Experimental Studies on Test Pattern Generation for BDD Circuits

Junhao Shi Görschwin Fey Rolf Drechsler

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

Abstract

Synthesis for Testability has become a major issue as the size and complexity of circuits and systems is rapidly increasing. Due to their good testability design styles based on multiplexors have become very popular. Starting from a function description as a Binary Decision Diagram (BDD) the circuit is generated by a linear time mapping algorithm. Only one additional input and one inverter are needed to achieve 100% testable circuits under the Stuck-At Fault Model. In this paper we study Automatic Test Pattern Generation (ATPG) for circuits derived from Binary Decision Diagrams (BDDs) under the Stuck-At fault model. Experimental studies for circuits derived from BDDs in comparison to optimized circuits are carried out. For the benchmark circuits the number of test patterns needed is comparable to the relative sizes of the circuits. Even a gain up to a factor of four occurred

1. Introduction

The size of circuits and systems is increasing rapidly. Due to this increase the test for failures becomes intractable if not considered during early design stages. This observation lead to approaches of *Synthesis for Testability*. Another important aspect is to consider layout aspects as early as possible in the design cycle. Binary Decision Diagrams (BDDs) [6] are a data structure that allow to combine both perspectives.

BDDs have originally been proposed as a data structure for ef£cient Boolean function representation and manipulation. Due to their compactness they have also been frequently used in logic synthesis approaches, since they allow to combine aspects of circuit synthesis and technology mapping [9].

Furthermore, on BDD circuits and BDDs, respectively, many operations can be carried out ef£ciently, like estimating power or considering layout aspects [8]. One further important argument for the use of BDD circuits are testability aspects. Due to the structural restrictions of BDDs, i.e. the ordering of the variables, testability can be ensured by construction. Multiplexor circuits derived from BDDs have been studied intensively under various fault models [2, 1, 4, 3]. (For an overview see [5].) Recently in [7] a technique has been proposed that ensures 100% testability of circuits derived from BDDs under the stuck-at fault model and even the path delay fault model at the cost of one additional input.

In this paper, ATPG was carried out on the combinational circuits from the LGSynth'91 [13] benchmark set, using TEGUS [12], an ATPG tool based on a Boolean satis£ability (SAT) formulation. BDD circuits were built for the benchmark circuits using the technique from [7] and tested using TEGUS. In experimental studies the resulting circuits are compared to circuits optimized by SIS [10]. The results show that all redundancies were removed. The overhead in size is moderate and also the number of test patterns needed is tightly correlated to the circuit size.

2. Binary Decision Diagrams and Circuits

As is well-known a Boolean function $f : B^n \rightarrow B$ can be represented by a BDD which is a directed acyclic graph

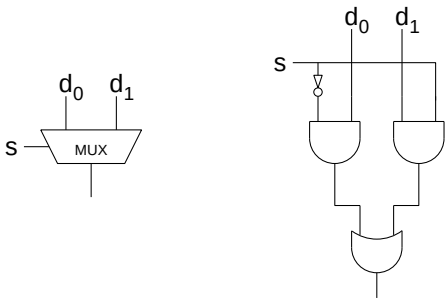

Figure 1. BDD node over MUX and STD

where a Shannon decomposition

$$\begin{aligned} f &= \overline{x}_i f_{\overline{x}_i} + x_i f_{x_i} \quad (1 \leq i \leq n) \\ f_{\overline{x}_i} &= f(x_1, \cdots, x_i = 0, \cdots, x_n) \\ f_{x_i} &= f(x_1, \cdots, x_i = 1, \cdots, x_n) \end{aligned}$$

is carried out in each non-terminal node with the low-edge pointing to $f_{\overline{x}_i}$ and the high-edge pointing to f_{x_i}. 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 vertices with both edges pointing to the same node. Reduced and ordered BDDs are a canonical representation since for each Boolean function the BDD is uniquely speci£ed. Furthermore, for functions represented by BDDs ef£cient manipulations can be carried out [6].

It is well-known, that BDDs directly correspond to multiplexor based Boolean circuits, called BDD circuits in this paper. More exactly: BDD circuits are combinational logic circuits de£ned over a £xed library. As been proposed in [4], we consider two libraries in the following (see Figure 1):

1. MUX: BDD nodes are substituted by multiplexor cells. Internal signals of these cells are not considered.
2. STD: BDD nodes are substituted by the AND-, OR-, NOT-realization of the Shannon decomposition.

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 $high(v)$. Finally, connect the output of the multiplexor which substituted the root node with a primary output.

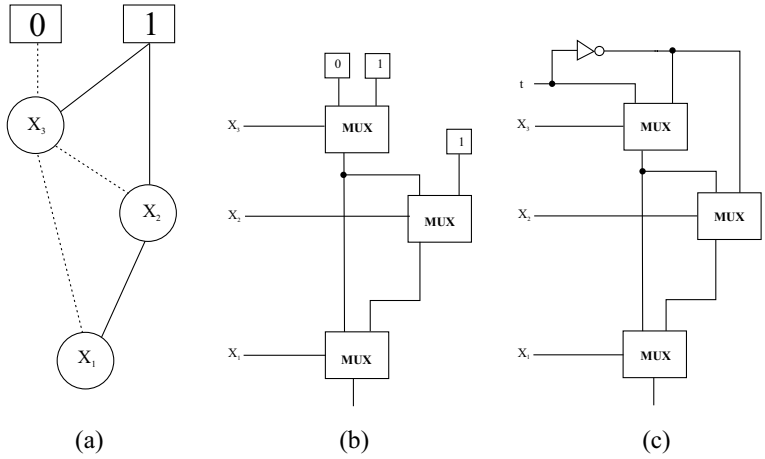

Figure 2. Mapping a BDD to a circuit

Remark 1. *The handling of nodes that have at least one pointer to a constant has direct implications on the testability of the resulting circuit:*

1. *As has been suggested in [4, 3], the MUX cells connected to constant values can be simpli£ed. This can result in redundancies, but the £nal circuits are smaller counted in the number of literals (see Figure 2b).*
2. *In [7] all terminals are connected to a new test input. By this 100% testability can be ensured by construction (see Figure 2c).*

3. Experimental Results

The techniques described above have been implemented in C. All experiments are run on a SUN Fire 280R with 3 Gbyte of main memory. The benchmarks are taken from LGSynth91 [13]. For each circuit initially the BDD is constructed and then mapped to the MUX library. This circuit is decomposed to cells from the STD library using SIS [10]. CUDD [11] has been used as the underlying BDD package. The number of literals needed for a circuit was determined using SIS, the number of test patterns, caught faults and redundancies was determined using the tool TEGUS [12].

Selected results are given in Tables 1 and 2 for circuits *optimized* by SIS using *script.rugged* and *BDD circuits* retrieved using the technique from [7]. In contrast to the experiments in [7] all circuits were mapped to the STD library. Table 1 shows characteristics regarding circuit sizes. The name of the benchmark is given in the £rst column. The number of inputs, the number of outputs and the number of literals are given in columns *in*, *out*, *lits*, respectively.

Column *lits(BDD)/lits(opt)* gives the ratio of literals in the BDD circuit to literals in the optimized circuit. The table shows that for several benchmarks a moderate overhead in size is introduced when BDD circuits are considered. But for other benchmarks the opposite is true, i.e. the BDD circuits are smaller than those optimized by SIS, e.g. for *t481* the optimized circuit is more than four times larger than the BDD circuit. Note, that the mapping of BDDs onto the STD library can be seen as a worst case for BDD circuits as the realization of a MUX cell in multiplexor based design styles is much cheaper.

In Table 2 data regarding the testability is shown. The number of test patterns, caught faults and redundancies are given in column *pattern*, *caught*, *red.*, respectively. The time for test pattern generation was small in all cases (less than 0.5 CPU seconds) and is therefore omitted. Column *pat(BDD)/pat(opt)* gives the ratio of test patterns for the BDD circuit to test patterns for the optimized circuit. The list of test patterns calculated during test pattern generation was compacted using fault simulation on the reversed list afterward.

Table 2 shows that the optimized circuits can contain a large number of redundancies. The synthesis approach facilitated by SIS does not take this aspect into account. On the other hand no redundancies are contained in the BDD circuits. Still the number of test patterns needed is very moderate.

Due to the structure of a BDD the number of test patterns could grow very large with the size of BDD circuits, as only one path from primary inputs to primary outputs can be tested at a time. But the experiments show the opposite. Figure 3 visualizes this observation. Each dot denotes results for one benchmark circuit. Compared are the relative circuit sizes and the relative number of test patterns needed for optimized and BDD circuits, respectively. Most circuits are below the bisecting line. This suggests that the growth of the number of test patterns is smaller on average than the growth of the circuit size.

In summary full testability can be achieved at a moderate overhead in circuit size and – if at all – at the cost of a small number of additional test patterns. These £gures even hold for the case of the STD library which is disadvantageous for BDD circuits. Even better results can be expected for multiplexor based design styles.

References

[1] P. Ashar, S. Devadas, and K. Keutzer. Gate-delay-fault testability properties of multiplexor-based networks. In *Int'l Test*

Table 1. Characteristics of circuits

	optimized			BDD circuit			
Circ.	in	out	lits	in	out	lits	$\frac{\text{lits(BDD)}}{\text{lits(opt)}}$
5xp1	7	10	277	8	10	238	.85920
C17	5	2	14	6	2	37	2.64285
alu2	10	6	700	11	6	1124	1.60571
b12	15	9	218	16	9	330	1.51376
b9	41	21	287	42	21	702	2.44599
c8	28	18	242	29	18	332	1.37190
cc	21	20	84	22	20	271	3.22619
cht	47	36	240	48	36	564	2.35000
clip	9	5	466	10	5	435	.93347
con1	7	2	36	8	2	73	2.02777
count	35	16	256	36	16	384	1.50000
cu	14	11	95	15	11	213	2.24210
decod	5	16	52	6	16	195	3.75000
duke2	22	29	818	23	29	2195	2.68337
e64	65	65	1054	66	65	903	.85673
f51m	8	8	285	9	8	218	.76491
frg1	28	3	404	29	3	626	1.54950
i1	25	13	74	26	13	252	3.40540
i3	132	6	620	133	6	756	1.21935
i5	133	66	264	134	66	792	3.00000
i6	138	67	762	139	67	1500	1.96850
i7	199	67	1016	200	67	1429	1.40649
i9	88	63	1048	89	63	9672	9.22900
lal	26	19	179	27	19	471	2.63128
ldd	9	19	128	10	19	409	3.19531
o64	130	1	258	131	1	774	3.00000
parity	16	1	90	16	1	91	1.01111
pcler8	27	17	160	28	17	511	3.19375
pm1	16	13	73	17	13	249	3.41095
rd53	5	3	60	6	3	91	1.51666
rd73	7	3	362	8	3	174	.48066
rd84	8	4	614	9	4	243	.39576
sao2	10	4	440	11	4	514	1.16818
sqrt8	8	4	146	9	4	180	1.23287
squar5	5	8	122	6	8	184	1.50819
t481	16	1	812	17	1	192	.23645
table3	14	14	1622	15	14	4787	2.95129
table5	17	15	1660	18	15	4158	2.50481
tcon	17	16	48	17	16	48	1.00000
term1	34	10	455	35	10	441	.96923
ttt2	24	21	416	25	21	669	1.60817
unreg	36	16	166	37	16	304	1.83132
vda	17	39	938	18	39	2943	3.13752
vg2	25	8	236	26	8	493	2.08898
x1	51	35	644	52	35	2879	4.47049
x2	10	7	84	11	7	165	1.96428
x3	135	99	1534	136	99	3401	2.21707
x4	94	71	626	95	71	2474	3.95207
xor5	5	1	24	5	1	24	1.00000
z4ml	7	4	76	8	4	90	1.18421

Table 2. Testability of circuits

Circ.	optimized			BDD circuit			$\frac{\text{pat(BDD)}}{\text{pat(opt)}}$
	pattern	caught	red.	pattern	caught	red.	
5xp1	41	228	0	28	242	0	.68292
C17	7	19	0	10	45	0	1.42857
alu2	92	562	13	203	995	0	2.20652
b12	38	220	5	45	321	0	1.18421
b9	58	293	1	83	702	0	1.43103
c8	30	263	0	46	337	0	1.53333
cc	18	118	0	42	283	0	2.33333
cht	35	315	0	62	601	0	1.77142
clip	75	376	2	57	410	0	.76000
con1	11	40	0	22	78	0	2.00000
count	29	276	0	42	390	0	1.44827
cu	24	99	0	42	220	0	1.75000
decod	19	88	0	30	192	0	1.57894
duke2	116	755	3	189	1944	0	1.62931
e64	348	1021	0	143	957	0	.41091
f51m	40	218	0	27	225	0	.67500
frg1	63	325	0	187	588	0	2.96825
i1	28	89	0	40	271	0	1.42857
i3	121	478	0	125	950	0	1.03305
i5	58	332	0	66	994	0	1.13793
i6	22	844	0	110	1538	0	5.00000
i7	90	1038	0	125	1419	0	1.38888
i9	79	1054	0	272	8536	0	3.44303
lal	34	209	0	52	476	0	1.52941
ldd	19	134	0	32	378	0	1.68421
o64	130	195	0	197	970	0	1.51538
parity	20	122	0	19	122	0	.95000
pcler8	26	165	0	61	504	0	2.34615
pm1	20	80	0	33	246	0	1.65000
rd53	13	66	0	13	98	0	1.00000
rd73	64	300	2	27	182	0	.42187
rd84	101	518	6	31	247	0	.30693
sao2	68	329	0	72	467	0	1.05882
sqrt8	28	131	0	34	175	0	1.21428
squar5	18	116	0	20	175	0	1.11111
t481	164	751	9	50	206	0	.30487
table3	260	1417	0	486	4209	0	1.86923
table5	302	1433	0	472	3675	0	1.56291
tcon	21	82	0	24	82	0	1.14285
term1	91	392	2	64	464	0	.70329
ttt2	57	389	0	63	626	0	1.10526
unreg	25	187	0	25	294	0	1.00000
vda	104	866	0	152	2602	0	1.46153
vg2	62	218	7	89	480	0	1.43548
x1	86	603	0	321	2641	0	3.73255
x2	21	84	0	24	168	0	1.14285
x3	113	1569	17	222	3185	0	1.96460
x4	61	717	0	185	2311	0	3.03278
xor5	6	34	0	6	34	0	1.00000
z4ml	21	79	0	15	100	0	.71428

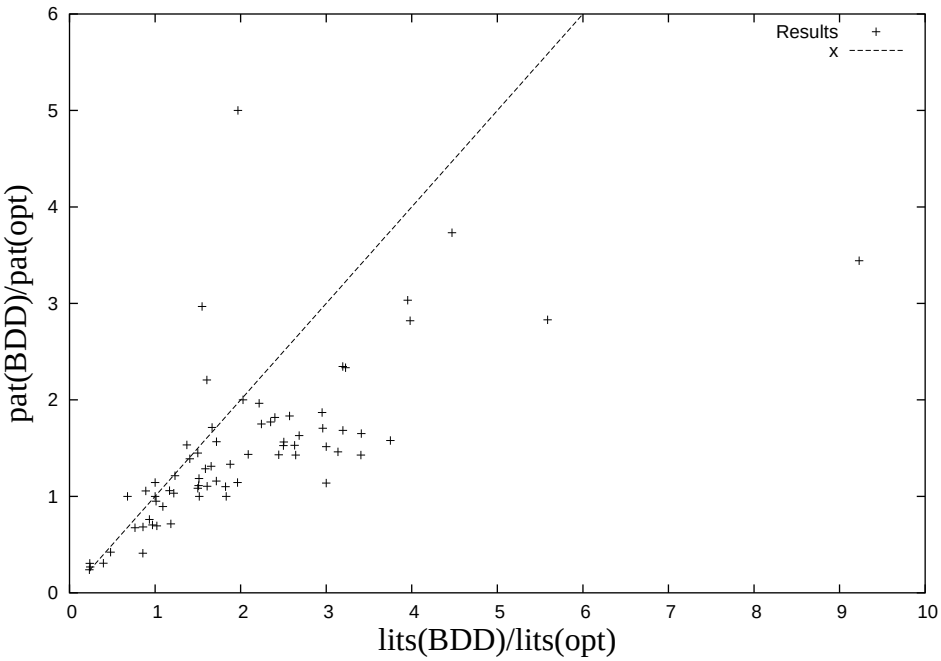

Figure 3. Relation between test patterns and literals

Conf., pages 887–896, 1991.

[2] P. Ashar, S. Devadas, and K. Keutzer. Testability properties of multilevel logic networks derived from binary decision diagrams. *Advanced Research in VLSI: UC Santa Cruz*, pages 33–54, 1991.

[3] 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.

[4] 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.

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

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

[7] R. Drechsler, J. Shi, and G. Fey. Synthesis of fully testable circuits from BDDs. *IEEE Trans. on CAD*, 23(3):440–443, March 2004.

[8] T. Eschbach, R. Drechsler, and B. Becker. Placement and routing optiization for circuits derived from bdds. In *IEEE International Symposium on Circuits and Systems*, 2004.

[9] 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.

[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] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Combinational test generation using satis£ability. In *IEEE Trans. on CAD*, volume 15, pages 1167–1176, September 1996.

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