Partial Functional Manipulation Based Wirelength Minimization

Avijit Dutta and David Z. Pan, Comp Engineering Research Center, University of Texas at Austin

***Abstract*—In-place flipping of rectangular blocks/cells can potentially reduce the wirelength of a floorplan/placement solution without changing the chip area, In a recent work [Hao 05], the flipping optimization is solved through a binary decision diagram (BDD) based approach. However, the BDD-based approach is not scalable for large SOC designs with many blocks due to memory and runtime blow-up. This paper presents a new approach using the partitioned ordered partial decision diagrams (POPDD) for wirelength minimization. POPDD is based on a novel compact partial functional representation between flip configurations and corresponding wirelengths. By controlling the number of nodes allowed per POPDD and the iterations, easy trade-off between runtime/memory and accuracy/optimality can be achieved. Experimental results clearly demonstrate the efficiency of the proposed approach.**

***Index Terms*—OPDD, wirelength minimization, BDD, Symbolic algorithm**

I. Introduction

During floorplanning, both location and orientation of a set of rectangular blocks are decided such that no blocks overlap. The area of a floorplan is determined by the area of the rectangular bounding box containing all the blocks. The wire length is determined by the sum of the lengths of all the nets. Typically half perimeter wire length (HPWL) is used as an estimate of the total wire length since accurate wire length information is not available during floorplanning. In [Shahooker 91] it was shown that the orientations of the macro cells can greatly impact the total wire-length. Flip operation on blocks can be performed either along x-axis or y-axis or both unless the orientations of the blocks are pre-specified. In this paper we use HPWL as a measure of wire-length for each net.

One approach to minimize wire-length is to flip each block successively and keep the floorplan with a smaller HPWL. Such greedy algorithms [Yao 90] usually produce poor results [Hao 05]. Better heuristic approaches, e.g., simulated annealing [Kim 91], genetic algorithm [Yan 96] typically provide better solutions. [Funabiki 98] introduced an evolutionary neural network based solution. Another heuristic approach using a fuzzy c-means clustering was presented in [Yan 91]. However since the search space is typically very large, it is difficult to derive parameters to properly guide the search using these heuristics. Since only a small set of configurations provide optimal solutions, heuristic approaches may still fail to provide good solutions.

In [Cheng 91], [Yan 93], the block flipping problem has been proved to be NP-complete. Pure enumeration based approaches are not applicable because the total number of flip configurations is exponential in the number of blocks and even for a moderate number of blocks this approach is intractable. Several other algorithms have been proposed e.g., graph decomposition [Cheng 91], shortest path computation on the directed acyclic graph representation of the floorplan [Chong 93]. All these algorithms are heuristics.

In [Jeong 91] an integer linear programming based optimal solution was presented which could provide an optimal solution up to 33 blocks. However the runtime complexity was not reported. Another optimal solution was presented in [Yamada 88].

In [Hao 05] an optimal solution was presented based on a compact representation of different flip configurations using a binary decision diagram (BDD) [Bryant 86]. This approach can provide a flip-optimal wire-length solution for up to 20-30 blocks. In [Kim 91] it was observed that some pins can not be corner of a net in any flip configuration. [Hao 05] uses this information to further reduce the BDD size. There are several advantages of the method proposed in [Hao 05] compared to the previous approaches. By compactly representing the flip configurations using the BDDs, it avoids repeated computations. The HPWL values of a net corresponding to the different configurations are computed in parallel during BDD traversal. It also allows dynamic pruning of the unpromising solutions by first computing the HPWL at a lower resolution and later on at a higher resolution only for the promising solutions. To apply the algorithm in [Hao 05] for large floorplans, the number of blocks considered for flipping has to be limited to avoid runtime and memory blowup.

Full functional representations, e.g., BDDs inevitably suffer from the problems of blow-up both in the memory and the runtime. The number of nodes required can be exponential in the number of the variables in the worst case. For the method proposed in [Hao 05], the BDDs corresponding to all the nets have to be kept in the memory to calculate the final summation and for selecting the floorplan with the mimimum wire-length. The BDDs for the individual nets can also be large for nets having a large number of boundary-terminals. The terminals which appear at a boundary (of the bounding box of the net) in at least one of the configurations are referred

to as the boundary terminals. Due to these limitations, the full BDD based method may not be applicable for larger floorplans with a large number of flippable blocks.

In [Ross 90] the ordered partial decision diagram [OPDD] was introduced. Unlike a BDD the number of nodes in an OPDD is bounded by a constant. An extra terminus called the unknown (U) terminus is introduced. In this paper, to avoid the problem of memory blowup associated with the BDDs, the OPDD representation is explored. The block orientations that can not be represented within the memory limit are represented by the paths ending at the U-terminus. The paths ending at the 1-terminus denote the valid orientations which can be represented under the memory constraint.

In this paper we propose a method to compactly represent HPWLs of each net corresponding to the different flip configurations of the blocks using the zero suppressed partitioned ordered partial decision diagram (POPDD). The 0-paths in the OPDD are not included to make it zero suppressed thereby saving on memory. Furthermore, the variable set is partitioned into two groups i.e., the block variables and the coordinate variables. Due to the partitioning not all possible variable orderings are possible. Same variable order is maintained along every path from the root to a terminal node (ordered). We present a depth first search (DFS) based ordering scheme to iteratively build POPDDs for each net representing a partial set of flip configurations and the corresponding HPWLs. The bound on the number of nodes and the number of the iterations determine the accuracy of the method. Unlike the method in [Hao 05] the memory requirement is constant as the OPDD sizes are bounded and also the proposed approach does not require the number of the flippable blocks to be bounded. At the same time the proposed method retains all the advantages of the method proposed in [Hao 05].

II. Proposed Algorithm

In this section we describe our proposed approach to minimize the wirelength of a floorplan consisting of rectangular blocks using only the flip operation. First we describe the main steps involved in the optimization process and a few basic definitions. Next we show how the POPDDs compactly represent several flip configurations and how they differ from the BDD based approach. We also present a partitioning and a variable ordering scheme based on depth first search (DFS) and gate input ordering.

To calculate the HPWL of a floorplan for a given block orientation, for each net all the vertex positions are calculated and the corresponding HPWLs are calculated. Next the HPWLs of all the nets are added to get the total HPWL. These steps have to be repeated for each flip configuration of the blocks to find out the minimal cost configurations. The HPWL is defined as the sum of $HPWL_X$ and $HPWL_Y$ where:

$HPWL_X = \Sigma_i(\max_j(X_{i,j}) - \min_j(X_{i,j}))$

$HPWL_Y = \Sigma_i(\max_j(Y_{i,j}) - \min_j(Y_{i,j}))$

where *i* is the index of each net and *j* is the index of each terminal. The terminals could either be connected to the frame (pad) or to the blocks (pin). The block or the frame to which the terminal is connected is called the *host* of the terminal. For each block the bottom left coordinate, the height and the width of the host can be found from the initial floorplan. Given the relative position of the terminal in the block, the actual coordinate of the terminals can simply be calculated as:

$X = x_{host} + width * px$

$Y = y_{host} + height * py$

If the block is flipped along *X*-axis then

$X = x_{host} + width * (1 - px)$

If the block is flipped along *Y*-axis then

$Y = y_{host} + height * (1 - py)$

where *px* and *py* are the relative *x* and *y* coordinates of the terminal with respect to the host block and can be any real number in the range (0,1). For the case when only in place flip operation is allowed, the $HPWL_X$ and $HPWL_Y$ are independent of each other and can be calculated separately. For the rest of the paper we will only consider $HPWL_X$. $HPWL_Y$ can be computed the same way. [Hao 05] introduced a compact way to compute HPWL for all the flip configurations using BDDs. Consider the floorplan shown in Fig. 1 with 4 blocks and 2 nets. Each block is identified with a variable (*w0* through *w3*). If a block is flipped along the *x*-axis, it is represented by the same variable complemented. Let *x0* through *x3* be the variables representing the binary values of each coordinate of the terminals of each net. Note than only boundary terminals have to be considered. Non-boundary terminals do not impact the $HPWL_X$ of the nets. The following equations represent the relationship between the terminal coordinates and the flip configuration of the host blocks:

F_{net1} = *w0(4)+(!w0)(6)+w1(11)+(!w1)(14)+ w2(3)+(!w2)(2) +(10)*

= *(w0)(!x3)(x2)(!x1)(!x0) + (!w0)(!x3)(x2)(x1)(!x0) + (w1)(x3)(!x2)(x1)(x0) + (!w1)(x3)(x2)(x1)(!x0) + (w2)(!x3)(!x2)(x1)(x0) + (!w2)(!x3)(!x2)(x1)(!x0) +(x3)(!x2)(x1)(!x0)* (1)

F_{net2} = *w0(2)+(!w0)(8)+w1(13)+(!w1)(12)*

= (w0)(!x3)(!x2)(x1)(!x0) + (!w0)(x3)(!x2)(!x1)(!x0) + (w1)(x3)(x2)(!x1)(x0) + (!w1)(x3)(x2)(!x1)(!x0) (2)

Note that block 3 (w3) does not appear in the equation for net1. This is because the corresponding terminal coordinate appears exactly at the middle of the block w3 and hence is independent of the flip configuration of the block.

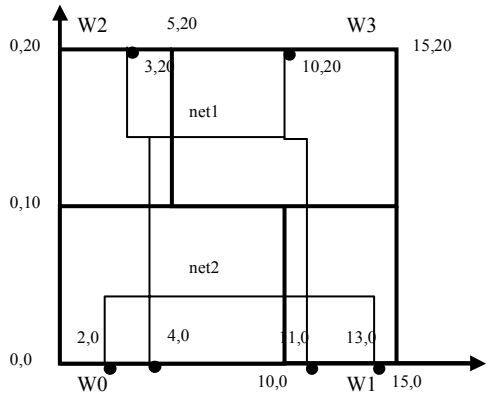

Figure 1. An example floorplan with 2-nets.

These equations can be compactly represented using BDDs. Figure 2 shows the BDD representing the ON-set of Eqn. (2). The dashed lines represent the 0-arcs and 1-arcs are represented with solid lines. Each node is tagged with the variable on which shanon-decomposition [Bryant 86] is performed. The OBDD for Eqn. 2 has 16 nodes. The OBDD for Eqn. 1 has 24 nodes. For reasonably large floorplans maintaining full BDDs simultaneously for each net is not feasible due to memory constraints. Figures 3 and 4 show the POPDDs exploring different regions of the ON-set of Eqn. (2) which maintain a bound of 10 and 8 on the number of nodes respectively. Similarly POPDDs can be constructed for net1 (Eqn. 1). Only one POPDD for each net is active in memory at anytime. The POPDDs are constructed in the following manner. First, the set of variables is partitioned into two groups, i.e., block variables and coordinate variables. The block variables are processed before the coordinate variables. The equations representing the flip configurations and the terminal coordinates can be viewed as a combinational network. The combinational network representing Eqn. 2 is shown in Fig. 5. The POPDD is constructed while traversing the network in DFS manner starting from the output. The partial solution space targeted by the POPDD of Fig. 3 is given by:

$$(!w0)(x3)(!x2)(!x1)(!x0) + (w1)(x3)(x2)(!x1)(x0) + (!w1)(x3)(x2)(!x1)(!x0) \quad (3)$$

While building the POPDD, the AND nodes in the combinational network as shown in Fig. 5 are visited in the following order: <M2,M4,M3,M1>. Note that Eqn. (3) does not contain all minterms of the entire solution space corresponding to w0=1. It only contains a partial set of minterms corresponding to w0=1. Due to this, the $HPWL_X$ corresponding to any configuration having w0=1 can not be computed from the POPDD of Fig. 3. All the paths corresponding to w0=1 is terminated at the unknown (U) terminus. This allows further saving on the memory and reduces computation time. Eqn. (3) fully represents the ON-set (solution space) corresponding to w0=0, w1=0 and w0=0, w1=1. Hence the $HPWL_X$ corresponding to the configurations (!w0)(!w1) and (!w0)(w1) can be computed from the POPDD of Fig. 3.

The partial solution space targeted by the POPDD of Fig. 4 is given by:

$$(w0)(!x3)(!x2)(x1)(!x0)+(!w1)(x3)(x2)(!x1)(!x0) \quad (4)$$

Using similar reasoning as the above, it can be seen that the $HPWL_X$ corresponding to w0=0 and w1=1 (i.e., (!w0)(!w1), (!w0)(w1) and (w0)(w1)) can not be computed from the POPDD of Fig. 4 and the corresponding paths are directed to the unknown (U) terminus. Only the $HPWL_X$ corresponding to the configuration (w0)(!w1) can be computed from the POPDD of Fig. 4. Thus the input ordering in this case is <M1,M4,M3,M2>. Based on the order in which the inputs to the OR gate (combinational network representation) are processed and the bound on the number of nodes, different regions of the ON-set of the equation can be explored. Ordering should be such that the overlap between the explored solution spaces in different iterations is minimized. In both cases the base variable ordering is <w0,w1,w2,w3 | x3,x2,x1,x0>. This helps in maximizing sharing of OPDD nodes. A different input ordering is used for each of the iterations. The POPDDs built this way represent a partial set of flip configurations and the HPWL of the floorplan can be computed for those partial set of flip configurations in each iteration.

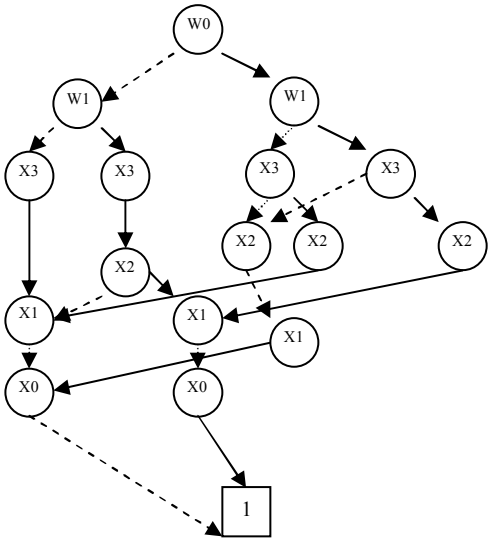

Figure 2. Zero-suppressed OBDD for Eqn. (2).

A. Computing HPWL from POPDD

Note that the same input-ordering (for OR gate) and base ordering have to be used for every net in any one iteration. This ensures that in any one iteration, the number of compatible flip-configurations across all nets is maximized. However these orderings will vary from one iteration to the next. This is required to explore hitherto unexplored regions of the solution space (i.e., flip conditions). While building the POPDD, whenever the bound on the number of nodes is reached, all the remaining paths are directed to the unknown (U) terminus. Paths terminating at the U-terminus represent the unexplored solution space.

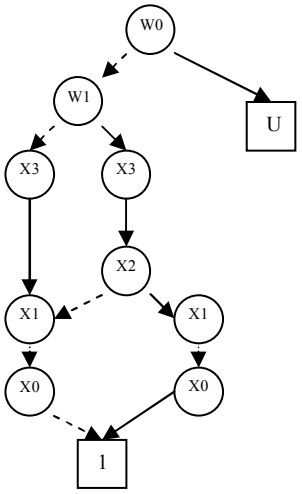

Figure 3. Zero-suppressed POPDD of Eqn. 2 (node limit 10: input ordering for the OR gate<2,4,3,1>)

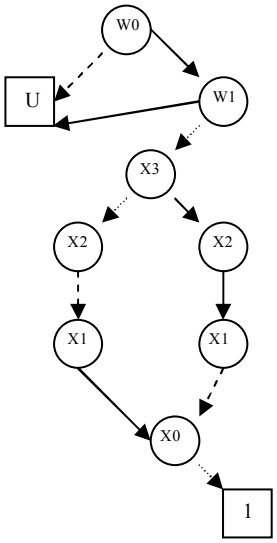

Figure 4. Zero-suppressed POPDD of Eqn 2 (node limit 8: input ordering for the OR gate<1,4,3,2>)

The HPWLX of the floorplan can be calculated after each iteration by adding up the HPWLX of the individual nets corresponding to the compatible configurations. Each path from from the root to the 1-terminus of the POPDD represents a cube of the ON-set of the corresponding function. The cubes can be extracted in parallel while traversing the POPDD. If there are m block variables and k coordinate variables then there are m+k entries in each cube. Absence of a variable is denoted by a x in that position. All the paths to the 1-terminus represent the explored portion of the ON-set as a set of disjoint cubes. Table 1 and Table 2 shows the cubes encoded by the paths to the 1-terminus and corresponding configurations, coordinates and HPWLX (maxX – minX) computed from the coordinates for each of the configurations for 2 iterations (Eqn. 2). Finally the HPWLX of every net corresponding to the compatible configurations are added to obtain the HPWLX for the floorplan. Note that even though in the worst case the number of the distinct configurations is exponential in the number of the blocks, due to a compact POPDD based representation the actual number of distinct configurations to be considered is typically much less.

Further the reduction in the number of distinct configurations per net is achieved by merging configurations yielding same $HPWL_X$ for the net. Moreover configurations resulting in poor cost can be thrown away from further consideration. While adding up the $HPWL_X$ of the nets corresponding to a configuration if the sum exceeds previously found best $HPWL_X$ (minimum) for the floorplan then the configuration can be immediately removed from consideration. In this example, for iteration 1, only 2 configurations had to be considered (000X, 001X) out of possible 16 configurations. The same holds for iteration 2. As can be seen, in this case optimal solution can be achieved after 2 iterations. The peak memory requirement is 20 nodes. For the BDD based approach the peak memory requirement would be 40 nodes. The quality of the solutions obtained depends on the input ordering, the bound on the number of the POPDD nodes and the number of iterations. From each of the iterations only the configurations yielding the minimum $HPWL_X$ is retained. Performing addition at lower resolution (in terms of the number of coordinate bits being used) can be used to prune the unpromising solutions in the first-place. Next addition can be performed at a higher resolution using a larger number of coordinate variables. The final solution is the minimum across all the iterations. The bound on the number of the nodes can also be varied across iterations. The number of iterations can be increased at some cost of runtime to explore other configurations. Note that by bounding the maximum number of nodes per net, the total number of nodes required to represent the HPWL for the entire floorplan is also bounded. The POPDD nodes having same functional representation are shared across the different OPDDs corresponding to the different nets. So the actual number of OPDD nodes required for all the nets is much less than (max number of nodes per net × #nets).

In our proposed approach, sub-optimality is introduced because configurations across iterations can not be considered together. However as our experimental results suggest, by maximizing compatible configurations among nets per iteration and by exploring the unexplored solution space in successive iterations, very high quality solution in terms of minimized wirelength can be obtained while maintaining a bound on the memory requirement and fast runtime.

Table 1. Iteration 1

Net	Cubes	Config	Coordinate (int)	$HPWL_X$ max-min	CONFIG ($\Sigma HPWL_X$)
1	000XXX10	000X	XX10=> 2,6,10,14	(14-2)=12	CONFIG= 001X,00XX =>001X ($\Sigma HPWL_X$)= (11+4=15)
	001XX110 001X0011 001X1010	001X	X110=>6,14 0011=>3 1010->10	(14-3)=11	
2	00XX1X00	00XX	1X00=>8,12	(12-8)=4	

Table 2. Iteration 2

Net	Cubes	Config	Coordinate (int)	$HPWL_X$ max-min	CONFIG ($\Sigma HPWL_X$)
1	010X101X 010X0010 010X0110	010X	101X=> 10,11 0010=>2 0110=>6	(11-2)=9	CONFIG= 011X,01XX =>011X ($\Sigma HPWL_X$)= (8+5=13)
	011X101X 011X0011 011X0110	011X	101X=> 10,11 0011=>3 0110=>6	(11-3)=8	
2	01XX1000 01XX1101	01XX	1000=8 1101=13	(13-8)=5	

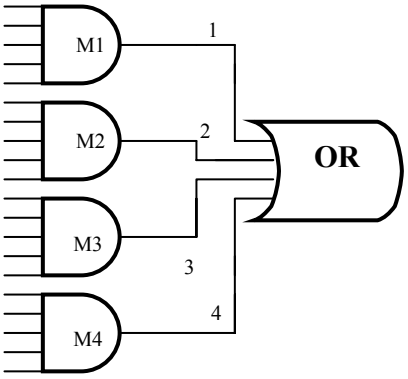

Figure 5. Combinational Network Representation of Eq.2

In [Hao 05] several strategies were presented to reduce the sizes of the BDDs. All those strategies can be applied without modification to the POPDDs. For the larger floorplans, the algorithm in [Hao 05] can be applied by limiting the number of blocks. A block weighting stategy was proposed to choose a set of flipping blocks. In our proposed algorithm we don't limit the number of flipping blocks rather we explore different set of flip configurations iteratively in different iterations. Experimental results show that our proposed agorithm consistently achieves better result when compared with the block limiting approach of [Hao 05] for large floorplans.

III. EXPERIMENTAL RESULTS

We performed two sets of experiments to demonstrate the efficiency of our proposed approach: one with randomly generated large floorplans and the other one with some of the MCNC benchmarks. We compare our results with another boolean symbolic approach [Hao 05] based on the BDDs. In all the experiments, the maximum number of OPDD nodes per net was kept at 500. Table 3 shows the floorplan information for the randomly generated floorplans.

Table 3. Randomly generated floorplans

	Block#	**Net#**	**Pin#**	**pins/net**
Ckt1	10	100	300	2-4
Ckt2	20	200	1000	2-10
Ckt3	50	1000	3000	2-4
Ckt4	50	1000	5000	2-10
Ckt5	100	500	1500	2-4
Ckt6	300	1200	6000	2-10

Table 4. Comparison (Proposed:#iter=3)

	Hao[05]			Proposed		
	Run Time (s)	#bdd nodes	$hpwl_x$	Run Time (s)	#opdd nodes	$hpwl_x$
Ckt1	3	5040	2450	3.2	2520	2450
Ckt2	7	26100	7100	8	17500	7300
Ckt3	13	33500	35000	12.2	24900	34000
Ckt4	15	53500	42000	14.8	43500	42000
Ckt5	21	25450	17500	24.1	24900	17000
Ckt6	228	170400	44980	194	156600	42793

Table 4 compares the [Hao 05] with our proposed approach for the randomly generated floorplans. Wirelength minimization based on in place flipping is targeted. For [Hao 05] the maximum number of flipping blocks is limited to 25 (due to memory constraint). For ckt1 and ckt2 optimum wirelength can be obtained using [Hao 05]. In these cases, close to optimum solution can be obtained using the proposed approach while using much less memory and a comparable runtime. The 5th column of Table 4 shows the total runtime for 3 iterations for the proposed approach. Figure 6 shows accuracy vs no of POPDD nodes (memory) for ckt2. It also shows how the accuracy is affected by the number of the iterations. The x-axis plots ((obtained (proposed) − optimum) / optimum)*100% and y-axis plots the maximum number of POPDD nodes for any iteration. For larger floorplans, consistently better solutions can be obtained using the proposed method both in terms of reduction in the wirelength and the memory (#nodes).

The details of the MCNC benchmarks are shown in Table 5. Table 6 compares the result with that of [Hao 05]. Same resolution is used for all the experiments. 1st column of Table 6 reports ((runtime per iteration (proposed) − runtime (Hao 05)) / runtime (Hao 05))×100%. 2nd column of Table 6 reports ((memory (proposed) − memory (Hao 05))/ memory (Hao 05))×100%. 3rd column of Table 6 reports (($HPWL_X$ (proposed) − $HPWL_X$ (Hao 05)) / $HPWL_X$ (Hao 05))×100%. For the smaller floorplans an optimal solution is obtained using [Hao 05]. In these cases, the proposed algorithm can obtain a near optimal solution using only a fraction of the memory required by [Hao 05] and the runtime is comparable. For the larger floorplans the proposed algorithm outperforms the BDD based block limiting (#max flippable blocks= 20) approach both in terms of the reduction in wirelength and the memory requirement.

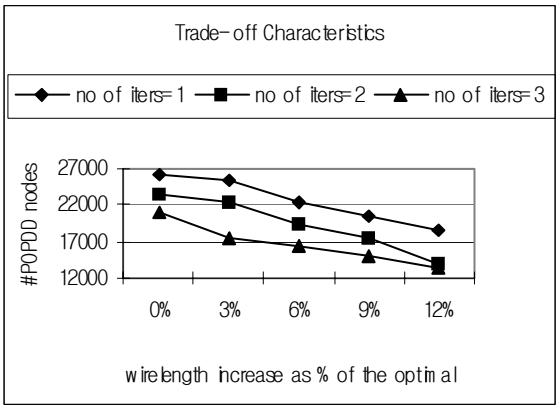

Figure 6. Trade-off characteristics (ckt2)

Table 5. MCNC Benchmarks

	Block#	Net#	Pin#	Pad#
Apte	9	97	214	73
Xerox	10	203	696	2
Hp	11	83	264	45
Ami33	33	123	480	42
Ami49	49	408	931	22

Table 6. Comparison with [Hao 05] (Proposed:#iter=3, resolution = 1000)

	Δ(runtime)	Δ(memory)	Δ($HPWL_X$)
Apte	-7.5%	-45%	+4.2%
Xerox	-8%	-40%	+5.0%
Hp	-12%	-50%	0.0%
Ami33	-20%	-30%	-2.0%
Ami49	-40%	-40%	-12.0%

IV. CONCLUSION

A compact partial functional representation based algorithm is presented for wirelength minimization with in-place cell/macro flipping in large-scale system-on-chip floorplans. The proposed algorithm provides smooth tradeoff between the accuracy and the memory requirement and consistently outperforms the full Boolean symbolic representation based methods both in terms of reduction in wirelength and memory requirement for floorplans with a large number of blocks.

REFERENCES

[Bryant 86] Bryant, R. E. "Graph Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, Vol c-35, No. 8, August 1986, pp. 677-691.

[Cheng 91] Cheng, C.K., S.Z. Yao, and T.C. Hu, "The orientation of modules based on graph decompositions," *IEEE Transactions on Computers*, 40(6):774.780, Jun 1991.

[Chong 93] Chong, K., and S. Sahni, "Minimizing total wire length by flipping modules," *Computer-Aided Design of Integrated Circuits and Systems,IEEE Transactions on*, 12(1):167.175, Jan 1993.

[Funabiki 98] Funabiki. N., J. Kitamichi, and S. Nishikawa,"An evolutionary neural network approach for module orientationproblems,"*IEEE Transactions on Systems, Man and Cybernetics*, 28(6):pp. 849-855, Dec 1998.

[Hao 05] Hao, Xin, and Forrest Brewer, "Wirelength Optimization by Optimal Block Placement," *ICCAD*, 2005.

[Jeong 91] Jeong, J.C. and C.-M. Kyung,"Finding optimal module orientation in macro cell placement circuits and systems,"In *Circuits and Systems, ISCAS. IEEE International Symposium on*, pages 3118-3121 vol.5, 1991.

[Kim 91] Kim, S. and C.M. Kyung,"Module orientation algorithm using reconstruction of nets and meanfield annealing," *Electronics Letters*, 27(13):1198-1200, Jun 1991.

[Murata 96] Murata, H., K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Vlsi module placement based on rectangle-packing by the sequence-pair," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 15(12):1518-1524, Dec 1996.

[Ross 90] Ross, D. E., "Functional Calculation Using Ordered Partial Multi Decision Diagrams," *Ph. D. dissertation*. University of Texas, Austin, August 1990.

[Shahookar 91] Shahookar, K. and P. Mazumder, "VLSI cell placement techniques," *ACM Computing Surveys (CSUR) archive*, 23(2):143-220, Jun 1991.

[Yamada 88] Yamada, M., and C. L. Liu, "An analytical method for optimal module orientation," In *Circuits and Systems, ISCAS. IEEE International Symposium on*, pages 1679-1682, vol.2, 1988.

[Yan 93] Yan, J.-T., and P.-Y. Hsiao, "The module orientation problem based on Manhattan wire measure is still NP-complete", In *Circuits and Systems, MWSCAS. The Midwest Symposium on*, pages 526-529 vol.1, 1993.

[Yan 95] Yan, J.-T., and P.-Y. Hsiao, "Orientation assignment of standard cells using a fuzzy mathematical transformation," *Proc. of Computers and Digital Techniques*, 142(2):157.164, Mar 1995.

[Yan 96] Yan, Jin-Tai, "A simple yet effective genetic approach for the orientation assignment on cell-based layout," In *International Conference on VLSI Design: VLSI in Mobile Communication*, pages 33.36, 1996.

[Yao 90] Yao, Xianjin, and C. L. Liu, "Solution of a module orientation and rotation problem," In *Design Automation Conference, EDAC. Proceedings of the European*, pages 594 -588, 1990.