BDD Circuit Optimization for Path Delay Fault-Testability

Görschwin Fey   Junhao Shi   Rolf Drechsler
Institute of Computer Science
University of Bremen
28359 Bremen, Germany
{fey,junhao,drechsle}@informatik.uni-bremen.de

Abstract

The complexity of integrated circuits is rapidly growing. This leads to more and more time and money spent on the test of these circuits. Besides minimizing the logic needed for a given function the testability of the resulting circuit becomes a major issue during synthesis. One way to synthesize a circuit for a given function is to directly convert the Binary Decision Diagram *(BDD) of that function into a circuit. It is known that optimizations of the BDD transfer to the derived circuit. Therefore in this paper we evaluate different optimization techniques for BDDs based on variable reordering with respect to the path delay fault testability of the resulting circuit. We show an optimization strategy that allows to compromise during synthesis between logic size and testability.*

1. Introduction

Size and complexity of physically producible circuits steadily increase. This establishes the need to consider the aspect of testing for physical faults already during synthesis since otherwise this task becomes too time consuming.

Binary Decision Diagrams (BDDs) are often used in VLSI CAD systems for efficient representation and manipulation of Boolean functions [5, 6]. In particular they have been studied in logic synthesis, since they allow to consider aspects of technology mapping [11]. It is a well-known synthesis approach to map the BDD to a multiplexor circuit. Testability of these circuits with respect to the *Path Delay Fault Model* (PDFM) has been investigated in [2, 3] and [1] already. The PDFM is a very powerful fault model that allows to detect static and dynamic faults. The major drawback is the large number of paths that have to be tested. Solutions based on test points or synthesis transformations have been proposed in [12, 14], but e.g. test point insertion requires additional hardware.

Here, we use techniques to minimize the BDD. Thus, the optimization can be carried out earlier in the synthesis process by using symbolic methods. Our methods are based on variable reordering. Sifting is a well-known and frequently used reordering heuristic to minimize the number of nodes in a BDD [15]. But we apply sifting to minimize with respect to the number of paths, the average path length or a combined measure incorporating number of nodes and paths. This optimizations of the BDD directly transfer to the circuit. Experiments show the feasibility of the approach.

The structure of the paper is as follows: In Section 2 BDDs and BDD circuits are defined and the PDFM is briefly reviewed. The minimization of BDDs is described in Section 3. Experimental results are given in Section 4, followed by the conclusions in Section 5.

2. Preliminaries

2.1. Binary Decision Diagrams and Circuits

As is well-known, each Boolean function $f : B^n \to B$ can be represented by a BDD, i.e. a directed acyclic graph where a Shannon decomposition is carried out in each node. A BDD is called ordered if each variable is encountered at most once on each path from the root to a terminal node and if the variables are encountered in the same order on all such paths. A BDD is called reduced if it does neither contain isomorphic sub-BDDs nor vertices with both edges pointing to the same node. In the following, only reduced, ordered BDDs are considered and for briefness these graphs are called BDDs. BDDs are a canonical representation, i.e. for each Boolean function the BDD can be uniquely determined. Furthermore, for functions represented by BDDs efficient manipulations are possible [5].

A combinational circuit implementing a function f can be retrieved from a BDD by replacing each node of the function with a multiplexer (see Figure 1). This multiplexer can be implemented by basic gates over the standard library consisting of primary input and output ports, the 2-input/1-output AND- and OR-gate and the 1-input/1-output inverter NOT.

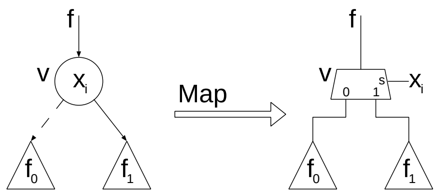
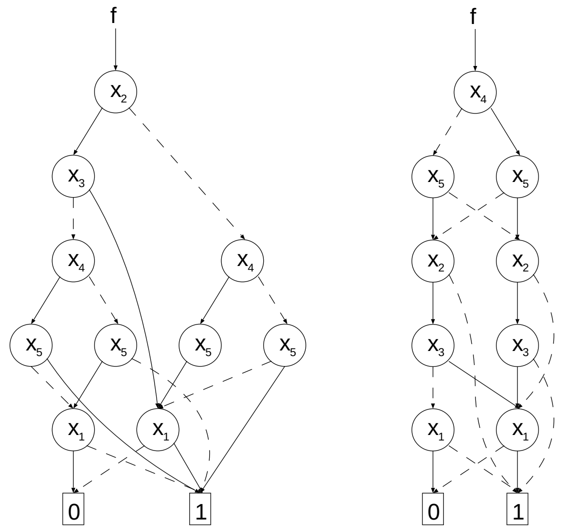


Figure 1. Mapping a BDD to a circuit



(a) Order $(x_2, x_3, x_4, x_5, x_1)$ (b) $(x_2, x_3, x_4, x_5, x_1)$

Figure 2. BDDs of f with respect to two variable orders

Definition 1. *The* size *of a circuit is given by the number of gates it contains.*

Definition 2. *The* depth *of a circuit is the maximal number of gates on a path from an input to an output.*

2.2. Path Delay Fault Model

In this paper we consider a dynamic fault model, i.e. the *Path-Delay Fault Model* (PDFM) [18]. In the PDFM it is checked whether the propagation delays of all physical paths in a given combinational circuit are less than the system clock interval. To this end a transition, i.e. a change in the value of a signal, has to be propagated from primary inputs to primary outputs. A transition ($0 \to 1$ = rising, $1 \to 0$ = falling) propagates along a path, if a sequence of transitions $t_0, t_1, \dots, t_{n+1}$ occurs at the nodes $v_0, v_1, \dots, v_{n+1}$, such that t_i occurs as a result of t_{i-1}.

Definition 3. *A path has a* Path Delay Fault *(PDF), if the actual propagation delay of a (rising or falling) transition along the path exceeds the system clock interval.*

For the detection of a path delay fault a pair of patterns (I_1, I_2) is required: The initialization vector I_1 is applied and all signals of the circuit are allowed to stabilize; then the propagation vector I_2 is applied and after the system clock interval the outputs of the circuit are controlled. Two tests for each physical path have to be carried out, since a check is done for rising and falling transitions.

Definition 4. *A two-pattern test is called a* robust test *for a path delay fault (RPDF test) on a path, if it detects that fault independently of all other delays in the circuit and all other delay faults not located on the path.*

For a detailed discussion of the PDFM see [13].

3. Minimizing the BDDs

The variable order of a BDD has a major influence on the number of its nodes. But to decide, if the number of nodes of a given BDD can be improved by variable reordering, is NP-complete [4]. Therefore heuristics are used to minimize BDDs of non-trivial functions.

One well-known heuristic is sifting [15]: A variable is chosen and moved to any position of the variable order based on exchanging adjacent variables. Then it is fixed at the best position (i.e. where the smallest BDD results), afterwards another variable is chosen. No variable is chosen twice during this process. This heuristic can be used to minimize a BDD with respect to other goals than the number of nodes by changing the objective function that measures the size.

We consider four objective functions:

1. Number of paths: P
2. Number of nodes: N
3. Average path length: s_{av}
4. Weighted sum of the number of paths and the number of nodes: s_w

Number of paths An efficient algorithm to carry out sifting with respect to the number of paths P was introduced in [9]. Paths from roots to terminals in the BDD correspond to paths in the circuit. Therefore minimizing the number of paths in the BDDs leads to a smaller number of possibly erroneous paths in the circuit. By this, less test patterns are needed to test the resulting circuit.

Number of nodes The change of the number of nodes P can efficiently be calculated during sifting. Therefore this objective function leads to a fast minimization algorithm. Since a multiplexor corresponds to each node in the BDD, the size in terms of logic elements in the circuit is minimized by this function.

Average path length Let p_i be the number of paths with length i and P the total number of paths, then the average length of paths s_{av} in the BDD is given by:

$$s_{av} = \frac{1}{P} \sum_{i=1}^{n} i \; p_i$$

The minimization of the average path length leads to circuits with a smaller depth on the paths from inputs to outputs. By this, the circuit is optimized for speed on the one hand and on the other hand the number of very long paths is reduced.

Weighted sum of the number of paths and the number of nodes A weighted objective function has already been used in [16] to optimize for fast evaluation of multi-output functions and with respect to the number of nodes. Here, the weighted function is used to optimize with respect to number of nodes and number of paths. Let the number of paths and nodes before sifting be given by P_{old} and N_{old}, respectively. Furthermore let P_{act} and N_{act} denote the current number of paths and nodes. Then the objective function for the current size s_w of the BDD was chosen as:

$$s_w = \alpha \frac{P_{act}}{P_{old}} \times \beta \frac{N_{act}}{N_{old}}$$

Parameters α and β can be chosen to emphasize testability or size during synthesis.

The following example shows the differences between objective functions N and P.

Example 1. *Consider the BDDs in Figure 2 that were introduced in [8]. Both BDDs represent the function*

$$f(x_1, \ldots, x_5) = x_1 \overline{(x_4 \oplus x_5)} + \overline{x}_2 (x_4 \oplus x_5) + x_1 x_3 + \overline{x}_1 x_2 \overline{x}_3$$

The BDD with respect to variable order $(x_2, x_3, x_4, x_5, x_1)$ *has 12 nodes and 14 paths. The second BDD with respect to the order* $(x_2, x_3, x_4, x_5, x_1)$ *has 11 nodes and 20 paths, i.e. less nodes but more paths.*

Thus, there is no direct correlation between size and number of paths in a BDD [8]. It can also be shown that depending on the variable order the number of paths in a BDD may vary from linear to exponential for the same function [10]. This motivates the use of the weighted objective function.

4. Experimental Results

Combinational benchmarks from the LGSynth93 set were investigated with respect to the proposed objective functions. Selected results are given in Tables 1-3. From the original circuit given as a blif-file the BDD was built.

Table 1. Characteristics of circuits

Circ.	#in	#out	#lits	PDFC	#paths
5xp1	7	10	158	19.9 %	1071
alu2	10	6	411	0.8 %	64428
b9	41	21	171	89.0 %	410
clip	9	5	270	82.1 %	482
count	35	16	159	100.0 %	384
frg1	28	3	219	67.3 %	575
i5	133	66	198	100.0 %	672

Then the minimization was carried out and finally the BDD-circuit was written (for more details see [2]). The number of literals needed for a circuit was determined using SIS [17], also the number of physical paths in the circuit and the path delay fault coverage were determined. In all of the following tables the notations of Section 3 are used to denote objective functions and sizes.

Table 1 shows characteristics of each circuit. Besides the number of primary inputs and outputs the number of literals needed after optimization by 'script.rugged' from SIS is listed. Column 'PDFC' gives the path delay fault coverage of the optimized circuit, i.e. the percentage of paths that can be covered by a RPDF test. For this coverage large variations can be observed, in one case even less than 1% of all paths can be tested ('alu2').

Next, Table 2 states the statistics on the BDDs that have been minimized with respect to the different objective functions. In case of the weighted objective function s_w both parameters α and β were set to 0.5. The lowest number of paths for each circuit is printed bold.

The minimization of a BDD with respect to one criterion does not always lead to the best result with respect to this criterion, e.g. '5xp1' turns out to have the least number of nodes when minimized with respect to paths. This is due to the heuristic minimization algorithms: The objective function influences which parts of the search space are considered. The average path length does not vary too much for the different objective functions. Using it as the only optimization criterion can lead to large BDDs in terms of number of nodes. Much better results are produced when applying the weighted objective function, which often leads to the optimum for P and N within the four strategies. Especially remarkable is 'count'. Here s_w leads to a BDD that has by far less nodes than in all other cases and also the number of paths is the smallest.

Finally, Table 3 gives the data for the circuits derived from the BDDs. Here again, SIS was used to count the number of literals for the circuit. SIS was also used to propagate constants into the gates of the circuit using the command 'sweep'. This can result in differently sized circuits for same sized BDDs (e.g. '5xp1' for methods N and s_{av}). In general the BDD smallest in the number of nodes leads to the smallest circuit.

One of the objective functions P and s_w always led to the circuit with the least number of physical paths. Especially remarkable are the results for 'i5', where the number of physical paths is reduced by a factor of 12. The objective function s_w compared to method N always lead to less physical paths while the size of the circuit on average dropped.

In comparison to SIS in some cases ('5xp1', 'alu2') a significant improvement in the number of paths and in

Table 2. BDD-statistics

Obj. func.	P			N			s_{av}			s_w		
Circ.	N	P	s_{av}	N	P	s_{av}	N	P	s_{av}	N	P	s_{av}
5xp1	44	**161**	4.42	51	180	4.63	51	165	4.40	42	**161**	4.42
alu2	195	**493**	7.59	172	530	7.58	171	509	7.44	168	506	7.47
b9	145	**558**	6.90	121	785	7.48	185	697	6.63	128	616	7.03
clip	75	**428**	6.73	98	626	7.10	117	478	6.73	75	**428**	6.73
count	217	368	8.09	218	1317	9.44	253	605	7.18	81	**352**	8.32
frg1	118	**349**	11.86	110	759	13.13	194	680	10.57	115	**349**	11.86
i5	135	**738**	6.60	326	8008	10.46	482	1348	6.00	134	**738**	6.60

Table 3. Results for the circuits

	P			N			s_{av}			s_w		
Circ.	#lits	PDFC	#paths	#lits	PDFC	#paths	#lits	PDFC	#paths	#lits	PDFC	#paths
5xp1	137	82.1 %	219	172	79.2 %	256	163	82.6 %	225	131	**83.0 %**	218
alu2	696	**85.9 %**	749	624	84.3 %	814	604	81.4 %	776	598	81.3 %	773
b9	420	65.2 %	848	364	62.7 %	1228	564	63.0 %	1095	390	**65.7 %**	934
clip	259	72.6 %	644	357	**78.2 %**	969	414	72.8 %	744	260	72.2 %	648
count	704	**94.0 %**	536	831	67.2 %	2163	837	79.6 %	961	221	58.9 %	624
frg1	326	65.9 %	529	399	75.0 %	1135	649	**77.4 %**	1024	320	65.9 %	529
i5	330	**68.9 %**	975	1022	68.0 %	11671	1462	64.1 %	1997	330	**68.9 %**	975

PDFC was gained, for '5xp1' even the size decreased. Opposed to the large variations of the PDFC obtained for optimized circuits from SIS, the PDFC was never below 50% for any of the BDD circuits.

5. Conclusions and Future Work

Different objective functions for BDD minimization were evaluated with respect to testability of the resulting BDD circuits. It was shown that the reduction of the number of paths in the BDD can significantly reduce the number of test patterns needed to test the circuit with respect to the PDFM. An adjustable objective function was used to take more than one goal during synthesis into account and thereby to compromise e.g. between size and number of test patterns needed.

The major drawback of the current synthesis approach is the drop in the path delay fault coverage for some circuits. Therefore it is focus of current work to combine the proposed technique with the one from [7]. In [7] MUX-circuits are synthesized that are guaranteed to be 100% testable under the PDFM and the Stuck-at Fault Model at the cost of one additional input.

References

[1] P. Ashar, S. Devadas, and K. Keutzer. Path-delay-fault testability properties of multiplexor-based networks. *INTEGRATION, the VLSI Jour.*, 15(1):1–23, 1993.

[2] B. Becker. Synthesis for testability: Binary decision diagrams. In *Symp. on Theoretical Aspects of Comp. Science*, volume 577 of *LNCS*, pages 501–512. Springer Verlag, 1992.

[3] B. Becker. Testing with decision diagrams. *INTEGRATION, the VLSI Jour.*, 26:5–20, 1998.

[4] B. Bollig and I. Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Trans. on Comp.*, 45(9):993–1002, 1996.

[5] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[6] R. Bryant. Binary decision diagrams and beyond: Enabeling techniques for formal verification. In *Int'l Conf. on CAD*, pages 236–243, 1995.

[7] R. Drechsler, J. Shi, and G. Fey. MuTaTe: An efficient design for testability technique for multiplexor based circuits. In *Great Lakes Symp. VLSI*, 2003.

[8] E. Dubrova and D. Miller. On disjoint covers and ROBDD size. In *Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 162–164, 1999.

[9] G. Fey and R. Drechsler. Minimizing the number of paths in BDDs. In *Symposium on Integrated Circuits and System Design*, pages 359–364, 2002.

[10] G. Fey and R. Drechsler. Utilizing BDDs for disjoint SOP minimization. In *Midwest Symposium on Circuits and Systems*, volume 2, pages 306–309, 2002.

[11] W. Günther and R. Drechsler. ACTion: Combining logic synthesis and technology mapping for MUX based FPGAs. *Journal of Systems Architecture*, 46(14):1321–1334, 2000.

[12] A. Kristič and K.-T. Cheng. Resynthesis of combinational circuits for path count reduction and for path delay fault testability. *Jour. of Electronic Testing: Theory and Applications*, 11:43–54, 1997.

[13] I. Pomeranz and S. Reddy. Delay fault models for VLSI circuits. *INTEGRATION, the VLSI Jour.*, 26:21–40, 1998.

[14] I. Pomeranz and S. M. Reddy. Design-for-testability for path delay faults in combinational circuits using test points. *IEEE Trans. on CAD*, 17:333–343, 1998.

[15] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.

[16] C. Scholl, R. Drechsler, and B. Becker. Functional simulation using binary decision diagrams. In *Int'l Conf. on CAD*, pages 8–12, 1997.

[17] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical report, University of Berkeley, 1992.

[18] G. Smith. Model for delay faults based upon paths. In *Int'l Test Conf.*, pages 342–349, 1985.