1-1-2003

Non-Crossing Ordered BDD for Physical Synthesis of Regular Circuit Structure

Aiqun Cao

Cheng-Kok Koh

Non-Crossing Ordered BDD for Physical Synthesis of Regular Circuit Structure*

Aiqun Cao and Cheng-Kok Koh
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University West Lafayette, IN 47907-1285
{caoa,chengkok}@ecn.purdue.edu

March 11, 2003

*This research is supported in part by NSF (CAREER Award CCR-9984553).

Contents

1 Introduction **1**

2 Non-crossing OBDD (NCOBDD) **2**

2.1 Crossing minimization in graph drawings 3

2.2 NCOBDD construction 4

2.2.1 Top-down level-by-level sweep 4

2.2.2 Back-tracing level sweep 5

3 Physical mapping and wave steering **6**

4 Experimental results **7**

List of Tables

1 Experiment results. 8

List of Figures

1 NCOBDD vs. YADD. 2
2 Insertion of dummy nodes. 3
3 The structure between 2 levels of NCOBDD. 4
4 Reordering and duplication operation. 4
5 A ROBDD representation. 5
6 Two NCOBDD representations. 5
7 Nodes collapsing. 7
8 Wave steering on YADD (adapted from [14]). 8

Abstract

In this paper, we propose a novel compact BDD structure, called Non-crossing ordered BDD (NCOBDD), that can be mapped directly to a regular circuit structure. Compared with other BDD-based regular structures, NCOBDD-mapped circuits reduce the costs of area, power and latency, while preserving the regularity of the structure. We also present an algorithm that uses a top-down level-by-level sweep and a back-tracing mechanism to construct the minimum sized NCOBDD. Experimental results show that for asymmetric benchmark circuits, the average reduction on area, power and latency are 57.8%, 49.9% and 68.7%, respectively, compared with Yet Another Decision Diagram (YADD) [14].

1 Introduction

Interconnects play an important role in determining the performance, reliability, and robustness in today's VLSI circuits and will continue to do so in the future. Many new design paradigms advocate the incorporation of interconnect effects at every level of the design process. However, the analysis and prediction of interconnect effects are difficult in the early design stages. In particular, the irregularity of global interconnects renders the prediction of delay and crosstalk difficult. In Ref. [8], it was argued that regular circuit structures could overcome the timing closure problem and offset the process variation, and would scale better with technology. In particular, regular design structures are desirable because they eliminate irregular global interconnects. Two main categories of regular structures have been proposed in the recent years: Programmable Logic Array (PLA)-based and Binary Decision Diagram (BDD)-based.

Single-PLA is a regular structure that has the advantage that the implementation of a design does not require technology mapping, placement, and routing. In order to accommodate more complex logic, a network of PLAs (NPLA) can be constructed with placement and routing [7]. However, that compromises the global regularity of the structure. In Ref. [9], a multiple-PLA structure called River PLA (RPLA) and its reconfigurable version, Glacier PLA, are proposed. The RPLA is a structure with stacked PLAs. The interconnections among stacked PLAs are ordered and realized via river routing, which is local and regular. Thus, RPLA preserves both global and local regularity. Whirlpool PLA (WPLA) presented in Ref. [10] is a cyclic four-level Boolean NOR network. It is superior than RPLA in terms of area and delay, but can implement only smaller logic.

The main idea behind BDD-based regular structure is to represent a given Boolean function with a decision diagram that has a regular structure. In Refs. [4, 3], a pseudo-symmetric BDD (PSBDD) is presented for FPGA synthesis. Yet Another Decision Diagram (YADD), proposed in Refs. [14, 13, 12], can be directly mapped to a network of multiplexers. In both PSBDD and YADD, the number of nodes of each level grows linearly with the number of levels at the cost of some variables appearing on several levels during the application of Shannon's expansion. YADD is more general than PSBDD in the sense that it does not restrict the ordering of child nodes of a parent. In the mapping of FPGA (from PSBDD) and network of multiplexers (from YADD), only predictable local interconnects are needed.

Although PSBDD and YADD are fairly compact representations for symmetric boolean function, repetition of variables in several levels is typically needed for non-symmetric functions. In other words, they trade off the height, and thus area and latency, of the BDDs to achieve structural regularity. As reported in [14] for example, the ROBDD for one of the primary outputs of the benchmark circuit *'alu2'* with 10 primary inputs has 10 levels only, whereas the number of levels for the corresponding YADD has 33 levels.

Between the PLA-based and BDD-based structures, PLA-based structures have similar area and timing performance as standard-cell designs [10, 9]. The YADD structures (with wave-steering) have far superior timing performance when compared to standard-cell designs [14, 13]. However, YADD structures have high area and power penalties due to the repetition of variables in multiple levels. The objective of this work is to achieve similar timing performance as YADD structures as well as far superior area and power performance.

In this paper, we propose a more compact BDD structure that can also be directly mapped to a regular circuit

structure. The regularity of the proposed BDD structure is achieved by: 1) restricting all connections to be between adjacent levels, 2) enforcing no crossings between connections of the BDD nodes [1]. Similar to the ordered BDD (OBDD), the variables are ordered and one variable appears in only one level. We refer to this regular structure as Non-crossing OBDD (NCOBDD).

In an NCOBDD, the number of nodes of each level may grow more than linearly with the number of levels, but the total number of levels is typically smaller than that required by PSBDD or YADD. Figure 1 is a simple example that illustrate that. In Figure 1 (and subsequent figures), the dash line from node x (function f) connects to the low child (cofactor $f_{x=0}$) and the solid line to the high child (cofactor $f_{x=1}$). It is also evident from Figure 1 that mapping an NCOBDD structure to a network of multiplexers is as straightforward as YADD. Although the interconnects in different levels may have different parameters, they can be accurately predicted as a consequence of the structured placement of the cells.

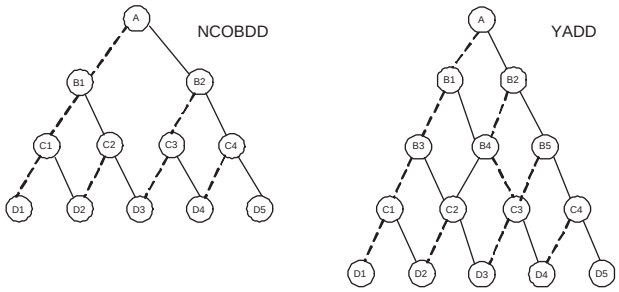

Figure 1: NCOBDD vs. YADD.

NCOBDD preserves the performance and the structural regularity, whereas has less area, power and latency than YADD. Experimental results show that when compared with YADD, NCOBDD reduces the area by 20.4% and power by 26.4% averagely for partially symmetric circuits; for asymmetric circuits, the reductions on area, power and latency in average are 57.8%, 49.9% and 68.7%, respectively. NCOBDDs even have better PDPs (product of delay and power) than standard-cell designs for bigger asymmetric circuits.

2 Non-crossing OBDD (NCOBDD)

An NCOBDD has the following attributes:

1. It is an ordered Binary Decision Diagram (OBDD);
2. Every variable characterizes only one level of the OBDD;
3. All connections are between two adjacent levels;
4. There are no crossings between connections.

[1] Strictly speaking, it is not necessary to impose the no crossings restriction as long as the interconnect parasitics can be accurately predicted.

Of particular interest is the construction of the minimum NCOBDD, the NCOBDD with minimum number of nodes, for a given logic function. However, the construction of an NCOBDD from a logic function is more complicated due to attributes (3) and (4). These two attributes imply that the child nodes of a parent node must be adjacent in the next level. In this paper, we take the approach of constructing an NCOBDD from an ROBDD of the given logic function. The key is to reorder and duplicate appropriate nodes in the ROBDD structure. In the ROBDD structure, however, the connections may not be restricted to between two adjacent levels. To overcome this problem, dummy nodes can be inserted, as shown in Figure 2.

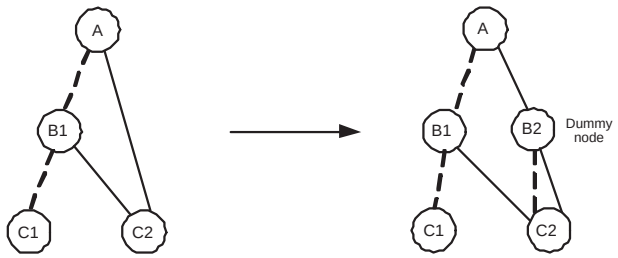

Figure 2: Insertion of dummy nodes.

2.1 Crossing minimization in graph drawings

A related problem to NCOBDD construction is that of crossing minimization in the drawings of leveled directed graphs, i.e., creating polyline drawings of directed graphs with nodes arranged in horizontal levels. For a directed graph with k levels of nodes, the problem of k-level crossing minimization is that of finding the ordering of the nodes in each level such that the number of edge (straightline) crossings is minimized [1, 5].

Unfortunately, the problem of seeking an ordering for each level in order to minimize the crossings in a leveled directed graph is NP-hard, even for 2-level directed graphs [6]. Moreover, the 2-level crossing minimization problem with a fixed node ordering in one level is still NP-hard [5]. Nonetheless, the solution to this problem is the key to solving the general k-level problem. Several methods have been developed to solve the 2-level crossing minimization with a fixed node ordering in one level [1, 5].

For the k-level crossing minimization problem, a level-by-level sweep technique is used to decompose the k-level problem into a series of $(k-1)$ 2-level problem (with a fixed node ordering in one level). First, a vertex ordering of the top level L_1 is determined. Then, for $i = 2,3,\cdots,k$, the vertex ordering of level L_{i-1} is fixed while the nodes in level L_i are permuted to reduce the number of crossings between edges whose endpoints are in level L_{i-1} and level L_i. In this approach, the level-by-level sweep is performed in a top-down fashion. A bottom-up level-by-level sweep approach, starting from the bottom level L_k, is feasible, too.

2.2 NCOBDD construction

By treating the ROBDD as a directed graph of k levels (k is the number of variables, and L_k refers to the bottom level), we refer to the problem of constructing a non-crossing version of it as the k-level non-crossing problem. A straightforward solution to the k-level non-crossing problem is to first apply the method for k-level crossing minimization to the given ROBDD. Then, the remaining crossings can be eliminated by duplicating appropriate nodes. However, such an approach has two shortcomings. First, duplication cost minimization is not congruent with crossing minimization. In other words, minimizing the crossing does not guarantee minimization of the number of duplicated nodes. Second, it does not consider the structure of BDD nodes, i.e., the out-degree of each BDD node (except two terminal nodes) is 2. Figure 3 shows the structure between two levels L_{i-1} and L_i of an NCOBDD. It consists of several "zig-zag" structures.

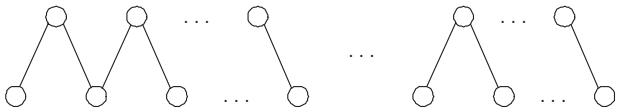

Figure 3: The structure between 2 levels of NCOBDD.

Similar to the k-level crossing minimization presented in Section 2.1, we also apply a level-by-level sweep approach to decompose the k-level non-crossing problem into a series of $(k-1)$ 2-level non-crossing problem with the node ordering in one level fixed. The difference is that besides the reordering of nodes, we also consider the duplication of appropriate nodes in order to eliminate all crossings. Another difference is that for NCOBDD construction, only top-down level-by-level sweep is possible. If bottom-up level-by-level sweep is applied, the duplication of a node x in the upper level L_{i-1} will also lead to the duplication of its child nodes in the lower level L_i in order to avoid crossings. In other words, it is impossible to fix the node ordering in the lower level while trying to determine the ordering and duplication in the upper level.

2.2.1 Top-down level-by-level sweep

Assuming that the ordering and the number of nodes in the current upper level L_{i-1} are fixed, we perform a left-to-right sweep of the ordered nodes $\{V_1, \cdots, ..., V_n\}$ in level L_{i-1} to determine the ordering and, if necessary, duplication of their child nodes in the current lower level L_i.

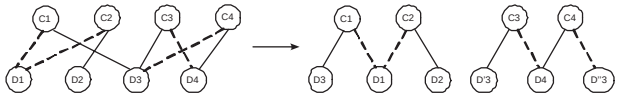

Figure 4: Reordering and duplication operation.

Consider two adjacent nodes V_{j-1} and V_j in level L_{i-1}. If the position of a child node of V_{j-1} (V_j) has already been determined because it is also a child node of V_h, $h < j-1$, and its connection to V_{j-1} (V_j) causes crossings, the duplication of this child node is necessary (see child node D_3 of C_3 in Figure 4). The duplicate replaces the original

child node of V_{j-1} (V_j). Then, there are three possible scenarios when we consider the relation of the child nodes of V_{j-1} and V_j.

(1) If V_{j-1} and V_j have exactly one common child node in level L_i, the result of reordering should have the common child node in the middle, the two remaining child nodes of V_{j-1} and V_j should reside on the left and right of the common child node, respectively (see nodes C_1 and C_2 in Figure 4). The next two nodes under consideration in level L_{i-1} are V_{j+1} and V_{j+2}.

(2) If V_{j-1} and V_j have no common child nodes, the order between two child nodes of V_{j-1} can be arbitrary (see nodes B_1 and B_2 in Figure 5). We will discuss in the next subsection how the arbitrariness can be exploited to reduce the duplication cost of the descendant nodes in the lower levels. The next two nodes to consider are V_j and V_{j+1}.

(3) If V_{j-1} and V_j have two common child nodes, duplication is required. However, any of the two child nodes can be duplicated (see nodes C_3 and C_4 in Figure 4). Again, the arbitrariness can be exploited for the reduction of duplication cost in subsequent levels (see Section 2.2.2). Nodes V_{j+1} and V_{j+2} are considered next in the left-to-right sweep.

2.2.2 Back-tracing level sweep

In scenarios (2) and (3) described in section 2.2.1, the ordering and duplication of the child nodes can be arbitrary. While it may not affect the number of nodes in the current lower level (L_i) under consideration, the ordering will affect the results of reordering and duplication operations on subsequent levels. In Figure 5, we show the original ROBDD structure. Two different NCOBDDs due to different orderings of C_1 and C_2, and C_3 and C_4 are shown in Figure 6. The right one has fewer nodes than the left one.

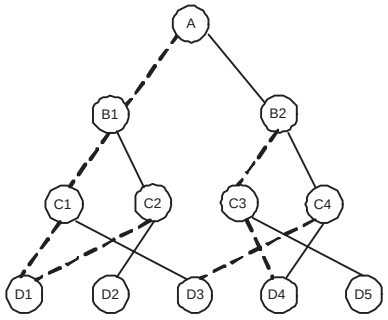

Figure 5: A ROBDD representation.

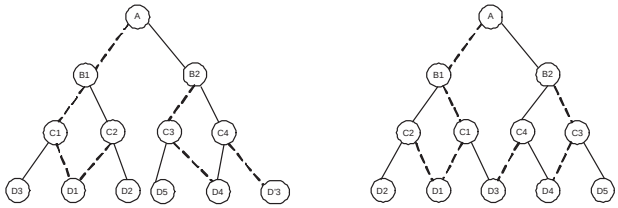

Figure 6: Two NCOBDD representations.

Therefore, the question is how we can exploit the arbitrariness introduced at the upper levels during the level-by-level sweep operations on lower levels. Due to page limitation, we present only the details of the approach that we use to exploit the arbitrariness introduced in scenario (2). The arbitrariness introduced in scenario (3) can be handled with a simple extension.

We propose a back-tracing mechanism as follows: For adjacent nodes, if the order between them can be arbitrary, we collapse them into one super node. For example, Figure 7 shows the collapsing of BDD nodes in Figure 5.

In Figure 7, the collapsed nodes C_{12} and C_{34} are considered when we try to eliminate the crossings between levels C and D. As node D_3 is the only common child node of C_{12} and C_{34}, it should be placed in the middle, i.e., it should be the last node among all the child nodes of C_{12} and the first node among those of C_{34}. Given the position of D_3, the order of the nodes within the collapsed nodes, i.e., C_1 and C_2, and C_3 and C_4 can be determined accordingly; We refer to that as de-collapsing. Subsequently, the ordering of other child nodes of C_1 and C_2, and C_3 and C_4 are considered, respectively.

The incorporation of node collapsing during the level-by-level sweep operation implies that the order of the nodes in the current upper level under consideration (L_{i-1}) is no longer completely fixed. In general, there may be multiple levels of collapsed nodes (see node B_{12} for example in Figure 7). This provides even more flexibility to the node ordering in level L_{i-1} and above; not only is the ordering *within* the collapsed nodes in level L_{i-1} not fixed, but the ordering *among* those collapsed nodes can be changed, too.

For that reason, we first perform a branch and bound algorithm to search for an ordering *among* the collapsed nodes in level L_{i-1} that minimizes the duplication of nodes in the current lower level L_i. (Two different possible orderings between collapsed nodes C_{12} and C_{34} lead to the same optimal number of nodes when we try to minimize the duplication cost of level D in Figure 7). Collapsed nodes in higher levels L_j, $j < i-1$, are de-collapsed after this step. However, if the duplication cost is independent of the de-collapsing of a super node, the node will remain collapsed for future consideration.

After the ordering among the collapsed nodes in level L_{i-1} is obtained, the ordering *within* each of those collapsed nodes is determined with the same goal of minimizing the number of nodes in L_i. Some collapsed nodes in level L_{i-1} are de-collapsed after this step. The reordering and duplication operations in level L_i are carried out at the same time. Nodes collapsing in level L_i is also performed.

Although such an back-tracing mechanism in general has exponential time complexity, the number of possible orders to be enumerated using the branch-and-bound algorithm in level L_{i-1} in this context is very small as the collapsed nodes are usually de-collapsed after a few levels. Therefore, the computational overhead is acceptable, as shown by the run-times reported in Section 4.

3 Physical mapping and wave steering

The mapping of NCOBDD to a physical layout is similar to the physical mapping of YADD. In this section, we review the physical mapping of YADD and the wave steering technique applied to YADD [14] briefly.

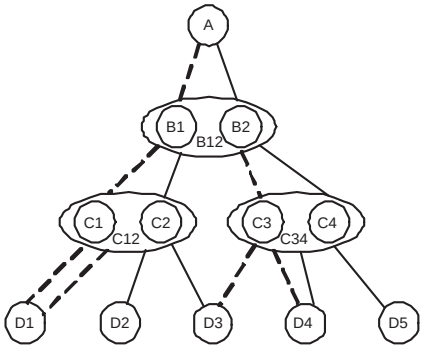

Figure 7: Nodes collapsing.

Mapping BDD structure to pass transistor logic is presented in Ref. [2]. Each node in a BDD structure is an ITE (if-then-else) node, which makes an decision based on the value of the variable characterizing the level. It is equivalent to a 2-to-1 multiplexer. The 2 data inputs correspond to the two child nodes and the controlling input corresponds to the variable characterizing the level. All the variants of BDD structure, including PSBDD, YADD, and NCOBDD in this paper, can be mapped directly to multiplexer-based networks.

For YADD and NCOBDD, the placement of all the multiplexer units can be easily determined. Moreover, there are no crossings among the interconnects at all and all the signal interconnects are locally abutted between adjacent units. Therefore, the delay of the interconnects can be computed accurately and the crosstalk easily controlled.

In Ref. [14], wave steering (wave pipelining) is applied to the YADD structure. The function synthesized by the YADD is evaluated using a two-phase clocking scheme in a bottom-up fashion. Each level alternates between two modes: “hold” and “evaluate”. While level L_{i-1} evaluates, level L_i holds. The results of level L_{i-1} are then fed to level L_{i-2}, which will evaluate in the next clock phase while L_{i-1} holds. For the correct operation, the input variable that characterizes the level L_i holds when L_i is evaluating. It is allowed to change its value when L_i is holding. This is called wave steering, i.e., there are several computing waves propagating through the circuit at the same time. For wave steering to work properly without any data corruption or race, the inputs to YADD structure have to be applied with appropriate phase difference. Figure 8 shows the floorplan of YADD and inputs configuration of wave steering. In Figure 8, each input is first delayed by a suitable number of flip-flops (FFs) with an appropriate clock phase, and then driven by a properly sized driver (Dr). $CLK1$ and $CLK2$ are 2-phase non-overlapping complementary clocks. Obviously, the wave steering technique can also be applied to NCOBDD.

4 Experimental results

We evaluate the area, power, and timing performance of the NCOBDD structure using nine MCNC benchmark circuits. For each circuit, the ROBDD representation is first obtained using the *’BuDDy’* BDD package. We use the dynamic variable reordering feature of ’BuDDy’ to determine the variable ordering of ROBDD (and NCOBDD). Note that minimized ROBDD does not imply minimized NCOBDD.

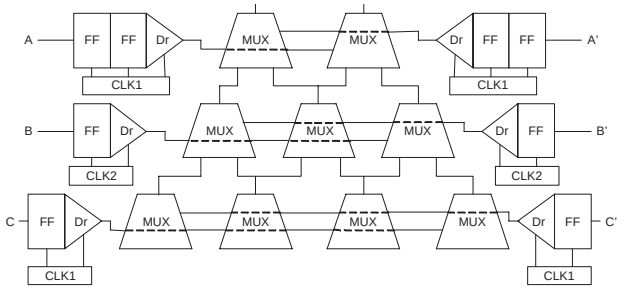

Figure 8: Wave steering on YADD (adapted from [14]).

circuit		xor5	xor7	9sym	cm138a	z4ml	rd53	rd73	alu2	alu4
primary inputs		5	7	9	6	7	5	7	10	14
primary outputs		1	1	1	8	4	3	3	6	8
NCOBDD	levels	5	7	9	6	7	5	7	10	14
	nodes	15	28	33	48	77	35	68	465	3482
	run time(s)	1	2	4	3	3	2	3	8	18
	area(x1000μm^2)	1.532	2.947	4.102	6.682	7.848	3.882	7.603	35.023	210.070
	latency(ns)	1.5	2.1	2.7	1.8	2.1	1.5	2.1	3	4.2
	power(mW)	0.294	0.499	1.322	0.925	0.912	0.640	1.229	5.635	12.245
	PDP(ns x mW)	0.176	0.300	0.793	0.555	0.547	0.384	0.737	3.38	7.347
YADD	levels	5	7	9	6	7	5	7	33	43
	nodes	15	28	45	168	77	45	84	1369	6264
	area(x1000μm^2)	1.532	2.947	4.869	12.850	7.848	4.253	8.354	89.021	465.760
	latency(ns)	1.5	2.1	2.7	1.8	2.1	1.5	2.1	9.9	12.9
	power(mW)	0.294	0.499	1.534	1.601	0.912	0.792	1.761	9.749	28.802
	PDP(ns x mW)	0.176	0.300	0.920	0.961	0.547	0.475	1.233	5.849	17.281
Standard-Cell	area(x1000μm^2)	0.627	1.132	2.544	0.996	1.988	0.947	1.584	9.975	48.363
	delay(ns)	0.645	0.678	1.086	0.598	0.862	0.797	0.934	3.36	5.772
	power(mW)	0.109	0.298	0.577	0.346	0.389	0.192	0.495	2.698	5.866
	PDP(ns x mW)	0.070	0.202	0.627	0.207	0.335	0.153	0.462	9.065	33.859

Table 1: Experiment results.

For each benchmark circuit, the NCOBDD is mapped to a network of multiplexers and wave steering is applied as described in Section 3. The multiplexer can be implemented by pass transistors. Two pass n-FET transistors are used, where each cell requires the variable and its complement as the controlling inputs to the two pass transistors. To make up for the voltage loss due to NMOS-only pass transistor logic, each cell is followed by a level-restoring inverter. A 2-phase non-overlapping clocking scheme is applied. The drivers and flip-flops are carefully designed so that wave steering can work properly.

The benchmark circuits are implemented with 0.25μm technology. Layout extraction and simulation are done to obtain the power and timing. The delay of each level is less than 0.3ns, making 1.66GHz (a clock cycle of 0.6ns) the highest achievable clock frequency with which each circuit can work properly. Table 1 shows the (maximum) number of levels and the (total) number of nodes for the NCOBDDs constructed for each benchmark circuit. It also reports the area, the (longest) latency, and the power dissipation of each physically mapped circuit.

For comparison, we also synthesize the nine benchmark circuits with YADD. As the source codes for YADD construction are not available, we reimplement the algorithm described in Ref. [14]. For each circuit, the variable ordering for YADD is the same as that of NCOBDD. The same layout mapping is applied for YADD using the same technology. The same clock frequency 1.66GHz can be achieved for YADD mapped circuits. Table 1 also includes the results of circuits mapped from YADDs.

From Table 1, we can see that for symmetric circuits (xor5 and xor7), NCOBDD has the same area, power and latency as YADD; (they are exactly the same circuits). For partially symmetric circuits (9sym, cm138a, rd53, and rd73), NCOBDD reduces the area by 20.4% and power by 26.4%, while maintaining the same latency. For asymmetric circuits (alu2 and alu4), the reductions on area, power and latency are 57.8%, 49.9% and 68.7%, respectively.

We would like to point out that the area of YADD implementation for some benchmark circuits, *'alu2'* and *'alu4'*, reported in Table 1 are significantly larger than those reported in Ref. [14], after taking into account the different technology used. (Ref. [14] used 0.5 μm technology). To the best of our knowledge [11], the area for each of the two circuits reported in Ref. [14] are only for one of the primary outputs of the circuit.

We also implement the nine benchmark circuits with standard-cells. We use Cadence Silicon Ensemble to generate the layout using 0.25 μm standard-cell library. Parasitic extraction is then carried out and simulation performed. The area, timing, and power performance of these standard-cell circuits are also included in Table 1. When compared with standard-cells, the PDPs of NCOBDD-mapped circuits are better for bigger asymmetric circuits. Of course, the standard-cell design is the most area efficient among the three.

References

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[2] V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. De Micheli. Decision diagrams and pass transistor logic synthesis. In *Proc. Int. Workshop on Logic Synthesis*, pages 1–5, May 1997.

[3] M. Chrzanowska-Jeske, X. Y. Ma, and W. Wang. Pseudo-symmetric functional decision diagrams. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 175–178, 1998.

[4] M. Chrzanowska-Jeske, Z. Wang, and Y. Xu. A regular representation for mapping to fine-grain, locally-connected FPGAs. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 2749–2752, 1997.

[5] P. Eades and S. Whitesides. Drawing graphs in two layers. *Theoretical Computer Science*, 131(2):361–374, 1994.

[6] M. Garey and D. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4(3):312–316, 1983.

[7] S. P. Khatri, R. K. Brayton, and A. Sangiovanni-Vincentelli. Cross-talk immune VLSI design using a network of PLAs embedded in a regular layout fabric. In *Proc. Int. Conf. on Computer Aided Design*, pages 412–418, November 2000.

[8] F. Mo and R. K. Brayton. Regular fabrics in deep sub-micron integrated-circuit design. In *Proc. Int. Workshop on Logic Synthesis*, pages 7–12, June 2002.

[9] F. Mo and R. K. Brayton. River PLAs: a regular circuit structure. In *Proc. Design Automation Conf*, pages 201–206, June 2002.

[10] F. Mo and R. K. Brayton. Whirlpool PLAs: a regular logic structure and their synthesis. In *Proc. Int. Conf. on Computer Aided Design*, pages 543–550, November 2002.

[11] A. Mukherjee. Personal communication, September 2002.

[12] A. Mukherjee. *Wave steering as a means for achieving performance and predictability in DSM circuits.* PhD thesis, University of California, Santa Barbara, 2002.

[13] A. Mukherjee, M. Marek-Sadowska, and S. I. Long. Wave pipelining YADDs - a feasibility study. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 559–562, May 1999.

[14] A. Mukherjee, R. Sudhakar, M. Marek-Sadowska, and S. I. Long. Wave steering in YADDs: a novel non-iterative synthesis and layout technique. In *Proc. Design Automation Conf*, pages 466–471, June 1999.