \\ b_1 & b_y & b_3 \\ b_4 & b_1 & b_2 \\ b_5 & b_4 & b_2 \end{pmatrix} \equiv \begin{pmatrix} b_3 & b_y & b_x \\ b_1 & b_y & b_3 \\ b_2 & b_3 & b_x \\ b_4 & b_1 & b_2 \\ b_5 & b_4 & b_2 \end{pmatrix} \quad (1)$$

Lemma 1: There is no overlapping issue occurring in CIS representation.

Proof: By updating the corner coordinates after each block is inserted into its placement position, the exact locations of the corners are always accurate. When the algorithm brings in a new block, it will be placed onto the exact corner, and by disqualifying the overlap cases, the overlap issue will never happen in the final floorplan. By using CIS representation, there is no need of any extra overlapping removal algorithm to cope up with this issue, and hence leading to a very brief and efficient algorithm.

Lemma 2: The CIS solutions' search complexity is bounded by $O(n!2^{n-1})$, where n is the number of blocks.

Proof: Since there are n blocks to be placed in a chip, there are $n!$ permutations in the placement sequence. Initially, there is only one corner available, which is left with a single choice of inserting the block. After the first block is inserted, two corner edges arise, giving two choices for the second block. After the second block is placed, the previous corner edges will be ignored and two new corner edges appear, leading to two choices for the third block. As the blocks' placements proceed, there are always two corner edges and hence, two possible choices for the next block. Therefore, the solution space searching complexity of the CIS is bounded by:

$$(n!) \times \underbrace{1 \times 2 \times \cdots \times 2}_{n} = (n!)2^{n-1}. \quad (2)$$

Lemma 3: The transformation between CIS matrix representation into placement uses $O(n)$ time, where n is the number of blocks.

Proof: The algorithm takes a constant time to insert a block into the designated corner. If there are n blocks, the time complexity to insert the blocks is $O(n)$. Therefore, the time complexity of the CIS is $O(n)$, which is linear.

IV. Ant System- Corner Insertion Sequence

Ant-based meta-heuristics approach is a powerful searching method, inspired by the foraging behavior of ant colony [19-20]. In general, the ant-based meta-heuristics algorithm consists of three parts, which are initialization, construction, and feedback [21]. After initializing the parameters in the AS algorithm, the ant k will construct the tour from city-i to city-j thoroughly based on the Roulette-wheel based probability P_{ij}^{k}, which is defined as:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{n} \Delta\tau_{ij}^{k} \quad (3)$$

where τ_{ij} is the pheromone intensity of the trail and N^k is the set of unvisited cities. If the distance between the cities i and j is denoted by d_{ij}, then η_{ij} is defined as:

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (4)$$

The parameters α and β are used to control the impingement of the pheromone τ_{ij} (global information) and η_{ij} (local information). After constructing the complete path, the ant k will update the pheromone based on formula given below:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{n} \Delta\tau_{ij}^{k} \quad (5)$$

where
p = evaporation constant, $0 < p < 1$,
C^k = length of the tour T^k, and

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{C^k}; & if \quad path \quad (i,j) \quad belongs \quad to \quad T^k \\ 0; & otherwise \end{cases} \quad (6)$$

The original Ant System was proposed [19] to handle the TSP problem which is one dimensional problem. However, VLSI placement is a two-dimensional problem and hence, in this work, some modifications are carried out. The variable d_{ij} in Eq. (4) is replaced with the instant whitespace created after placing a block and the C^k mentioned in Eq. (6) is referring to the final deadspace at the final floorplan.

V. Experimental Results

The proposed algorithm is implemented in C++ on Athlon 750, 750-MHz workstation with 512MB memory and tested under the commonly used MCNC benchmark circuits [22]. The comparisons are made on the following floorplanning algorithms: O-Tree [12], B*-tree [13], enhanced O-tree [23], CBL [14], TCG [15], CS [1], DPSO [24] and ESA [25]. The MCNC benchmarks are used as the standard. The area of a chip is defined as smallest rectangle that encloses all the modules while relative whitespace is ratio of the unutilized area to the total area of the chip. As can be seen from Table I, the results of AS-CIS are comparable with all the other representations in terms of area utilization. By referring the cases sampled from Table I, it is seen that AS-CIS shows improved results in terms of areas in 30 out of 38 cases (79%) as compared to other algorithms. By referring to Table I, the runtime for AS-CIS is much shorter compared to other representations. The runtime comparisons are illustrated in Figure 2. Based on the trendlines, the runtimes of all the floorplanning algorithms are becoming longer than AS-CIS when the size of the benchmark problems is increasing, except CBL. It is to be noted that CBL has performed poorly in terms of area, ranging from 0.82 to 1.89 times more than relative whitespace, which is evident from Table I.

TABLE I. MCNC Hard Module Benchmark Comparisons.

MCNC circuits		*apte*	*xerox*	*hp*	*ami*33	*ami*49
Area (mm^2)	O-tree	47.1	20.1	9.21	1.25	37.6
	B*-tree	46.92	19.83	8.947	1.27	36.80
	Enhanced O-tree	46.92	20.21	9.16	1.24	37.73
	CBL	-	20.96	-	1.20	38.58
	TCG	46.92	19.83	8.947	1.20	36.77
	DPSO	47.31	20.2	9.50	1.28	38.8
	ESA	47.37	19.83	8.94	1.24	36.50
	CS	48.5	20.4	9.6	1.25	38.2
	AS-CIS	46.92	19.83	9.03	1.21	37.58
Runtime (sec.)	O-tree	38	118	57	1430	7428
	B*-tree	7	25	55	3417	4752
	Enhanced O-tree	11	38	19	118	406
	CBL	-	30	-	36	65
	TCG	1	18	20	306	434
	DPSO	-	-	-	-	-
	ESA	1	3	7	24	53
	CS	29	40	27	476	2103
	AS-CIS	1	5	5	24	75

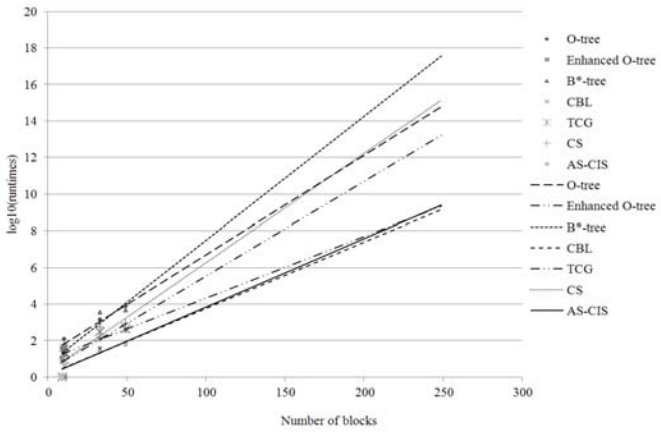

Figure 2: Runtime (in log. Scale) comparisons.

VI. Conclusion

In this paper, Ant System based Corner Insertion Sequence placer (AS-CIS) has been proposed which is capable of generating compact placements in a chip layout within a short period of time. A fast and comparable result is obtained using CIS because of the search and time complexities are limited to $O(n!2^{n-1})$ and $O(n)$ respectively. Though CIS's search complexity is reduced as compared to CS, CIS has the similar search bound by introducing preset boundary which enables the previous unutilized corner edges to become viable. Hence, a much less placement search complexity contributes to reduction in runtime as compared to other floorplan models, while maintaining the possible optimal solution search. The experimental results show that the overall runtime of the program is reduced considerably due to the simplicity of the algorithm proposed. On the other hand, the resulting placement of proposed AS-CIS is quite promising and is found to be

better than most of the existing floorplan models as discussed in Section V.

REFERENCES

[1] J. M. Lin, Y. W. Chang, and S. P. Lin, "Corner sequence–A p-admissible floorplan representation with a worst case linear-time packing scheme," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 4, pp. 8-12, 2003. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2003.816137.

[2] J. Liu, W. C. Zhong, L. C. Jiao, and X. Li, "Moving block sequence and organizational evolutionary algorithm for general floorplanning with arbitrarily shaped rectilinear blocks," IEEE Transactions on Evolutionary Computation, vol. 12, no. 5, pp. 630-646, 2008. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2008.920679.

[3] S. N. Adya, and I. L. Markov, "Fixed-outline floorplanning: enabling hierarchical design," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 6, pp. 1120-1135, 2003. [Online]. Available: http://dx.doi.org/ 10.1109/TVLSI.2003.817546.

[4] J. Cong, M. Romesis, and J. R. Shinnerl, "Fast floorplanning by look-ahead enabled recursive bipartitioning," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 9, pp. 1719-1732, 2006. [Online]. Available: http://dx.doi.org/ 10.1109/TCAD.2005.859519.

[5] J. M. Lin, and Z. X. Hung, "UFO: Unified convex optimization algorithm for fixed-outline floorplanning considering pre-placed modules," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 7, pp. 1034-1044, 2011. [Online]. Available: http://dx.doi.org/ 10.1109/TCAD.2011.2114531.

[6] Q. Ma, Z. Qian, E. F. Y. Young, and H. Zhou, "MSV-driven floorplanning," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 8, pp. 1152-1162, 2011. [Online]. Available: http://dx.doi.org/ 10.1109/TCAD.2011.2131890.

[7] Y. C. Chen, and Y. Li, "Temperature-aware floorplanning via geometric programming," Mathematical and Computer Modelling, vol. 51, no. 7-8, pp. 927-934, 2010. [Online]. Available: http://dx.doi.org/ 10.1016/j.mcm.2009.08.026.

[8] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning," in Proceedings of IEEE/ACM International Conference on Computer-Aided Design, 1999, pp. 358-262.

[9] A. Jahanian, and M. S. Zamani, "Metro-on-chip: an efficient physical design technique for congestion reduction," IEICE Electronics Express, vol. 4, no. 16, pp. 510-516, 2007. [Online]. Available: http://dx.doi.org/ 10.1587/elex.4.510.

[10] M. R. Garey, and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness", W. H. Freeman: San Francisco, 1979.

[11] S. Sahni, and T. Gonzalez, "P-complete approximation problems," Journal of the ACM, vol. 23, no. 3, pp. 555-565, 1976.

[12] P. N. Guo, T. Takahashi, C. K. Cheng, and T. Yoshimura, "Floorplanning using a tree representation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 2, pp. 281-289, 2001. [Online]. Available: http://dx.doi.org/ 10.1109/43.908471.

[13] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu, "B*-trees: A new representation for nonslicing floorplans," in Proceedings of Design Automation Conference, 2000, pp.458-463.

[14] X. Hong, G. Huang, Y. Cai, S. Dong, C. K. Cheng, and J. Gu, "Corner block list representation and its application to floorplan optimization," IEEE Transactions on Circuits and Systems-II: Express Briefs, vol. 51, no. 5, pp. 228-233, 2004. [Online]. Available: http://dx.doi.org/10.1109/TCSII.2004.824047.

[15] J. M. Lin, and Y. W. Chang, "TCG: A transitive closure graph-based representation for nonslicing floorplans," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 2, pp. 288-292, 2005. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2003.816137.

[16] J. G. Kim, and Y. D. Kim, "A linear programming-based algorithm for floorplanning in VLSI design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 5, pp. 584-592, 2003. [Online]. Available: http://dx.doi.org/ 10.1109/TCAD.2003.810748.

[17] C. Luo, M. F. Anjos, and A. Vannelli, "Large-scale fixed-outline floorplanning design using convex optimization techniques," in Proceedings of Asia and South Pacific Design Automation Conference, 2008, pp. 198-203.

[18] S. Alupoaei, and S. Katoori, "Ant Colony System application to macrocell overlap removal," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 10, pp. 1118-1123, 2004. [Online]. Available: http://dx.doi.org/ 10.1109/TVLSI.2004.832926.

[19] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in European Conference on Artificial Intelligence, 1991, pp. 134-142.

[20] C.-S. Hoo, H.-C. Yeo, K. Jeevan, V. Ganapathy, H. Ramiah, I.A. Badruddin, "Hierarchical Congregated Ant System for Bottom-up VLSI Placements," Engineering Applications of Artificial Intelligence, vol. 26, no. 1, pp. 584-602, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2012.04.007.

[21] C. W. Chiang, Y. Q. Huang, and W. Y. Wang, "Ant colony optimization with parameter adaption for multi-mode resource-constrained project scheduling", Journal of Intelligent and Fuzzy Systems, vol. 19, no. 4,5, pp. 345-358, 2008.

[22] J. Rabaey, Gigabyte systems research center 2005.

[23] Y. Pang, C. K. Cheng, and T. Yoshimura, "An enhanced perturbing algorithm for floorplan design using the O-tree representation," in Proceedings of International Symposium on Physical Design, 2000, pp.168-173.

[24] G. L. Chen, W. Z. Guo, and Y. Z. Chen, "A PSO-based intelligent decision algorithm for VLSI floorplanning," Soft Computing-A Fusion of Foundations, Methodologies and Applications, vol. 14, no. 12, pp. 1329-1337, 2009. [Online]. Available: http://dx.doi.org/ 10.1007/s00500-009-0501-6.

[25] J. Chen, and J. Chen, "A hybrid evolution algorithm for VLSI floorplanning," International Conference on Computational Intelligence and Software Engineering, 2000, pp. 1-4.