# 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, 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. BDDs are used for the synthesis step. 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 technique. 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 behaviour. One very powerful fault model is the *Path Delay Fault Model* (PDFM) that allows to detect static and dynamic faults [8].

*Binary Decision Diagrams* (BDDs) are known to be an efficient representation of Boolean functions [4]. They can be used for synthesis by mapping the BDDs to a multiplexor circuit. The testability of such circuits has already been investigated in [2, 1, 3]. 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 [18], and are widely used in cryptology [5, 17]. To synthesize for delay fault testability, several approaches were proposed, such as constrained two-level minimization procedures [11, 10], three- or four-level circuits [15], the unate decomposition [9], or the cellular logic array [7].

The approach presented in this paper 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.

## 2. Preliminaries

### 2.1. Symmetric Functions and BDD Circuits

Let $f(x_1, x_2, \dots, x_n)$ 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 $f(x_1, x_2, \dots, x_n)$ is said to be totally symmetric with respect to the variables $(x_1, x_2, \dots, x_n)$ if it is invariant under any permutation of the variables [18]. Total symmetry can be expressed in terms of a set of integers (called a-numbers [9]) $A = \{a_i, \dots, a_j, \dots, a_k\}$, where $A \subseteq \{0, 1, 2, \dots, n\}$. All the vertices with weight $w \in A$ will appear as true minterms in the function. An $n$-variable symmetric function is denoted as $S_n(a_i, \dots, a_j, \dots, a_k)$. For $n$ variables, $2^{n+1}$ totally symmetric functions exist.

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 Figure 1).

### 2.2. Path Delay Fault

In this paper we consider a dynamic fault model, i.e. the *Path Delay Fault Model* (PDFM) [13]. 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 1.** *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 PDF 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 2.** *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.*

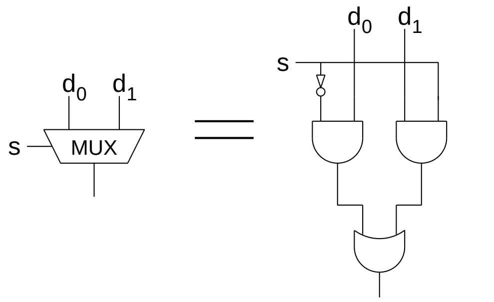


**Figure 1. Implementation of a multiplexor**

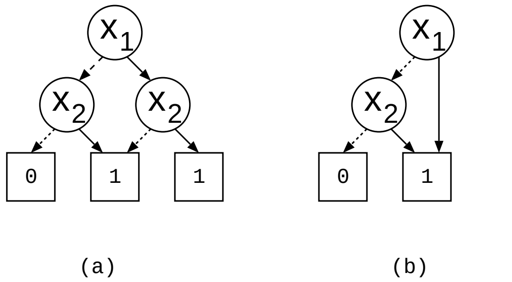


**Figure 2. BDD for symmetric function** $S_2(1,2)$

# 3. BDD based Synthesis

## 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 node is the right son of the left son of the node. 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.

**Example 1.** *In Figure 2(a), a BDD for the symmetric function $S_2(1,2) = x_1 + x_2$ is shown. The low-edges of the nodes are given as dotted lines. If the reduction rules from [4] are applied, the BDD in Figure 2(b) results.*

## 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 by substituting an AND gate.

Here, all nodes - also the ones pointing to terminals - are substituted by complete multiplexor cells. The terminal node 0 is then replaced by a new primary input $t$ (=test).

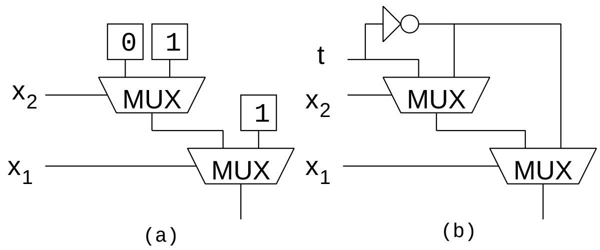


**Figure 3. Example for transformation**

Furthermore, $t$ is connected to the 1-terminal of the BDD by an inverter.

**Example 2.** *In Figure 2(b), the reduced BDD for the symmetric function $S_2(1,2)$ from Example 1 is shown. If the approach from [2] is applied, the BDD circuit in Figure 3(a) results (shown without simplification). While the transformation described above generates the circuit in Figure 3(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 [6]). This guarantees a simple application of the values needed at the fault location.

The simplification of the MUX cells as in [2] can destroy the testability. But not all types of simplifications have this property, e.g. if both data inputs have constant values the MUX cell can be substituted by a simple wire or an inverter (see [6]).

## 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.*

*Proof.* Any path from a primary input to a primary output consists of multiplexors. In the following it is shown, that the propagation of the output of any multiplexor to a primary output is possible and that all paths within a single multiplexor are testable. As a result any path is testable.

At each node the propagated path is selected by a primary input. Due to the read-once property of BDDs each primary input only influences one node on a path. Therefore, propagation is no problem.

Now, consider an arbitrary multiplexor node internal to the circuit. At the data inputs $(d_0, d_1)$ at least one of the patterns $p_0 = (0,1)$ or $p_1 = (1,0)$ is applicable. This holds, because the node corresponds to a node in the BDD. Let this node be labelled by variable $x$ and represent the function $f$. Then, its children are the positive $(f_x)$ and negative cofactors $(f_{\overline{x}})$ of $f$. If neither $p_0$ nor $p_1$ is applicable, this implies $f_x \equiv f_{\overline{x}}$ and this is a contradiction because the BDD is reduced. If only $p_0$ $(p_1)$ is applicable, by inverting the terminals (i.e. setting $t$ to 1) all internal signals change their values, i.e. the opposite pattern is applied.

Having $p_0$ and $p_1$ applicable to the data inputs of a multiplexor implies full testability with respect to the PDFM [3]

**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 |

(see the realization of a multiplexer in Figure 1). There are four paths from inputs to the output that have to be tested: One from each input $d_0$ and $d_1$, and two paths from the select input $s$. Now testing the path along $d_0$ with respect to a rising transition is done by setting $s$ to 0, applying $p_0$ and switching to $p_1$. The falling transition and testing $d_1$ are analog cases. Testing the non-inverted path starting at $s$ is done by applying $p_0$ and switching $s$, the inverted path is tested using $p_1$.

A similar argumentation holds for paths starting at the test input $t$.

All these tests are independent from other PDFs, because there is only one path at a time propagating values. A change of the signal values is propagated on the path under test only. ∎

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

*Proof.* Let the BDD circuit be based on multiplexors. Due to the structure of the circuit, a propagation path from the fault location to the output can be generated in linear time. This is done by setting the select inputs of the following multiplexors, such that the current output is selected.

Let $v$ be any node in the BDD. Let the node be labelled by variable $x_i$ and represent a function $f$. If $f$ is not a constant, let $x_i = 1$ and check $f_{x_i}$. If $f_{x_i} \neq constant$, set $x_{i+1}, x_{i+2}, \ldots = 1$, until $f_{x_i \ldots x_j} = b \in \{0,1\}$. This determines the value of the $d_1$-input to the MUX. The input value at $d_0$ is given by $f_{\overline{x}_i x_{i+1} \ldots x_j}$. Now we have the following three cases.

1. $f_{\overline{x}_i, x_{i+1}, \ldots, x_j} = \overline{b}$. The test path has been found in time $O(n)$.

2. $f_{\overline{x}_i, x_{i+1} \ldots, x_j} \neq constant$. Set $x_{j+1}, \ldots, x_k = 0$ until $f_{\overline{x}_i, x_{i+1} \ldots, x_j, \overline{x}_{j+1} \ldots, \overline{x}_k} = c \in \{0,1\}$. Because $f$ is a symmetric function, $c = \overline{b}$. This can be seen as follows:

$$\begin{array}{rcll} & & f_{\overline{x}_i, x_{i+1} \ldots, x_j, \overline{x}_{j+1} \ldots, \overline{x}_{k-1}, x_k} & \\ & = & f_{x_i, \ldots, x_j, \overline{x}_{j+1} \ldots, \overline{x}_k} & \text{(symmetry)} \\ & = & f_{x_i, x_{i+1} \ldots, x_j} & \\ & = & b & \\ & \Rightarrow & & \\ c & = & f_{\overline{x}_i, x_{i+1} \ldots, x_j, \overline{x}_{j+1} \ldots, \overline{x}_{k-1}, \overline{x}_k} & \\ & = & \overline{b} & \text{(reduction)} \end{array}$$

   Otherwise the node would have been reduced.

   The test vector has been found in time $O(n)$.

3. $f_{\overline{x}_i, x_{i+1} \ldots, x_j} = b$. There is $x_l$: $f(\overline{x}_i, \ldots, x_l) = b$ and $f(x_i, \ldots, x_l) \neq constant$. This case is analog to 2. The test vector has been found in time $O(n)$.

The vector to apply $(b, \overline{b})$ at the multiplexor is found. In the circuit, the terminals are connected to $t$, so the values can be inverted by switching $t$. All test vectors are found in linear time. ∎

## 4. Experimental Results

The technique described above has been implemented using CUDD as the underlying BDD package [14] 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 [16]. For each circuit we report the number of literals (lits) measured using SIS[12], 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" reports the numbers for the original circuits as given by the blif-description. Column "optimized" gives the numbers for

**Table 2. Comparison with earlier works**

| *name* | test cost (NoP) | | | | circuit cost (lits) | | | |
|---|---|---|---|---|---|---|---|---|
| | As in [15] | As in [9] | As in [7] | BDD method | As in [15] | As in [9] | As in [7] | BDD method |
| $S_5(1,2)$ | 35 | 23 | 24 | 43 | 47 | 32 | 42 | 33 |
| $S_6(3,4)$ | 90 | 53 | 54 | 91 | 112 | 73 | 62 | 45 |
| $S_7(2,3)$ | 182 | 102 | 96 | 169 | 219 | 138 | 86 | 61 |
| $S_8(3,4)$ | 448 | 236 | 208 | 355 | 520 | 306 | 114 | 85 |
| $S_9(5,6)$ | 882 | 442 | 369 | 652 | 1010 | 566 | 146 | 105 |
| $S_{10}(4,5)$ | 2100 | 1034 | 781 | 1366 | 2354 | 1296 | 182 | 133 |
| $S_{11}(3,4,5,6,7)$ | 1815 | 916 | 713 | 1543 | 2147 | 1108 | 222 | 137 |
| $S_{12}(6,7)$ | 9504 | 4469 | 3001 | 5245 | 10430 | 5463 | 264 | 189 |
| $S_{13}(7,8)$ | 18447 | 8451 | 5422 | 9667 | 20165 | 10261 | 314 | 217 |
| $S_{14}(6,7)$ | 42042 | 19081 | 11469 | 20161 | 45476 | 22877 | 366 | 253 |

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 not only suitable for functions with one output. For all the functions 100% PDFC is ensured. In case of the generated circuits, regarding the size, the BDD method is in the same order 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.

Table 2 presents the comparison to other methods [15, 9, 7] for various examples of consecutive symmetric functions. In all those methods 100% PDFC is ensured. It can be seen that our approach produces by far the smallest circuits. E.g. for $S_{14}(6,7)$, the circuit size is nearly a factor of 100 smaller in comparison to [15, 9]. Moreover the new approach guarantees linear time test pattern generation for a PDF.

## 5. Conclusions

In this paper a new approach to synthesis of symmetric functions using BDDs has been described. This method guarantees complete RPDF testability for all symmetric functions, and yields very low circuit cost compared to earlier approaches. Test patterns for a single PDF can be generated in linear time. The approach can also be extended to non-symmetric functions [6], but then no upper bounds on the circuit size and complexity of test pattern generation can be guaranteed.

## Acknowledgement

The authors like to thank Debesh K. Das for helpful discussions on this topic.

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

[5] C.Ding, G.Xiao, and W.Shan. *The Stability Theory of stream Ciphers*. Berlin: Spinger-Verlag LNCS, 1991.

[6] R. Drechsler, J. Shi, and G. Fey. MuTaTe: An efficient design for testability technique for multiplexor based circuits. In *Great Lakes Symp. VLSI*, pages 80–83, 2003.

[7] 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.

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

[9] 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.

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

[11] S.Devadas and K. Keutzer. Synthesis of robust delay-fault-testable circuits: Theory. *IEEE Trans. Computer-Aided Design*, 11:87–101, 1992.

[12] 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.

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

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

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

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

[17] Y.X.Yang and B.Guo. Further enumerating boolean functions of cryptographic significance. *J. Cryptol.*, 8:115–122, 1995.

[18] Z.Kohavi. *Switching and Finite Automata Theory*. New York: McGraw Hill, 1977.