

Decomposition of BDDs with Application to Physical Mapping of Regular PTL Circuits

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

ABSTRACT

Non-Crossing Ordered BDDs (NCOBDDs) [3], were proposed for mapping to pass transistor logic (PTL) circuits with very high performance and regular layout. However, the latency and area penalty prevents it to be applied to large circuits. In this paper, we propose a dynamic-programming based approach for decomposition of BDDs. The decomposed BDDs can be applied to construct more compact NCOBDDs that have lower latency, area, and power than the monolithic ones. Experimental results show that, on average, the decomposed NCOBDDs can reduce the latency by 45%, and reduce the area by 53%, respectively.

1. INTRODUCTION

Due to the technology scaling, interconnects have become more and more dominant in many aspects, such as performance, robustness, and reliability, in today's VLSI circuits and systems. On one hand, the analysis and prediction of the interconnect effects in the early design stages are difficult. On the other hand, it may be too late to incorporate the interconnect effects in the later physical design stage. One particular direction to solving this dilemma is to have regular circuit structures, where interconnect effects are predictable. Regular circuit structures could overcome the timing closure problem and offset the process variation, and would scale better with technology [10].

A BDD-based regular structure, called pseudo-symmetric BDD (PSBDD), was first proposed in [4] for Cellular-Array type FPGA synthesis. [12] presented a more generalized BDD-based regular structure, called Yet Another Decision Diagram (YADD). YADDs can be mapped to multiplexor-based regular pass transistor logic (PTL) circuits by mapping each node into a multiplexor. Both PSBDDs and YADDs eliminate the irregular interconnections between nodes in Reduced Ordered BDDs (ROBDDs), such that the mapped FPGAs or PTL circuits have regular layouts and predictable interconnects [4, 12]. However, the decision variables in a

PSBDD or YADD may appear in more than one level during the application of Shannon's expansion. With each decision variable potentially appearing on several levels, the total number of levels and thus the number of nodes in the PSBDD or YADD usually are much more than those of the corresponding ROBDD. This converts to longer latency and larger area penalty in the mapped FPGA or PTL circuit.

In order to reduce the latency and area overheads for PSBDDs and YADDs, a more compact structure, called Non-Crossing Ordered BDD (NCOBDD), was proposed in [3]. An NCOBDD allows each decision variable to appear in only one level. However, the regularity of interconnects is still retained in the NCOBDD by restricting all connections between nodes to be between adjacent levels and enforcing no crossings between connections. A top-down level-by-level sweep technique along with nodes reordering and duplicating operations were proposed in [3] to transform ROBDDs into NCOBDDs. NCOBDDs can also be directly mapped to Cellular-Array type FPGAs or PTL circuits.

The interconnects in an NCOBDD-mapped PTL circuit are local and have predictable delays so that wave steering (a form of wave pipelining) can be applied as in a YADD-mapped circuit [12]. By having a two-phase clocking scheme, each level of multiplexors in the PTL circuit alternates between two modes: "hold" and "evaluate", and two neighboring levels work in different modes. Therefore, there are several computing waves propagating through the circuit at the same time. Consequently, the PTL circuit can operate under very high clock frequency. Moreover, the crosstalk problem is alleviated in the PTL circuit.

An NCOBDD has a smaller size than its corresponding PSBDD and YADD as the number of levels equals that of the corresponding ROBDD. However, to implement a large circuit with a lot of primary inputs, the area penalty incurred by the NCOBDD-mapped circuit can still be substantial since the total number of nodes in the NCOBDD grows more than quadratically, sometimes even exponentially, with the number of levels. Therefore, if we can further reduce the number of levels in NCOBDDs, more reduction on area can be expected, making NCOBDDs more practical for large circuits. Furthermore, it can be expected that the delay (or latency if wave steering is applied) of the NCOBDD-mapped PTL circuits would also reduce. We observe that using decomposed NCOBDDs instead of the monolithic ones can achieve such objectives of minimizing area, delay, and/or

latency.

As an NCOBDD can be constructed from an ROBDD, we tackle the problem of decomposing the NCOBDD by first decomposing the corresponding ROBDD. Although the decomposed ROBDD may be larger in terms of the number of nodes than the monolithic one, the size of the NCOBDD constructed from the decomposed ROBDD is likely to be smaller.

In this paper, a dynamic-programming based method is proposed for the decomposition of a given ROBDD. The NCOBDD constructed from the decomposed ROBDD can achieve much smaller delay (or latency) after being mapped to a PTL circuit. The area penalty for the decomposed NCOBDD is also reduced tremendously. Experimental results show that, on average, the decomposed NCOBDDs reduce the delay (or latency) by 45% and the area by 53% when compared with the monolithic ones. Our approach can also be applied to the decomposition of PSBDDs and YADDs and similar improvements can be expected.

The paper is organized as follows. Section 2 introduces the decomposition of BDDs. We propose a dynamic-programming based BDD decomposition approach in Section 3. Experimental results are given in Section 4. We review some related work on BDD decomposition in Section 5 and conclude the paper in Section 6.

2. PRELIMINARIES

Functional decomposition has been studied widely as a technique in logic synthesis. The instance most studied is the one-sided decomposition, defined as follows:

Given a Boolean function $f(x_1, \dots, x_r, x_{r+1}, \dots, x_n)$, the functional decomposition of f with respect to the variable subset $\{x_1, \dots, x_r\}$ is a composition function:

$$g(h_1(x_1, \dots, x_r), \dots, h_t(x_1, \dots, x_r), x_{r+1}, \dots, x_n) = f. \quad (1)$$

The subset $\{x_1, \dots, x_r\}$ is denoted as the *bound set*, and $\{x_{r+1}, \dots, x_n\}$ as the *free set*. Each h_i , $1 \leq i \leq t$, is called an *encoding function or decomposition function* of the decomposition, and g is called the *composition function*.

ROBDD provides a compact representation for practical and efficient decomposition of a function. In ROBDD-based decomposition, let f be represented by an ROBDD with the bound set variables on the top. We take a cut across the ROBDD with the bound set variables above the cut line; let p denote the number of distinct nodes below the cut that are connected to the edges across the cut. These p number of nodes will be replaced by binary-encoded numbers in the encoding functions. Since the minimum number of bits of a binary number to encode p number of nodes is $\lceil \log(p) \rceil$, $\lceil \log(p) \rceil$ is also the minimum number of encoding functions. Figure 1 shows a simple example of a one-sided decomposition for the function $xor7$. Using ROBDD representations, the encoding functions become the decision variables of the ROBDD of the composition function. We call them *encoded variables*.

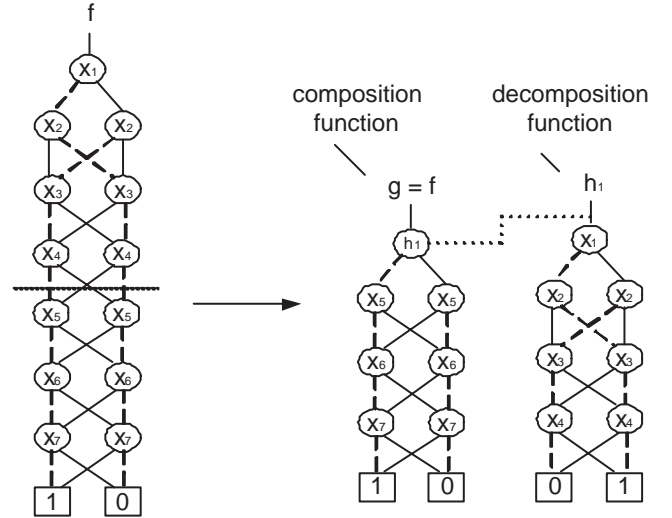


Figure 1: A one-sided decomposition for the function $xor7$.

More generally, a k -sided functional decomposition of f is:

$$g(h_1^1(x_1, \dots, x_r), \dots, h_{t_1}^1(x_1, \dots, x_r), \dots, h_1^k(x_{m+1}, \dots, x_n), \dots, h_{t_k}^k(x_{m+1}, \dots, x_n)) = f, \quad (2)$$

where bound sets $B_1 = \{x_1, \dots, x_r\}, \dots, B_k = \{x_{m+1}, \dots, x_n\}$ is a disjoint partition of the set $\{x_1, \dots, x_n\}$. More generally, the encoding function can be as trivial as $h(x_i) = x_i$. Therefore, Eqn. (1) becomes:

$$g(h_1(x_1, \dots, x_r), \dots, h_t(x_1, \dots, x_r), h(x_{r+1}), \dots, h(x_n)).$$

The k -sided decomposition can be performed by iteratively performing one-sided decompositions on the ROBDD. Figure 2 shows a possible k -sided decomposition for the function $xor7$.

In the next section, we propose a method of k -sided ROBDD decomposition with the objective of reducing delay (or latency) as much as possible when compared with the monolithic one.

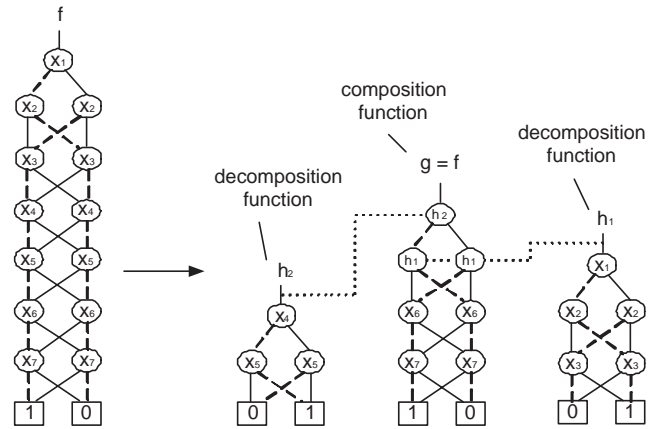


Figure 2: A possible k -sided decomposition for the function $xor7$.

3. BDD DECOMPOSITION

3.1 Problem formulation

Given an ROBDD whose decision variables are $\{x_1, \dots, x_n\}$, which are the control signals of the multiplexers after mapping the ROBDD to a multiplexor-based PTL circuit, assume that the variable ordering is (x_1, \dots, x_n) . Assuming that the arrival time of x_i , $1 \leq i \leq n$, is a_{x_i} , and each multiplexer has unit delay, the delay of the PTL circuit is:

$$\max_{1 \leq i \leq n} (a_{x_i} + i), \quad (3)$$

where i stands for i number of unit delays. In Figure 2, for example, if we assume that each primary input has a_{x_i} , $1 \leq i \leq n$, equal to 0, the delay of the mapped PTL circuit for the original function f is 7. Now consider the decomposition of f . The decision variables of the composition function $g(= f)$ are $\{h_1, h_2, x_6, x_7\}$. The delays of the two decomposition functions, a_{h_1} and a_{h_2} , are 3 and 2, respectively. Therefore, the delay for the composition function g is 5, which is determined by $(a_{h_1} + 2)$. That, however, is not the minimal delay for the composition function g .

Given the arrival times $\{a_{x_1}, \dots, a_{x_n}\}$, we can sort them from the latest to the earliest. Assuming that the sorted arrival times are $(a_{x'_1}, \dots, a_{x'_n})$, we reorder the ROBDD according to the variable ordering (x'_1, \dots, x'_n) . After mapping the reordered ROBDD to a new PTL circuit, obviously, the delay (or latency) of the new PTL circuit,

$$\max_{1 \leq i \leq n} (a_{x'_i} + i), \quad (4)$$

is less than or equal to the delay of the original one. Actually, the delay in Eqn. (4) is the minimal among the PTL circuits mapped from all possible ROBDDs with different variable orderings. Take the composition function g in Figure 2 for example. The sorted arrival time of the decision variables of the function g is $(a_{h_1}, a_{h_2}, a_{x_6}, a_{x_7})$. Therefore, the minimal delay for the composition function g is 4 ($a_{h_2} + 2$ or $a_{h_1} + 1$), as shown in Figure 3.

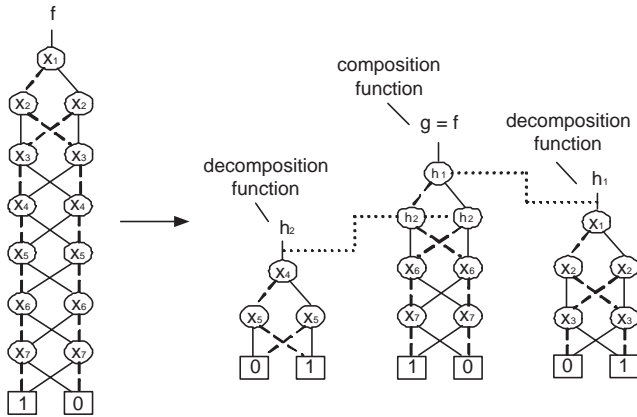


Figure 3: Another k -sided decomposition for the function $xor7$ with smaller delay.

The examples shown above have the same decomposition functions for the function f . Now consider different possibilities of decomposition. Obviously, the delay of the PTL

circuit mapped from the monolithic ROBDD of the function $f(x_1, \dots, x_n)$ is n assuming that all primary inputs arrive at time 0. Consider the one-sided functional decomposition of f in Eqn. (1); the minimal delay of the composition function g is:

$$\max(r, n - r) + t. \quad (5)$$

The first term in Eqn. (5) corresponds to the maximum between the height (r) of the encoding functions h_i , $1 \leq i \leq t$, and the height ($n - r$) of the stacked free set variables $\{x_{r+1}, \dots, x_n\}$. Note that the heights of some encoding functions may be smaller than r after performing the *reduction* operation to transform them into *reduced* form. Here, for simplicity, we assume that the height of the ROBDD of each encoding function retains value r after the *reduction* operation. The second term t in Eqn. (5) is the height of the stacked encoded variables that are put on the top of the ROBDD of the composition function g . When $r < \frac{n}{2}$ and $t < r$, Eqn. (5) becomes $n - r + t$, which is smaller than n .

If iterative decompositions are applied for the k -sided decomposition, the delay of the composition function may have more reduction. Our objective in this paper is to search for the decomposition such that the decomposed function has the minimum delay.

The problem is formulated as follows:

Given a function $f(x_1, \dots, x_n)$, decompose f based on the corresponding ROBDD, such that the PTL circuit mapped from the decomposed ROBDD has the minimum delay.

3.2 Ordered decomposition

Unfortunately, the problem formulated in the previous section is very difficult and probably is NP-hard. To simplify the problem, we make the assumption that the ROBDD of the function f has a fixed variable ordering (x_1, \dots, x_n) and the decomposition would not change this variable ordering. In other words, with the above assumption, a k -sided functional decomposition of the function f , whose ROBDD has the variable ordering (x_1, \dots, x_n) , is:

$$g(h_1^1(B_1), \dots, h_{t_1}^1(B_1), \dots, h_1^k(B_k), \dots, h_{t_k}^k(B_k)),$$

in which the bound sets for the decomposition have the format as follows:

$$B_1 = \{x_1, \dots, x_{|B_1|}\}, B_2 = \{x_{|B_1|+1}, \dots, x_{|B_1|+|B_2|}\}, \\ \dots, B_k = \{x_{|B_1|+\dots+|B_{k-1}|+1}, \dots, x_n\}.$$

We call this simplified problem the *ordered logic decomposition* problem. The restriction imposed by the ordered logic decomposition problem would limit the search for the minimum delay of the decomposed function. However, the delay reduction of the decomposed function compared with the original one can be substantial, as we shall see in Section 4.

The ordered decomposition problem can be rephrased as follows:

Given a fixed variable ordering (x_1, \dots, x_n) of the ROBDD of a function f , partition (x_1, \dots, x_n) into disjoint bound sets, each of which maintains the original variable ordering, and decompose f based on these bound sets, such that the

decomposed function has the minimal delay.

3.3 Dynamic-programming based heuristic

After restating the problem, the problem reduces to a disjoint partitioning of an ordered list, and it becomes clear that a divide-and-conquer methodology can be applied to obtain a satisfactory solution.

Let $d_{i,j}$, $i \leq j$, represent the minimal delay of the decomposed function by partitioning (x_i, \dots, x_j) into disjoint bound sets. Our final objective is to calculate $d_{1,n}$. Let $l_{i,j}$ represent the sorted list of the arrival time of the encoding variables corresponding to the optimal partitioning (or decomposing). $l_{i,j}$ is called the *companion list* of $d_{i,j}$.

For the divide-and-conquer methodology, let us assume that the minimal delays and the corresponding companion lists, $d_{i,k}$, $d_{k+1,j}$ and $l_{i,k}$, $l_{k+1,j}$, for all k , $i \leq k < j$, are known. For any k , $i \leq k < j$, define $D_{i,k,j}$ and $L_{i,k,j}$ to be the minimal delay and the companion list corresponding to the decomposition that x_k and x_{k+1} belong to different bound sets. $L_{i,k,j}$ can be easily derived by joining $l_{i,k}$ and $l_{k+1,j}$ and sorting the joined list, since $l_{i,k}$ and $l_{k+1,j}$ correspond to disjoint bound sets. We use S to represent the joining and sorting operations on two lists, so $L_{i,k,j} = S(l_{i,k}, l_{k+1,j})$. $D_{i,k,j}$ can be directly calculated from $L_{i,k,j}$ using Eqn. (4). For simplicity, here we assume that the arrival time of each primary input x_i , $1 \leq i \leq n$, is 0. However, the following dynamic-programming based formulation can be easily extended to consider the nonuniform arrival times of the primary inputs.

There are two special cases worthy of mentioning:

Case 1: When i equals j , there is only one variable x_i in a bound set; the decomposition on this bound set is trivial. Therefore, we treat x_i as a variable in the free set in this case and set $d_{i,i} = 1$, and $l_{i,i} = (0)$.

Case 2: When k equals j , it corresponds to the case that (x_i, \dots, x_j) forms one bound set and the decomposition is performed with respect to this bound set. Assume that the number of encoding functions corresponding to the bound set $\{x_i, \dots, x_j\}$ is $t_{i,j}$, then $D_{i,j,j} = j - i + 1$, and

$$L_{i,j,j} = \underbrace{(j - i + 1, \dots, j - i + 1)}_{t_{i,j} \text{ number of elements}} \quad (6)$$

After considering all cases, the formula of $d_{i,j}$ is as follows:

$$d_{i,j} = \begin{cases} 1, & \text{if } i = j; \\ \min(j - i + 1, \min_{i \leq k < j} D_{i,k,j}), & \text{if } i < j; \end{cases} \quad (7)$$

From Eqn. (7), we can get the value of k' corresponding to the minimal value of $d_{i,j}$. If $k' < j$, $l_{i,j} = S(l_{i,k'}, l_{k'+1,j})$. Note that the first term $(j - i + 1)$ corresponds to $k' = j$. If $k' = j$, Eqn. (6) is applied to derive $l_{i,j}$. When there is a tie for $d_{i,j}$, we pick the minimum of $|L_{i,k,j}|$ to break the tie, where $|L_{i,k,j}|$ is the number of elements in $L_{i,k,j}$. Note that $|L_{i,k,j}| = |l_{i,k}| + |l_{k+1,j}|$, denoting the number of encoding functions for the partition of (x_i, \dots, x_j) . It is intuitive that fewer encoding functions would result in smaller delay.

Moreover, it is also of benefit to area minimization.

Obviously, Eqn. (7) is suitable for dynamic programming as there are only $O(n^2)$ number of subproblems in total. However, the problem may not satisfy the property of optimal substructure for the dynamic programming. The optimal solutions to two subproblems may not always form an optimal solution as joining and sorting the two sorted sublists may destroy the optimality of Eqn. (7). Therefore, although we can still perform the dynamic programming, the optimality of the solution is not guaranteed.

In order to apply the dynamic programming, we have to calculate the number of encoding functions corresponding to the bound set $\{x_i, \dots, x_j\}$, $t_{i,j}$, $1 \leq i \leq j \leq n$. Assume that the bound set variables x_i to x_j are on the top of the ROBDD, and we take a cut below the bound set variables. If p denotes the number of distinct nodes below the cut that are connected to the edges across the cut, $t_{i,j} = \lceil \log(p) \rceil$. Therefore, in order to calculate $t_{i,j}$, first we reorder the original ROBDD to have the variables x_i to x_j on the top. We use the BDD package CUDD [18] to perform the variable reordering on the ROBDD.

4. EXPERIMENTAL RESULTS

We have implemented the dynamic-programming based approach presented in Section 3 in C++ language. The BDD package CUDD [18] is used for the ROBDD construction, variable reordering, and other BDD operations. We have performed the experiments on ten benchmark circuits. The BDD decomposition program runs on a PC with Pentium IV 2.8GHz CPU and 1Gigabytes memory.

For each benchmark circuit, we build the monolithic ROBDD first. After that, the decomposition is performed using the dynamic-programming based approach. Based on the decomposed ROBDD, the NCOBDD is constructed [3]. The decomposed NCOBDD is then mapped to the PTL circuit using 0.35 μ m technology. The wave-steering is applied with two-phase clocking [12]. The delay of each level of multiplexors is less than 0.4ns, making 1.25GHz (a clock cycle of 0.8ns) the highest achievable clock frequency at which each circuit can work properly. The latency, area, and power of the decomposed NCOBDD-mapped circuit are included in Table 1.

For comparison, the monolithic NCOBDD is also constructed based on the monolithic ROBDD for each benchmark circuit. The same physical mapping and wave-steering are applied. From Table 1, we can see that when compared with the monolithic ones, the decomposed NCOBDDs reduce the latency by 45%, the area by 53%, and the power by 33% on average. (Our decomposition approach can also be applied to decompose monolithic PSBDDs and YADDs to achieve similar improvements.)

We also implement the benchmark circuits with standard cells. We use Silicon Ensemble to generate the layout using 0.35 μ m standard cell library. Parasitic extraction is then carried out and simulation performed. The area, timing, and power performance of the standard cell circuits are also included in Table 1. The total area of the PTL circuits mapped from the decomposed NCOBDDs is three

circuit		C432	k2	C880	i9	C1355	C1908	dalu	C3540	C5315	x4	ratio
primary inputs		36	45	60	88	41	33	75	50	178	94	-
primary outputs		7	45	26	63	32	25	16	22	123	71	-
Monolithic NCOBDD	latency(ns)	14.4	18	24	35.2	16.4	13.2	30	20	71.2	37.6	1
	area(x1000 μm^2)	484.7	510.5	665.6	828.5	701.0	935.5	833.7	1297.6	1985.4	782.2	1
	power(mW)	81.0	360.4	221.4	388.4	276.5	303.2	368.2	605.7	913.4	401.4	1
	energy (ns x mW)	64.8	288.3	177.1	310.7	221.2	242.6	294.6	484.6	730.7	321.1	1
Decomposed NCOBDD	latency(ns)	8	10	12	18.4	9.6	9.6	16	10.8	36.4	16.8	0.548
	area(x1000 μm^2)	146.3	386.5	260.2	440.1	278.5	355.3	318.9	642.0	885.9	497.5	0.472
	power(mW)	57.6	234.7	192.5	249.8	188.7	211.5	220.5	326.3	545.7	272.9	0.667
	energy(ns x mW)	46.1	187.8	154	199.8	151	169.2	176.4	261	436.6	218.3	0.667
Decomposition time(s)		16	81	53	196	107	89	322	245	968	280	-
Standard Cell	delay(ns)	9.4	19.6	13.6	15.9	17.5	18	22.3	24.2	26.8	19.8	0.805
	area(x1000 μm^2)	34.8	287.6	71.4	90.4	107.3	111.0	149.9	167.7	258.5	91.3	0.168
	power(mW)	26.8	96.7	49.3	59.9	61.5	66.5	108.1	120.2	144.4	67.4	0.224
	energy(ns x mW)	251.9	1895.3	670.48	952.41	1076.3	1197	2410.6	2908.8	3869.9	1334.5	5.075

Table 1: Experimental results comparison among decomposed and monolithic NCOBDD and standard cell.

times larger than that of the standard cell designs. However, the energy consumed by the PTL circuits (defined to be the product of the clock period and power dissipation) is only a seventh of that of the standard cell designs. It should be noted that the energy consumed by the standard cell designs may be reduced by pipelining the logic at the expense of increasing area, power consumption, and latency.

5. RELATED WORK

Functional decomposition was first studied by Ashenurst and Curtis [1, 5]. After BDDs were utilized for functional decomposition, a lot of work had been done in this area, which can be classified into two categories.

Most of the work on the functional decomposition focused on the synthesis of different types of circuits [8, 7, 14, 15, 19, 20, 9, 2, 16, 17]. BDD-based functional decomposition was used in [8, 7, 14, 15] for the synthesis of lookup-table based FPGAs. [19, 20] performed iterative BDD decompositions by dominators leading to static CMOS logic gates decomposition. [2] used the heuristics proposed in [6] to synthesize the decomposed BDD-based PTL circuits. In [16, 17], a max-flow min-cut based recursive bipartitioning of a BDD was proposed for high performance or low power PTL synthesis.

The other category of work focused on the construction of a BDD. [13, 6] proposed two heuristics to build a decomposed BDD in order to avoid the peak intermediate memory requirement during the BDD construction. One heuristic is to find a good cut set of the logic network as the decomposition set, then build the BDD for each component of the decomposed logic network. The other one is to introduce a decomposition point during the process of the BDD construction whenever the BDD size is over a limit or increases by a disproportionate amount.

In this paper, our work can be classified as using iterative one-sided BDD decomposition for PTL synthesis. The decomposed ROBDD is applied to construct the NCOBDD. Our objective is to reduce both delay and area, which are the two limiting factors for NCOBDDs. The heuristics pro-

posed in [6] and applied in [2] for PTL synthesis may result in even longer delay for the decomposed BDD than the corresponding monolithic one. Longer delay usually means larger number of levels, which is not suitable for constructing NCOBDDs either. The problem in [16, 17] is that one-hot encoding was used to encode the decomposed BDD. Consequently, the number of decomposed BDDs grows very fast with the recursive bipartitioning, which is not suitable for NCOBDDs due to the area increase.

6. CONCLUSION AND FUTURE WORK

We propose a dynamic-programming based approach for BDD decomposition. The decomposed BDD can be applied to construct the NCOBDD. Our approach can result in much less delay and area after mapping the decomposed NCOBDD to the PTL circuit. However, the decomposition of NCOBDDs introduces additional irregular global interconnects. As a future research direction, we propose to investigate the problem of determining the placement of decomposed blocks, with the objective of eliminating irregular global interconnects. We may adopt the approach taken in the placement of PLAs in [11].

7. REFERENCES

- [1] R. Ashenurst. The decomposition of switching functions. In *Proc. Int. Symp. on the Theory of Switching Functions*, volume 29, pages 74–116, 1959.
- [2] P. Bush, A. Narayan, A. R. Newton, and A. Sangiovanni-Vincentelli. Logic synthesis for large pass transistor circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 663–670, Nov. 1997.
- [3] A. Cao and C.-K. Koh. Non-crossing OBDDs for mapping to regular circuit structures. In *Proc. IEEE Int. Conf. on Computer Design*, pages 338–343, Oct. 2003.
- [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] H. A. Curtis. *A New Approach to The Design of Switching Circuits*. D. Van Nostrand Co., Princeton, NJ, 1962.
- [6] J. Jain, A. Narayan, C. Coelho, S. P. Khatri, A. Sangiovanni-Vincentelli, R. K. Brayton, and M. Fujita. Decomposition techniques for efficient ROBDD construction. In *Proc. Int. Conf. on Formal Methods in Computer-Aided Design*, pages 419–434, Nov. 1996.
- [7] Y.-T. Lai, K.-R. Pan, and M. Pedram. OBDD-based function decomposition: algorithms and implementation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(8):977–990, 1996.
- [8] Y.-T. Lai, M. Pedram, and S. Vrudhula. BDD based decomposition of logic functions with application to FPGA synthesis. In *Proc. Design Automation Conf*, pages 642–647, June 1993.
- [9] A. Mishchenko, B. Steinbach, and M. Perkowski. An algorithm for bi-decomposition of logic functions. In *Proc. Design Automation Conf*, pages 103–108, June 2001.
- [10] 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.
- [11] 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, Nov. 2002.
- [12] 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.
- [13] A. Narayan, J. Jain, M. Fujita, and A. Sangiovanni-Vincentelli. Partitioned ROBDDs - a compact, canonical and efficiently manipulable representation for boolean functions. In *Proc. Int. Conf. on Computer Aided Design*, pages 547–554, Nov. 1996.
- [14] T. Sasao. FPGA design by generalized functional decomposition. In *Logic synthesis and optimization*. Kluwer Academic Publisher, 1999.
- [15] C. Scholl and P. Molitor. Communication based FPGA synthesis for multi-output boolean functions. In *Proc. Asia South Pacific Design Automation Conf.*, pages 279–287, Jan. 1995.
- [16] R. S. Shelar and S. S. Sapatnekar. Recursive bipartitioning of BDDs for performance driven synthesis of pass transistor logic circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 449–452, Nov. 2001.
- [17] R. S. Shelar and S. S. Sapatnekar. An efficient algorithm for low power pass transistor logic synthesis. In *Proc. Asia South Pacific Design Automation Conf.*, pages 87–92, Jan. 2002.
- [18] F. Somenzi. CUDD: CU Decision Diagram package, release 2.3.1.
- [19] C. Yang and M. Ciesielski. BDD decomposition for efficient logic synthesis. In *Proc. IEEE Int. Conf. on Computer Design*, pages 626–631, Oct. 1999.
- [20] C. Yang, M. Ciesielski, and V. Singhal. BDS: a BDD-based logic optimization system. In *Proc. Design Automation Conf*, pages 92–97, June 2000.