MuTaTe: An Efficient Design for Testability Technique for Multiplexor based Circuits

Rolf Drechsler

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

Abstract

We present a technique to derive fully testable circuits under the Stuck-At Fault Model (SAFM) and the Path-Delay Fault Model (PDFM). Starting from a function description as a Binary Decision Diagram (BDD) the netlist is generated by a linear time mapping algorithm. Only one additional input and one inverter are needed to achieve 100% testable circuits under SAFM and PDFM. Experiments show the advantages of the technique in comparison to previously presented methods.

1 Introduction

Since *Binary Decision Diagrams* (BDDs) have been proposed in [14, 1, 7] several applications have been studied, where this data structure can be successfully applied. In formal verification BDDs have become the state-of-the-art for function representation and manipulation. BDDs have also been studied in logic synthesis, since they allow to combine aspects of circuit synthesis and technology mapping [11]. Recently, there is a renewed interest in multiplexor based design styles, since often multiplexor nodes can be realized at very low cost (as e.g. *Pass Transistor Logic* (PTL)). In addition, these techniques allow to consider layout aspects during the synthesis step and by this guarantee high design quality [8, 9, 16, 15].

One of the most important steps during circuit design is the testability of the netlist. Multiplexor circuits derived from BDDs have been studied intensively under various fault models [3, 2]. (For an overview see [4].) But non of these approaches can guarantee 100% testability in a "systematic way". E.g. in [3] an algorithm is given that can compute all redundancies of the circuit in polynomial time. But the removal of these redundancies can generate new ones (so-called 2nd-generation redundancies). For their removal only classical ATPG can be applied.

In this paper, a simple transformation is presented that guarantees full testability of a circuit derived from a BDD description under the stuck-at fault model and the robust path-delay fault model. The size of the circuit is directly proportional to the given BDD size and only one extra input and one inverter are needed. The resulting circuits are free of redundancies. The algorithm has been implemented as a tool for *Multiplexor Transformation for Testabilty* (MuTaTe). Experimental results are given that show the advantages of the approach compared to "classical" mapping of BDDs.

The paper is structured as follows: In Section 2 BDDs and BDD circuits are defined and the fault models considered are briefly reviewed. The transformation algorithm is described in Section 3. Experimentals are given in Section 4. Finally, the results are summarized.

2 Preliminaries

2.1 Binary Decision Diagrams

As is well-known a Boolean function $f : \mathbf{B}^n \rightarrow \mathbf{B}$ can be represented by a *Binary Decision Diagram* (BDD) which is a directed acyclic graph where a Shannon decomposition

$$f = \overline{x}_i f_{low(v)} + x_i f_{high(v)} \quad (1 \leq i \leq n)$$

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 not contain isomorphic subgraphs nor does it have redundant nodes. Reduced and ordered BDDs are a canonical representation since for each Boolean function

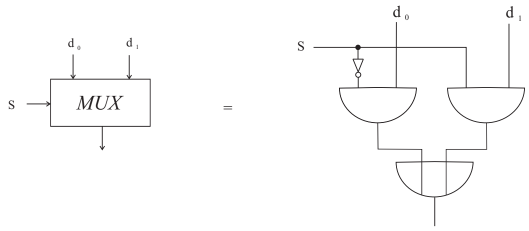

Figure 1. Multiplexor cell *MUX*

the BDD is uniquely specified [7]. In the following, we refer to reduced and ordered BDDs for brevity as BDDs.

2.2 BDD Circuits

BDDs directly correspond to multiplexor based Boolean circuits, called *BDD circuits* in this paper. More exactly: BDD circuits are combinational logic circuits defined over a fixed library. The typical multiplexor cell is denoted as MUX, and it is defined as given in Figure 1 by its standard *AND-, OR-, INVERTER*-based realization[1]. The left input is called *control input*, the upper inputs are called *data inputs* (left data input = *0-input*, right data input = *1-input*).

The *BDD circuit* of a BDD is now obtained by the following construction: Traverse the BDD in topological order and replace each non-terminal node v in the BDD by a *MUX* cell, connect the control input with the primary input x_i, corresponding to the label of the BDD node. Then, connect the 0-input to $low(v)$, the 1-input to $high(v)$. At the end connect the output of the multiplexor which substituted the root node with a primary output.

Remark 1 *As has been suggested in previous papers [3, 2], the MUX cells connected to constant values can be simplified. But in our approach we do not make use of this, since - as will be shown later - this "destroys" the testability.*

2.3 Fault Models

We consider a static and a dynamic fault model, i.e. the *Stuck-At Fault Model* (SAFM) [5] and the *Path-Delay Fault Model* (PDFM) [19].

A fault in SAFM causes exactly one input or output pin of a node in the circuit to have a fixed constant value (0 or 1) independently of the values applied to the inputs of the circuit.

In the PDFM it is checked whether the propagation delays of all paths in a given circuit are less than the system clock interval. For the detection of a path delay fault a pair of patterns (I_1, I_2) is required rather than a single pattern as in SAFM: 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 C are controlled.

[1]The results in the following also transfer to different realizations.

Definition 1 *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 this path.*

It turns out that for the circuits considered in this paper the construction of tests with the following (even stronger) property is possible: For each path delay fault there exists a robust test (I_1, I_2) which sets all off-path inputs to non-controlling values on application of I_1 and remains stable during application of I_2, i.e. the values on the off-path inputs are not invalidated by hazards or races[2]. Robust tests with the properties mentioned above are also called *strong RPDF tests.* In the following we only use such tests, but for simplicity we call them RPDF tests, too. For a detailed classification of PDFs see [17].

3 BDD Transformation

In this section we first describe the transformation how to derive a circuit from a given BDD description. Then, some properties of the resulting circuits are discussed. In the next sections testability properties regarding the SAFM and the PDFM are studied.

Analogously to the "standard approach" from [3] 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 *In Figure 2(a) a BDD for function $f = x_1x_2 + x_3$ is shown. It is drawn upside down to underline the similarities to the resulting circuit. The low-edges of the nodes are given as dotted lines. If the approach from [3] is applied, the BDD circuit in Figure 2(b) results (shown without simplification). While the transformation described above generates the circuit in 2(c).*

[2]A *controlling* value at the input of a node is the value that completely determines the value at the output, e.g. 1 (0) is the controlling value for *OR* (*AND*) and 0 (1) is the non-controlling value for *OR* (*AND*).

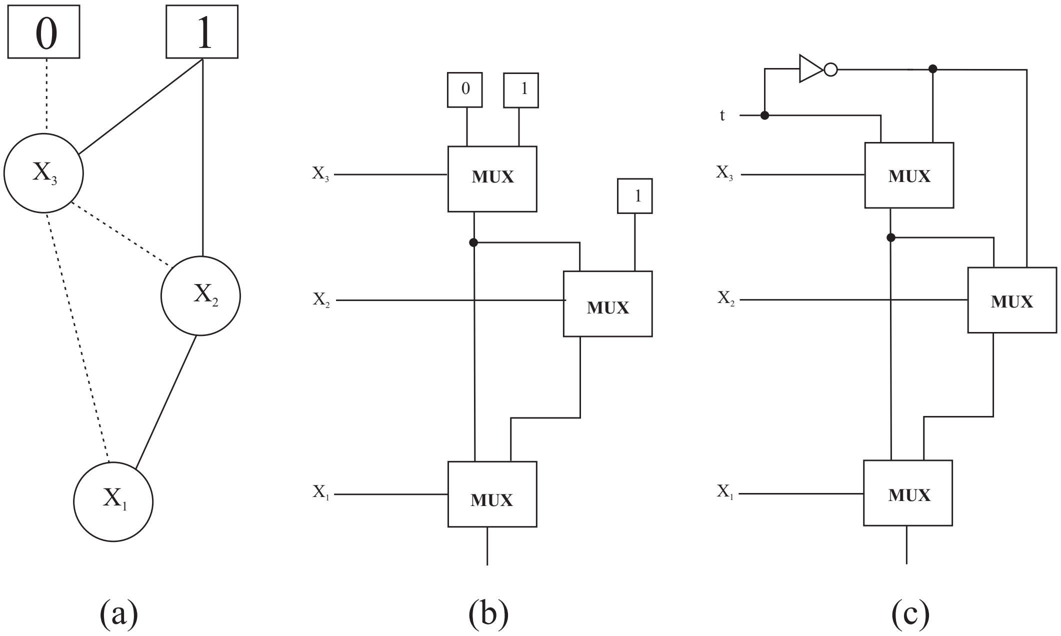

Figure 2. Example for transformation

Remark 2 *It is important to notice that for mutliplexor based design styles, like e.g. PTL, the "simplification" of the MUX cell does not really imply savings in area or delay, since the complete multiplexor cell is often easier to realize.*

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. This can be seen as follows:

$$\begin{aligned} \overline{f} &= \overline{\overline{x}_i g + x_i h} \\ &= (\overline{\overline{x}_i g})(\overline{x_i h}) \\ &= (x_i + \overline{g})(\overline{x}_i + \overline{h}) \\ &= \overline{x}_i \overline{g} + x_i \overline{h} \end{aligned}$$

Applying this recursively to the BDD or the BDD circuit, respectively, shows that all signals change their value. This guarantees a simple application of the values needed at the fault location.

Furthermore, the propagation of the faulty behavior to an output has to be ensured. This is one of the reasons, why circuits based on multiplexor cells became very popular, since due to the control input a propagating path can easily be generated. Thus, the propagation of a value from a fault location is no problem and the only thing to be done is to apply the value that shows the faulty behavior.

In previous approaches (see e.g. [3, 2]), modifications of the circuit were described, but these change the multiplexor structure and by this also destroy the propagation properties of multiplexors.

3.1 Stuck-At Fault Model

As has been observed in [3] stuck-at redundancies in a mapped BDD circuit can only occur if one of the values 01 or 10 is not applicable to the data inputs of this cell. On the other hand, a straightforward computation shows that at least one of the values is applicable. (A detailed analysis is left out due to page limitation.) But due to the properties of the new input t, i.e. all internal signals change their value, the missing value can be applied by changing the value at t. We obtain:

Theorem 1 *By one additional input and one inverter a circuit can be generated from a BDD that is 100% testable for single stuck-at faults.*

For the generation of a test, the efficient (polynomially bounded) synthesis operations on BDDs can be used [7]. For each multiplexor cell the set of applicable values can easily be computed by carrying out AND-operations on the corresponding BDD nodes. The propagating path can be determined by a linear time graph traversal.

Lemma 1 *In the resulting circuits, test pattern generation for stuck-at faults can be carried out in polynomial time.*

3.2 Path-Delay Fault Model

The same arguments as given above also ensure that all paths are testable under the PDFM. At each cell the values 10 or 01 can be applied (dependent on t). Thus, the paths starting at an input corresponding to the variable x_i can be propagated along any of the two AND-gates (see Figure 1). Furthermore, due to the propagation along the multiplexors, it is easy to see that the paths starting at t can be tested. We obtain:

Theorem 2 *By one additional input and one inverter a circuit can be generated from a BDD that is 100% testable for robust path-delay faults.*

After the applicable values have been determined based on BDD operations, two patterns for a robust test can be determined by a traversal of the circuit in linear time.

Lemma 2 *In the resulting circuits, test pattern generation for path-delay faults can be carried out in polynomial time.*

4 Experimental Results

The technique described above has been implemented as the tool MuTaTe (*Multiplexor Transformation for Testabilty*). The program is implemented in C and all experiments have been carried out on a SUN Ultra 1 with 256 MByte of main memory. For the experiments we used the combinational parts of the ISCAS89 benchmarks [6]. Some statistical information about the circuits is given in Table 1. We compare the approach of mapping multiplexors (incl. simplification) [3] to MuTaTe. For the BDD construction we always used the initial variable order as it appears in the benchmark description. Of course, all kinds of minimization techniques for the ordering, like e.g. dynamic variable reordering [18, 10], can be applied to BDDs, since the results in Section 3.1 and 3.2 are independent of the permutation of the variables.

In a first series of experiments we study the random pattern testability under the SAFM. To each circuit we apply 10, 100, 1000 and 10000 random patterns, respectively. The resulting fault coverage is given in Table 2. The coverage obtained by MuTaTe is higher in most cases and often 100% can be achieved. Since the circuits are modeled over basic gates, like AND and OR, more faults have to be considered for MuTaTe. But the test generation runs very fast due to the properties described above.

To get a better understanding of the properties of the generated circuits, we generated exact values for the number of redundancies (*red*), the minimal fault detection probability (*min*), the expected values (*exp val*), and the standard deviation (*std dev*) using the methods from [12]. The results are given in Table 3. As can be seen, the mapped multiplexor circuits contain up to a few hundred redundancies, while the circuits generated by MuTaTe are always fully testable. Furthermore, the circuits are often easier to test by random patterns (see e.g. *s00838*).

In a second series of experiments the PDF testability has been studied. For each circuit the coverage for RPDFs has been determined (see Table 4). As can be seen, circuits generated by MuTaTe always have 100% coverage, while the alternative approach often has less than 70%.

Table 1. Benchmarks

name	*in*	*out*	*signals*	*gates*
s00027	7	4	17	21
s00208	18	9	122	131
s00298	17	20	136	156
s00344	24	26	184	210
s00400	24	27	186	213
s00444	24	27	205	232
s00510	25	13	236	249
s00641	54	43	433	476
s00713	54	42	447	489
s00820	23	24	312	336
s00832	23	24	310	334
s00838	66	33	512	545
s00953	45	52	440	492
s01238	32	32	540	572
s01488	14	25	667	692
s01494	14	25	661	686

5 Conclusions

A new approach to generate multiplexor circuits from an initial BDD description has been described. The resulting circuits are fully testable under the stuck-at fault model and under the (robust) path-delay fault model. The transformation only needs one extra input and one inverter. The algorithm has been implemented as the program MuTaTe. Experimental studies have demonstrated the advantages of the approach.

Acknowledgement

The author likes to thank Bernd Becker for making the test tools of the *test_circ* library [13] available.

References

[1] S.B. Akers. Binary decision diagrams. *IEEE Trans. on Comp.*, 27:509–516, 1978.

Table 2. Random pattern testability

name	MUX-map				MuTaTe			
	10	100	1000	10000	10	100	1000	10000
s00027	84.0909	97.7273	97.7273	97.7273	58.6735	98.4694	100	100
s00208	40.625	68.2292	89.8438	95.8333	36.9485	71.3235	94.4853	99.8162
s00298	54.0948	90.7328	94.8276	94.8276	48.3184	91.9283	99.5516	100
s00344	53.0263	90.2632	98.1579	98.4211	41.8813	86.7301	98.3763	100
s00400	42.1965	76.0116	92.8709	95.9538	51.4024	87.0122	99.5732	100
s00444	53.5194	85.1942	96.6019	98.1796	51.746	90.4762	99.4444	100
s00510	41.9897	82.9457	96.3824	96.3824	37.6414	79.1599	99.1922	100
s00641	17.3562	40.8978	65.9495	81.6715	11.5536	35.1302	63.0224	82.1926
s00713	17.2878	40.8489	65.9213	81.6563	11.5536	35.1302	63.0224	82.1926
s00820	20.3252	60.4065	87.9675	95.2846	28.6924	65.9454	91.8627	98.8711
s00832	20.3252	60.4065	87.9675	95.2846	28.6924	65.9454	91.8627	98.8711
s00838	12.5127	26.58	37.0795	46.8145	19.7853	32.3181	43.4049	52.8922
s00953	23.5267	51.0129	79.7422	90.6998	28.9037	56.423	81.1739	96.4563
s01238	10.6222	35.2556	65.9778	85.2778	9.14175	35.2446	72.0939	92.165
s01488	31.9007	74.2598	95.7498	98.5673	28.5264	72.7645	96.8829	99.9685
s01494	31.9007	74.2598	95.7498	98.5673	28.5264	72.7645	96.8829	99.9685

Table 3. Fault classification

name	MUX-map				MuTaTe			
	red	*min*	*exp val*	*std dev*	*red*	*min*	*exp val*	*std dev*
s00027	1	0.03125	0.25071	0.226047	0	0.0078125	0.122768	0.133397
s00208	13	0.00012207	0.127295	0.184128	0	0.000244141	0.109557	0.146066
s00298	24	0.00390625	0.150071	0.17892	0	0.00390625	0.114715	0.132183
s00344	12	0.000732422	0.125115	0.162438	0	0.000366211	0.0887699	0.118504
s00400	42	0.000427246	0.0864746	0.149812	0	0.00128174	0.109474	0.124793
s00444	15	0.000427246	0.106209	0.151281	0	0.00149536	0.109416	0.124524
s00510	28	0.00195312	0.0868636	0.133419	0	0.00195312	0.0722072	0.10518
s00641	551	2.23517e-08	0.0472366	0.117848	0	2.23517e-08	0.0209178	0.0584455
s00713	551	2.23517e-08	0.0468617	0.116881	0	2.23517e-08	0.0209178	0.0584455
s00820	53	6.10352e-05	0.0572657	0.15185	0	6.10352e-05	0.0554375	0.0997485
s00832	53	6.10352e-05	0.0572657	0.15185	0	6.10352e-05	0.0554375	0.0997485
s00838	61	1.13687e-13	0.0450966	0.118233	0	1.45519e-11	0.0485575	0.109915
s00953	155	1.52588e-05	0.0568674	0.146001	0	3.8147e-06	0.0606365	0.117068
s01238	906	3.8147e-06	0.0202771	0.0706457	0	1.19209e-07	0.014555	0.0468527
s01488	30	0.000488281	0.060589	0.120934	0	0.000244141	0.047214	0.083452
s01494	30	0.000488281	0.060589	0.120934	0	0.000244141	0.047214	0.083452

Table 4. Robust path delay fault testability

name	*MUX-map*	*MuTaTe*
s00027	91.9	100.0
s00208	73.0	100.0
s00298	89.0	100.0
s00344	84.6	100.0
s00400	86.3	100.0
s00444	92.6	100.0
s00510	91.3	100.0
s00641	66.1	100.0
s00713	66.1	100.0
s00820	85.9	100.0
s00832	85.9	100.0
s00838	62.6	100.0
s00953	78.2	100.0
s01238	53.6	100.0
s01488	95.3	100.0
s01494	95.3	100.0

[2] 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.

[3] 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.

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

[5] M.A. Breuer and A.D. Friedman. *Diagnosis & reliable design of digital systems*. Computer Science Press, 1976.

[6] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Int'l Symp. Circ. and Systems*, pages 1929–1934, 1989.

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

[8] P. Buch, A. Narayan, A.R. Newton, and A.L. Sangiovanni-Vincentelli. Logic synthesis for large pass transistor circuits. In *Int'l Conf. on CAD*, pages 663–670, 1997.

[9] F. Ferrandi, A. Macii, E. Macii, M. Poncino, R. Scarsi, and F. Somenzi. Symbolic algorithms for layout-oriented synthesis of pass transistor logic circuits. In *Int'l Conf. on CAD*, pages 235–241, 1998.

[10] G. Fey, J. Shi, and R. Drechsler. BDD circuit optimization for path delay fault-testability. In *GI/ITG Workshop "Testmethods and Reliability of Circuits and Systems "*, 2003.

[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] R. Krieger, B. Becker, and R. Sinković. A BDD-based algorithm for computation of exact fault detection probabilities. In *Int'l Symp. on Fault-Tolerant Comp.*, pages 186–195, 1993.

[13] R. Krieger, R. Hahn, and B. Becker. test_circ: Ein abstrakter Datentyp zur Repräsentation von hierarchischen Schaltkreisen. Technical report, Universität Frankfurt, Fachbereich Informatik, 1993.

[14] C.Y. Lee. Representation of switching circuits by binary decision diagrams. *Bell System Technical Jour.*, 38:985–999, 1959.

[15] L. Macchiarulo, L. Benini, and E. Macii. On-the-fly layout generation for PTL macrocells. In *Design, Automation and Test in Europe*, pages 546–551, 2001.

[16] A. Mukherjee, R. Sudhakar, M. Marek-Sadowska, and S. Long. Wave steering in YADDs: A novel non-iterative synthesis and layout technique. In *Design Automation Conf.*, pages 466–471, 1999.

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

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

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