BDD based Synthesis of Symmetric Functions with Full Path-Delay Fault Testability

Junhao Shi Görschwin Fey Rolf Drechsler

Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
{junhao,fey,drechsle}@informatik.uni-bremen.de

Abstract

A new technique for synthesizing totally symmetric Boolean functions is presented that achieves complete robust path delay fault testability. We apply BDDs for the synthesis of symmetric functions. Only one additional input and one inverter are needed to achieve 100% Path Delay Fault (PDF) testability. The size of the circuit is guaranteed to be at most quadratic in the number of inputs. The test vectors for any PDF can be generated in linear time. Experimental results underline the efficiency of the approach. In contrast to previous approaches, the technique can also be applied to multi-output functions.

1. Introduction

Correct operation of a circuit is a major issue. Therefore circuits are tested for functional and temporal behavior. One very powerful fault model is the *Path Delay Fault Model* (PDFM) that allows to detect static and dynamic faults [7].

Binary Decision Diagrams (BDDs) are known to be an efficient representation of Boolean functions [3]. They can be used for synthesis by mapping the BDDs to a multiplexor circuit. The testability of such circuits has already been investigated in [1, 2]. Here, we propose a technique that provides a structural way for test vector generation in polynomial time.

Symmetric Boolean functions often appear in logic design, and are widely used in cryptology. To synthesize for delay fault testability, several approaches were proposed, such as constrained two-level minimization procedures [9], three- or four-level circuits [12], the unate decomposition [8], or the cellular logic array [5]. The new approach is more efficient, because for symmetric functions of n variables, the size of the circuit is in $O(n^2)$ and test patterns for any single PDF can be generated in $O(n)$. Moreover in contrast to previous approaches the new approach is capable of synthesizing multi-output functions. The approach can also be extended to non-symmetric functions [4], but then no upper bounds on the circuit size and complexity of test pattern generation can be guaranteed.

2. Preliminaries

Let $f(x_1, x_2, \ldots, x_n)$ denote a switching function of n Boolean variables. A minterm is a product of variables in which every variable appears once. The weight w of a minterm m is defined as the number of uncomplemented variables that appear in m. A switching function $f(x_1, x_2, \ldots, x_n)$ is said to be totally symmetric with respect to the variables $(x_1, x_2, \ldots, x_n)$ if it is invariant under any permutation of the variables. Total symmetry can be expressed in terms of a set of integers (called a-numbers [8]) $A = \{a_i, \ldots, a_j, \ldots, a_k\}$, where $A \subseteq \{0, 1, 2, \ldots, n\}$; all the vertices with weight $w \in A$ will appear as true minterms in the function. A n-variable symmetric function is denoted as $S_n(a_i, \ldots, a_j, \ldots, a_k)$. For n variables, $2^{n+1} - 2$ totally symmetric functions (excluding constant functions 0 and 1) can be constructed.

A combinational circuit implementing a function f can be retrieved from a BDD by replacing each node of the function with a multiplexor. This multiplexor 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 (see [2]).

3. BDD based Synthesis of Symmetric Functions

3.1. BDDs for Symmetric Functions

For a totally symmetric function it is well known that the size of the BDD is bounded by $O(n^2)$. This is due to the observation that for functions symmetric in (x_i, x_j) the equation $f_{x_i \overline{x}_j} = f_{\overline{x}_i x_j}$ holds. For BDDs this implies that for two symmetric variables the left son of the right son of the root is the right son of the left son of the root. Thus, BDDs representing totally symmetric functions grow in each level at most by one node. At the end, there are $n + 1$ terminal vertices $0, 1, \ldots, n$. An n-variables symmetric function is denoted as $S_n(a_i, \ldots, a_j, \ldots, a_k)$. So the terminal vertices $a_i, \ldots, a_j, \ldots, a_k$ are 1, and all others are 0. BDDs are reduced by application of reduction rules [3].

3.2. BDD Transformation

Analogously to the "standard approach" from [2] the circuit is generated by traversing the BDD and substituting each node with a MUX cell. But, the methods differ when reaching nodes that have one or two pointers to terminal nodes. In this case, usually the MUX cell is simplified. E.g. if the 0-input is connected to constant 0, the MUX cell can be simplified and can be substituted by an AND gate.

Here, all nodes - also the ones pointing to terminals - are substituted by complete multiplexor cells. The terminal node 0 is then substituted by a new primary input t (=test). Furthermore, t is connected to the 1-terminal of the BDD by an inverter.

Example 1. *If the approach from [2] is applied to the function $S_2(1, 2)$, the BDD circuit in Figure 1(a) results (shown without simplification). While the transformation described above generates the circuit in Figure 1(b).*

If t is set to constant 0, the circuit computes the original function. If t is set to 1, the complement is computed. It is important to observe, that by changing the value of t all "internal" signals, i.e. signals corresponding to edges in the BDD, change their value (see [4]).

Table 1. Path Delay Fault coverage of the circuit

name	*in*	*out*	original			optimized			BDD method		
			NoP	lits	PDFC	NoP	lits	PDFC	NoP	lits	PDFC
$S_{10}(4,5)$	10	1	1252	220	63	995	103	50	1366	133	100
$S_{11}(3,4,5,6,7)$	11	1	3716	264	19.1	533	127	92.6	1543	137	100
$S_{12}(6,7)$	12	1	4717	312	63.6	2634	149	53.8	5245	189	100
$S_{13}(7,8)$	13	1	8435	364	64.3	12966	183	25.9	9667	217	100
$S_{14}(6,7)$	14	1	17873	420	63.9	9216	213	48.1	20161	253	100
$S_5(1,2)$	5	1	39	60	61.5	37	25	67.5	43	33	100
$S_6(3,4)$	6	1	89	84	60.6	105	45	52.3	91	45	100
$S_7(2,3)$	7	1	152	112	63.1	191	65	58.1	169	61	100
$S_8(3,4)$	8	1	334	144	62.2	403	75	41.4	355	85	100
$S_9(5,6)$	9	1	580	180	63.7	472	90	51.5	652	105	100
rd53	5	3	144	182	97.2	155	39	48.7	123	61	100
rd73	7	3	840	741	97.3	681	220	49.4	528	116	100
rd84	8	4	3288	1442	55.7	1211	380	57.2	1066	163	100
9sym	9	1	522	655	96.5	518	333	91.1	490	93	100
9symml	9	1	276	381	97.2	519	309	72.3	490	93	100

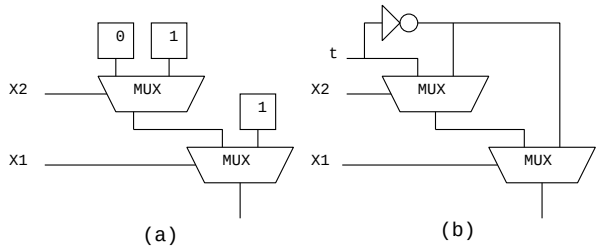

Figure 1. Example for transformation

3.3. Path Delay Fault Testability

Using the described transformation, full PDF-testability can be obtained for any Boolean function and the test pattern for a PDF can be generated in linear time.

Theorem 1. *By one additional input and one inverter a circuit can be generated from a BDD that is 100% testable for robust Path Delay Faults.*

Theorem 2. *In the resulting circuit for a symmetric function with n inputs, test pattern generation can be carried out in time $O(n)$ for any single PDF.*

4. Experimental Results

The technique described above has been implemented using CUDD as the underlying BDD package [11] and all experiments have been carried out on a SUN Sparc 20 with 64 Mbyte of main memory.

As benchmarks $S_n(a_i,\ldots,a_j)$ were generated as circuits in blif-format. Also other totally symmetric benchmarks were taken from the LGSynth91 [13]. For each circuit we report the number of literals(lits) (measured using SIS[10]), the number of paths (NoP) that have to be tested and the PDF coverage (PDFC) of the circuits in percent.

In Table 1 the name of the benchmark is given in the first column followed by the number of inputs and outputs in column two and three, respectively. Column "original" gives numbers for the original circuits as given by the blif-description. Column "optimized" gives the numbers for the circuits that have been optimized by SIS using *script. rugged*. As can be seen, the PDFC for the SIS circuits varies from 25.9% to 92.6%. Column "BDD method" gives the results for the approach described in this paper, i.e. the new testing input is connected to each constant input to a MUX cell and the MUX cell with both constant inputs is substituted by a simple wire or an inverter. As can be seen, the synthesis method is suited not only to one output symmetric functions but also to the symmetric circuits in LGSynth91. For all the functions, 100% PDFC is ensured. In case of the generated circuits, the BDD method is in the same order of size as SIS, but guarantees testability. For most benchmarks from LGSynth91, the size is even smaller. E.g. for rd84 and 9sym the BDD circuit is much smaller than the corresponding circuit produced by SIS.

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

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

[5] D. H.Rahaman and B.B.Bhattacharya. A simple delay-testable design of digital summation threshold logic (dstl) array. *Proc. of the 5th International Workshop on Boolean Problems*, pages 189–194, 2002.

[6] S. N.K.Jha, I.Pomeranz and R.J.Miller. Synthesis of miltilevel combinational circuits for complete robust path delay fault testability. *Proc.Int.Symp. Fault Tolerant Computing*, pages 280–287, 1992.

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

[8] D. S.Chakraborty, S.Das and B.B.Bhattacharya. Synthesis of symmetric functions for path-delay fault testability. *IEEE Trans. Computer-Aided Design*, 19:1076–1081, 2000.

[9] S.Devadas and K. Keutzer. Synthesis of robust delay-fault-testable circuits: Practice. *IEEE Trans. Computer-Aided Design*, 11:227–300, 1992.

[10] 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.

[11] F. Somenzi. *CUDD: CU Decision Diagram Package Release 2.3.1*. University of Colorado at Boulder, 2001.

[12] W.Ke and P.R.Menon. Delay-testable implementations of symmetric fuctions. *IEEE Trans. Computer-Aided Design*, 14:772–775, 1995.

[13] S. Yang. Logic synthesis and optimization benchmarks user guide. Technical Report 1/95, Microelectronic Center of North Carolina, 1991.