

# Non-Crossing OBDDs for Mapping to Regular Circuit Structures

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

## 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 structures. We also present an algorithm that uses a top-down level-by-level sweep to construct minimal NCOBDDs. Experimental results show that for asymmetric benchmark circuits, the average reduction on area, power and latency are 61.6%, 53.1% and 69.2%, respectively, compared with Yet Another Decision Diagram (YADD) [9].*

## 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 early design stages. In particular, the irregularity of global interconnects renders the prediction of delay and crosstalk difficult. In Ref. [6], 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.

A main category of regular circuit structures is based on Decision Diagrams. The main idea 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. [9, 8], can be directly mapped to a net-

work 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. Ref. [10] extends the concept of PSBDD by exploring more regular neighbor geometries, but other types of expansions besides canonical Shannon expansion are necessary [5]. In the mapping of FPGA (from PSBDD) and network of multiplexers (from YADD), only predictable local interconnects are needed.

However, repetition of variables in several levels is typically needed for PSBDD and YADD. As reported in [9] 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 corresponding YADD has 33 levels. Consequently, YADD structure has high area and power penalties, although it has far superior timing performance (by applying wave-steering) when compared to standard-cell designs [9]. The objective of this work is to achieve far superior area and power performances than YADD structures while maintaining similar speed 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.<sup>1</sup> Meanwhile, 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 illustrates that. In Figure 1 (and sub-

---

<sup>1</sup>Strictly speaking, it is not necessary to impose the no crossings restriction as long as the interconnect parasitics can be accurately predicted.

sequent 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.

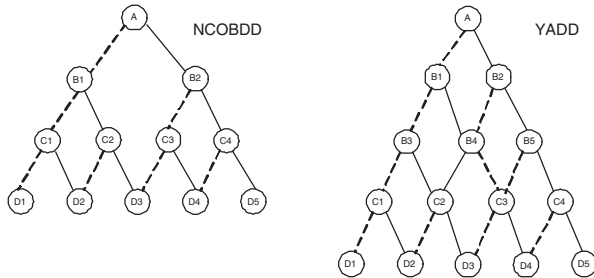


Figure 1. NCOBDD vs. YADD.

NCOBDD preserves the performance and the structural regularity, with less area, less power and lower latency than YADD. Experimental results show that when compared with YADD, NCOBDD reduces the area by 24.2% and power by 28.7% averagely for partially symmetric circuits; for asymmetric circuits, the reductions on area, power and latency in average are 61.6%, 53.1% and 69.2%, respectively. NCOBDDs also have better PDPs (power-delay products) than standard-cell designs for big asymmetric circuits.

## 2 Non-crossing OBDD (NCOBDD)

**Definition** A planar ordered BDD with all connections restricted between two adjacent levels is a **Non-crossing Ordered BDD (NCOBDD)**.

An NCOBDD has the following attributes:

1. It is an ordered BDD with each variable characterizing only one level;
2. All connections are between two adjacent levels;
3. There are no crossings between connections (planar).

Obviously, NCOBDD is neither reduced nor canonical due to attributes (2) and (3). Reduced planar BDD was studied earlier in Ref. [11], but only a small set of specific logic functions can produce planar decision diagrams.

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 as it is noncanonical. Attributes (2) and (3) 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.

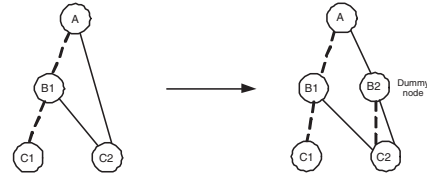


Figure 2. Insertion of dummy nodes.

### 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].

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. Moreover, the 2-level crossing minimization problem with a fixed node ordering in one level is still NP-hard [1]. 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].

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  $i = 2, 3, \dots, k$ , 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.

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  $\{v_1, \dots, v_n\}$  in level  $L_{i-1}$  to determine the ordering and, if necessary, duplication of their child nodes in the current lower level  $L_i$ .

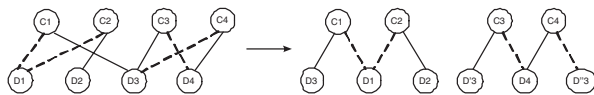


Figure 3. Reordering and duplication.

Consider two adjacent nodes  $v_j$  and  $v_{j+1}$  in level  $L_{i-1}$ . If the position of a child node of  $v_j$  ( $v_{j+1}$ ) has already been determined because it is also a child node

of  $v_h$ ,  $h < j$ , and its connection to  $v_j$  ( $v_{j+1}$ ) causes crossings, the duplication of this child node is necessary. The duplicate replaces the original child node of  $v_j$  ( $v_{j+1}$ ). There are three possible scenarios when we consider the relation of the child nodes of  $v_j$  and  $v_{j+1}$ .

(1) If  $v_j$  and  $v_{j+1}$  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  $v_j$  and  $v_{j+1}$  should reside on the left and right of the common child node, respectively (see nodes  $C_1$  and  $C_2$  in Figure 3). The next two nodes under consideration in level  $L_{i-1}$  are  $v_{j+2}$  and  $v_{j+3}$ .

(2) If  $v_j$  and  $v_{j+1}$  have no common child nodes, the order between two child nodes of  $v_j$  can be arbitrary (see nodes  $B_1$  and  $B_2$  in Figure 4). 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  $v_{j+1}$  and  $v_{j+2}$ .

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

### 2.2.2 Exploiting arbitrariness in reordering and duplication of nodes

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 may affect the results of reordering and duplication operations on subsequent levels. In Figure 4, 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 5. The right one has fewer nodes than the left one.

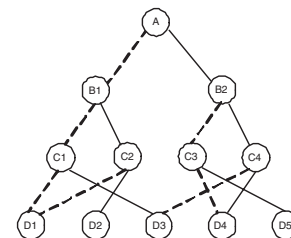


Figure 4. A ROBDD representation.

Therefore, the question is how we can exploit the arbitrariness introduced at the upper levels during the

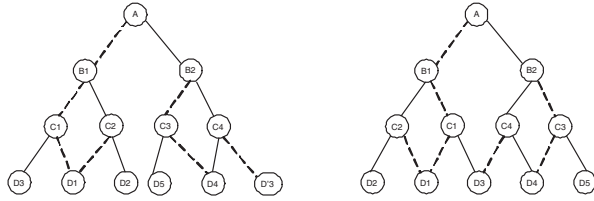


Figure 5. Two NCOBDD representations.

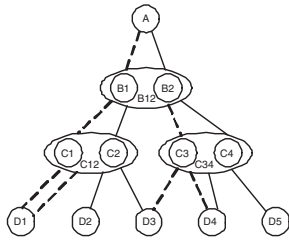


Figure 6. Node collapsing.

level-by-level sweep operations on lower levels. We use an example to illustrate the approach to exploit the arbitrariness introduced in scenario (2). The arbitrariness introduced in scenario (3) can be handled with a simple extension. For adjacent nodes, if the order between them can be arbitrary, we collapse them into one super node. Figure 6, for example, shows the collapsing of BDD nodes in Figure 4.

In Figure 6, nodes  $B_1$  and  $B_2$  are collapsed into a super node  $B_{12}$  first when we start the level-by-level sweep from the top as the order between them can be arbitrary. Similarly, 4 child nodes of super node  $B_{12}$  can be collapsed into two super nodes  $C_{12}$  and  $C_{34}$ . Note that the 4 child nodes cannot be collapsed into one super node since that would cause the possibility of  $C_3$  and  $C_4$  ( $C_1$  and  $C_2$ ) becoming the child nodes of  $B_1$  ( $B_2$ ). In other words, each super node contains only two regular nodes. The super nodes  $C_{12}$  and  $C_{34}$  are considered when we try to eliminate the crossings between levels  $C$  and  $D$ . Some modifications have to be made to the three scenarios of reordering and duplication of nodes described in Section 2.2.1 since the super nodes have more than 2 child nodes and the node collapsing step needs to be incorporated.

For scenario (1), the only common child node of nodes  $v_j$  and  $v_{j+1}$  should still be placed in the middle. With that as a constraint, if node  $v_j$  ( $v_{j+1}$ ) is a super node, the order of the regular nodes within the super node  $v_j$  ( $v_{j+1}$ ) can be determined accordingly. In Figure 6, 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 super nodes, i.e.,  $C_1$  and  $C_2$ , and  $C_3$  and  $C_4$  can be determined; 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.

For scenario (2), if node  $v_j$  is not a super node, the two child nodes of  $v_j$  are collapsed into a super node as the order between them can be arbitrary. If  $v_j$  is a super node, the four child nodes of  $v_j$  are collapsed into 2 super nodes properly (similar to the child nodes of  $B_{12}$  in Figure 6). For scenario (3), the number of common child nodes may be more than 2 if  $v_j$  and  $v_{j+1}$  are super nodes, but only one of the common child nodes does not need to be duplicated.

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 super nodes (nodes  $B_{12}$ ,  $C_{12}$  and  $C_{34}$  in Figure 6 for example). This provides even more flexibility to the node ordering in level  $L_{i-1}$  and above; not only is the ordering *within* the super nodes in level  $L_{i-1}$  not fixed, but the ordering *among* those super nodes can also be changed.

For that reason, we first enumerate all possible orderings *among* the super nodes in level  $L_{i-1}$  to search for the optimal ordering that minimizes the duplication of nodes in the current lower level  $L_i$ . (Two different possible orderings between super 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 6). super 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 super nodes in level  $L_{i-1}$  is obtained, the ordering *within* each of those super nodes is determined with the same goal of minimizing the number of nodes in  $L_i$ . Some super 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. Node collapsing in level  $L_i$  is also performed.

Although such a scheme in general has exponential time complexity, the number of possible orders to be enumerated in level  $L_{i-1}$  in this context is very small as the super 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 sec-

tion, we review the physical mapping of YADD and the wave steering technique applied to YADD [9] briefly.

Mapping BDD structure to pass transistor logic (PTL) 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. 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.

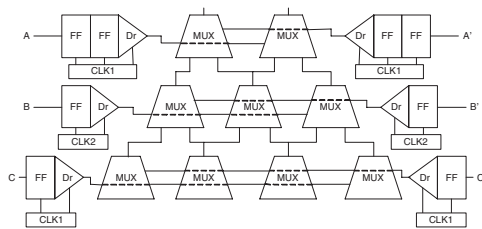


Figure 7. Wave steering (adapted from [9]).

In Ref. [9], 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, and vice versa. 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. 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 7 shows the floorplan of YADD and inputs configuration of wave steering. In Figure 7, 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 bench-

mark 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.

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 two pass n-FET transistors, 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  $0.25\mu\text{m}$  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, maximal and average width of levels, and the number of nodes for the NCOBDDs constructed for each benchmark circuit. It also reports the area (including flip-flops and drivers), the (longest) latency, and the power dissipation (including clock power) 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. [9]. 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 (xor7), NCOBDD has the same area, power and latency as YADD; (they are exactly the same circuits). For partially symmetric circuits (9sym, cm138a, and rd73), NCOBDD reduces the area by 24.2% and power by 28.7%, while maintaining the same latency. For asymmetric circuits (alu2, alu4, too\_large, apex4, and misex3), the reductions on area, power and latency are 61.6%, 53.1% and 69.2%, 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. [9], after taking into account the different technology used. (Ref. [9] used  $0.5\mu\text{m}$  tech-

circuit	xor7	9sym	cm138a	rd73	alu2	alu4	too_large	apex4	misex3
primary inputs	7	9	6	7	10	14	15	9	14
primary outputs	1	1	8	3	6	8	3	18	14
NCOBDD	levels	7	9	6	7	10	14	9	14
	max width	7	6	1	7	31	81	35	46
	avg width	4	4	1	3	13	37	11	12
	nodes	28	33	48	68	465	3482	446	1866
	run time (s)	2	4	3	3	8	18	11	16
	area (x1000 $\mu\text{m}^2$ )	2.947	4.102	6.682	7.603	35.023	210.070	37.564	98.938
	latency (ns)	2.1	2.7	1.8	2.1	3	4.2	4.5	2.7
	power (mW)	0.499	1.322	0.925	1.229	5.635	12.245	5.129	9.004
PDP (ns x mW)	0.300	0.793	0.555	0.737	3.38	7.347	3.077	5.402	
YADD	levels (= max width)	7	9	6	7	33	43	40	37
	avg width	4	5	3.5	4	14	20	18	13
	nodes	28	45	168	84	1369	6264	1978	5711
	area (x1000 $\mu\text{m}^2$ )	2.947	4.869	12.850	8.354	89.021	465.760	91.235	320.784
	latency (ns)	2.1	2.7	1.8	2.1	9.9	12.9	12.0	11.1
	power (mW)	0.499	1.534	1.601	1.761	9.749	28.802	9.057	21.890
	PDP (ns x mW)	0.300	0.920	0.961	1.233	5.849	17.281	5.434	13.134
Standard-Cell	area (x1000 $\mu\text{m}^2$ )	1.132	2.544	0.996	1.584	9.975	48.363	12.466	33.518
	delay (ns)	0.678	1.086	0.598	0.934	3.360	5.772	3.930	2.736
	power (mW)	0.298	0.577	0.346	0.495	2.698	5.866	3.645	5.061
	PDP (ns x mW)	0.202	0.627	0.207	0.462	9.065	33.859	14.325	13.847

**Table 1. Experiment results.**

nology). To the best of our knowledge [7], the area for each of the two circuits reported in Ref. [9] 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  $\mu\text{m}$  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 (products of power and minimum clock cycle used here) of NCOBDD-mapped circuits are better for bigger asymmetric circuits. Of course, the standard-cell design is the most area efficient among the three. The area penalty that NCOBDD incurs is quite high for large benchmark circuits. A possible solution to this problem is to partition a given circuit into smaller components, each of which can be implemented by a NCOBDD structure. This will be explored in the future.

## References

- [1] G. D. 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. D. 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] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. Perkowski. Efficient representation and manipulation of switching functions based on ordered kronecker functional decision diagrams. In *Proc. Design Automation Conf*, pages 415–419, June 1994.
- [6] 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.
- [7] A. Mukherjee. Personal communication, Sept. 2002.
- [8] A. Mukherjee. *Wave steering as a means for achieving performance and predictability in DSM circuits*. PhD thesis, University of California, Santa Barbara, 2002.
- [9] 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.
- [10] M. Perkowski, E. Pierzchala, and R. Drechsler. Layout-driven synthesis for submicron technology: mapping expansions to regular lattices. In *Proc. International Conf. on Information, Communications and Signal Processing*, pages 9–12, Sept. 1997.
- [11] T. Sasao and J. T. Butler. Planar multiple-valued decision diagrams. In *Proc. IEEE International Symp. on Multiple-Valued Logic*, pages 28–35, May 1995.