FreezeFrame: Compact Test Generation Using a Frozen Clock Strategy

Yanti Santoso† Matthew Merten† Elizabeth M. Rudnick† Miron Abramovici††

†Center for Reliable and High-Performance Computing, University of Illinois, Urbana, IL, USA
††Bell Labs - Lucent Technologies, Murray Hill, NJ, USA

Abstract

Test application time is an important factor in the overall cost of VLSI chip testing. We present a new ATPG approach for generating compact test sequences for sequential circuits. Our approach combines a conventional ATPG algorithm, a technique based on the frozen clock testing strategy, and a dynamic compaction method based on a genetic algorithm. The frozen clock strategy temporarily suspends the sequential behavior of the circuit by stopping its clock and applying several vectors to increase the number of faults detected without changing the circuit state. Results show that test sets generated using the new approach are more compact than those generated by previous approaches for many circuits.

1 Introduction

Automatic test equipment is expensive, and therefore it is desirable to minimize the time required to test a VLSI chip. Of course, the quality of a test must be maintained while test application time is reduced. Various approaches have been used in the past to reduce test application time. Test set compaction is the most common approach, and both static and dynamic compaction approaches have been used. With static compaction, a test set is first generated, and attempts are then made to shorten it without reducing fault coverage. Greater reductions in test set length have been obtained using dynamic compaction [1] [2] [3] [4] [5]. With this approach, compaction is performed while tests are being generated.

In this paper, we use *the frozen clock testing strategy* to reduce the length of the test sequences. This approach, first proposed by Abramovici et al. [6], temporarily suspends the sequential behavior of the circuit by stopping its clock and applying several combinational vectors to primary inputs (PIs) without changing the circuit state. This enables several additional faults to be detected in each clock cycle that would otherwise not be detected until much later in the test. Fault effects that may be present at several different flip-flops can be propagated to the primary outputs (POs) with additional vectors applied while the clock is frozen. We will refer to these additional vectors as *unclocked vectors*. We will use the term *clocked vector* to refer to a normal sequential vector, for which the clock is strobed at the end of the cycle after the vector is applied.

Our new package for maximal test compaction, FreezeFrame, optimizes test sequences generated by the HITEC deterministic test generator [7]. HITEC generates test sequences for sequential circuits by targeting each fault individually. Not all bits in a sequence are necessarily specified, and each partially specified sequence is optimized by FreezeFrame using the procedures developed for the Squeeze dynamic compactor [5]. After an optimized sequence is obtained, FreezeFrame generates unclocked vectors using a genetic algorithm (GA) to detect additional faults if possible. Finally, after a complete test set is generated, FreezeFrame performs static compaction using a new tool called IcePick to remove any redundant unclocked vectors.

The rest of this paper is organized as follows. The GA used to evolve test sequences and unclocked vectors is explained in Section 2. Section 3 gives an overview of FreezeFrame, and details of the approach are given in Section 4. Results for ISCAS89 sequential benchmark circuits are presented in Section 5, and Section 6 concludes the paper.

2 Genetic Algorithms

A simple GA is used to solve the test sequence and unclocked vector optimization problems. The simple GA contains a population of *strings*, or *individuals* [8], and in our application, each individual represents a test sequence or unclocked vector. Each string has an associated *fitness*, which indicates the number of faults detected by the corresponding test sequence or unclocked vector. For test sequence compaction, the initial population is seeded with the test sequence generated by the deterministic test generator, and unspecified bits are

*This research was supported in part by DARPA under Contract DABT63-95-C-0069 and by Hewlett-Packard under an equipment grant.

filled randomly [5]. For unclocked vector generation, the first half of the initial population is seeded with a test vector generated using PODEM while the second half is seeded randomly, as will be described in Section 4. A highly fit population is evolved through several generations by *selecting* two individuals, *crossing* the two individuals, and *mutating* characters in the resulting individuals with a given probability. Tournament selection without replacement and uniform crossover [9] are used. Distinct generations are evolved, and the processes of selection, crossover, and mutation are repeated until all entries in a new generation are filled. Then the old generation is discarded. Since selection is biased towards more highly fit individuals, the fitness of the overall population is expected to improve in successive generations. However the best individual may appear in any generation, and this best individual is returned as the solution.

Compaction and test generation are different from other applications of GAs in that execution time is a critical factor. The GA is used repeatedly on a number of similar problems, and the time to evaluate the fitness of an individual can be long. Therefore, instead of using a large population and allowing the GA to run for a large number of generations to ensure that an optimal solution is found, we use a small population of size 32 and limit the number of generations to 8 to reduce the execution time. The test sequence evolved may not be the best possible, but results will show that very compact tests are indeed obtained.

3 Overview

FreezeFrame produces a test set containing sequences of clocked vectors and unclocked vectors, as illustrated in Figure 1. Unclocked vectors are shown in an italic font, and clocked vectors are shown in a regular font. Each clocked sequential test is delineated by heavy black bars, and the unclocked vectors to be applied before a clocked vector are shown to the left on the same line. After each clocked vector is applied, transitions propagate through the combinational logic of the circuit, and at the end of the clock cycle, the flip-flops are clocked to update the circuit state. However, after an unclocked vector is applied, the clock is not strobed. Therefore, no new fault effects are stored in the flip-flops. This allows these test vectors to propagate faults to the POs without changing the state of the circuit. Thus, fault effects present at the flip-flops after application of a previous clocked vector may be detected by a number of different unclocked vectors.

FreezeFrame uses HITEC [7] to produce each test sequence. HITEC is a deterministic test generator, and it targets one fault at a time by trying to activate the fault and propagate the fault effects to a PO. HITEC makes

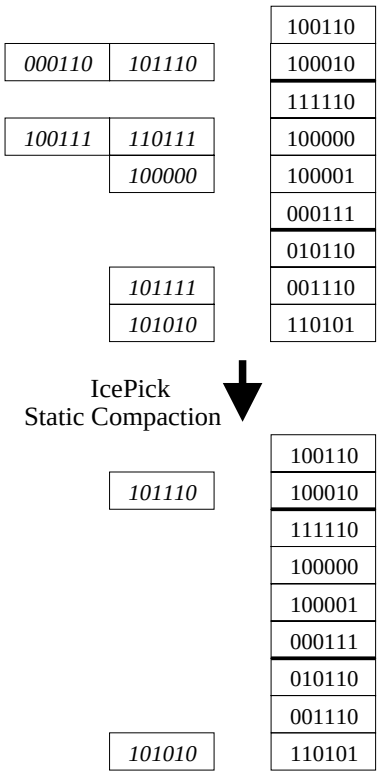

Figure 1: Test Set Before and After Compaction.

several passes through the fault list, with increasing time limits per fault for successive passes. The next fault in the fault list is selected as the target fault, and test generation is attempted. If HITEC is successful, the partially specified test sequence is sent to FreezeFrame, as shown in Figure 2. FreezeFrame uses the procedures developed for the Squeeze dynamic compactor [5] to produce a fully specified test sequence. Any unnecessary vectors at the beginning and end of the sequence are first trimmed using a fault simulator. HITEC assumes the circuit starts from an unknown state for every fault targeted. However, the fault simulator may have some information about what state the circuit is in after the previously applied sequences. Once the unnecessary vectors are removed, a GA is used to evolve the remaining vectors into a sequence that maximizes fault detection. The GA will randomly fill all unspecified bits with 1's and 0's in the first generation and optimize the sequence over a number of generations. Because the specified bits are allowed to change values, the original target fault may no longer be detected by the optimized sequence. However, it is likely that the fault will be detected in a later pass through the fault list, if it is not covered by a sequence generated for a different target fault.

Once a fully specified sequence is obtained, unclocked vectors are generated to propagate as many

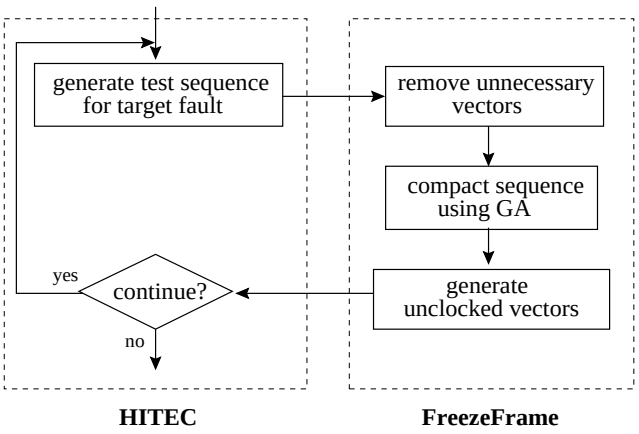

Figure 2: Test generation with dynamic compaction.

fault effects on the state lines to the POs as possible. For each clocked vector, each state line is considered individually, and an attempt is made to generate an unclocked vector that will propagate any fault effects on that state line. Unclocked vectors are applied when the clock is frozen, taking advantage of the internal state from the previous clocked vector to detect additional faults. Faults may be detected by later test vectors specifically generated to detect them, but by using the unclocked vectors, they can be detected sooner with fewer vectors. The earlier the fault is detected, the shorter the test generation time will be. Some of the unclocked vectors generated are redundant and can be removed, as shown in Figure 1. The redundant vectors detect faults that are detected by later clocked vectors or unclocked vectors. After test generation is completed, the IcePick tool performs fault simulation and set covering to generate the final optimized test set.

4 FreezeFrame

Figure 3 depicts the basic idea behind testing using a frozen clock strategy. Note that state lines that contain fault effects are shown on the left with the value 1/0 or 0/1. In this example, if the clocked test vector is applied by itself (Figure 3a), only one fault effect can be propagated to the PO. At this point, the circuit is clocked, and the state lines are changed; thus, the opportunity to detect the 0/1 fault effect is lost. If unclocked vectors can be applied to the PIs before the clocked test vector is applied (Figures 3b and 3c), two additional fault effects can be propagated to the POs. This way, the fault effects propagated to state lines are observed and faults are detected earlier. In the following subsections, test generation and static compaction of unclocked vectors will be described.

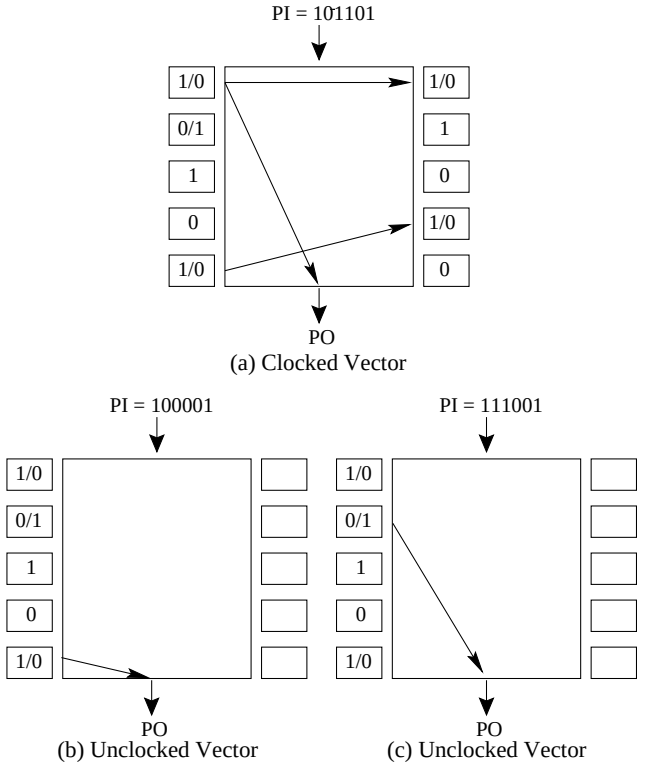

Figure 3: Unclocked vectors propagate fault effects to POs.

4.1 Test generation

Fault effects, simply put, are incorrect values on state lines due to one or more faults in the circuit. It is possible for several faults to propagate fault effects to the same state line, and the effects of a single fault may propagate to multiple state lines. To simplify unclocked vector generation, the faults themselves are not considered individually, but rather each state line that holds fault effects is considered. A deterministic algorithm is used to generate an unclocked vector to propagate fault effects from a given state line to the POs. However, the fault effects may be due to a single fault only, and fault effects for this fault may have propagated to another state line. These fault effects on the second state line could mask the fault effects from the first state line so that the fault is not detected. Therefore, we use a fault simulator combined with a GA to guarantee that the unclocked vector generated does in fact detect at least one fault. The purpose of the GA is to try to detect other faults in addition to the faults whose effects have propagated to the targeted state line.

After each clocked test vector sequence is generated by HITEC and compacted by the Squeeze algorithms, the faults detected by that sequence are eliminated

from the collapsed list of undetected faults. The modified fault list is then used to generate unclocked vectors for every vector in the original clocked sequence.

Unclocked vector generation requires that the state lines accurately represent the circuit's state after all clocked test vectors have been applied. This way, the state lines correctly hold the fault effects, and unclocked vectors can be produced to propagate these fault effects. To facilitate unclocked vector generation, a list of state lines that contain fault effects is built. Each flip-flop in that list is processed one-by-one. At each flip-flop, there is only one faulty value that may occur, unless the good value is undefined. When the good value is undefined, the fault effects propagated to that flip-flop are not put into the list, since observing the fault effects would not result in a detection. The 9-valued logic is used to specify completely the exact value of each node at each step.

For each state line in the list, FreezeFrame uses X-path check [10] to determine the feasibility of generating an unclocked vector. X-path check determines whether fault-effect propagation to at least one PO is possible. If X-path check does not fail, the PODEM algorithm [11] is used in an attempt to generate a partially specified vector that propagates fault effects from a particular flip-flop to the POs.

Next, a GA is used to generate a fully specified unclocked vector. The partially specified vector generated by PODEM is used as a seed for the first half of the initial GA population, while the second half is initialized randomly. If the X-path check fails or if PODEM fails, the entire GA population is initialized randomly. In order for the GA to work effectively, a variety of bit patterns is necessary for the unspecified bits in the unclocked vector. Therefore, each unspecified bit in each copy of the unclocked vector is filled at random with a 1 or 0. The fitness value for an unclocked vector was chosen to be the number of faults detected by the vector for a given fault sample. A small fault sample could be used to speed up the computation of the fitness value, since the entire fault list may be very large. Faults associated with the fault effects on the targeted state line should then be included in the fault sample. However, we have used the entire undetected fault list to simplify the implementation.

Once the population has evolved over 8 generations, the vector with the highest fitness value is added to the existing unclocked vectors for the particular clocked vector if it detects any additional faults, and the detected faults are dropped from the fault list. Unclocked vector generation continues for the next state line that contains fault effects. After all state lines that contain fault effects are processed, the clocked vector is applied, which propagates a new set of fault effects to the state lines, and the unclocked vector generation process begins again. Once all clocked vectors in the sequence have been processed in this manner, HITEC is called to generate a new clocked sequence to detect a fault in the remaining fault list. The test generation processes employed by FreezeFrame are described in the form of pseudocode in Figure 4.

```
For each test sequence s Do
    Compact and optimize s;
    Drop faults detected by s from fault list;
    For each vector v in s Do
        For each state line f with fault effects Do
            Reset GA;
            Run PODEM to produce unclocked
            vector seed;
            Run GA;
            If the most-fit unclocked vector, fv,
            detects at least 1 fault Then
                Add fv to the test set;
                Drop faults detected by fv;
```

Figure 4: Unclocked vector generation

4.2 Static compaction

Unclocked vectors are generated without the knowledge of the test sequences that are produced afterward. Therefore, there may be redundant unclocked vectors in the test set, and test compaction is needed. An unclocked vector will become redundant if all of the faults it propagates to POs are incidentally detected by other necessary clocked vectors.

Static compaction for unclocked vectors is done using a new tool called IcePick, as shown in Figure 1. The first step taken to compact the unclocked vectors eliminates unclocked vectors that detect only faults covered by clocked vectors. From the remaining unclocked vectors, the smallest set that covers the maximum number of undetected faults is selected. In general, the set cover problem is NP-complete. In practice, a greedy cover algorithm is often used that effectively approximates the NP-complete solution. Fault simulation is performed to construct a list of faults detected by each unclocked vector, and all essential unclocked vectors are chosen first. An essential unclocked vector detects a fault that no other unclocked vector detects. Those faults detected by the essential vectors are then dropped from the fault list. Finally, a greedy algorithm is employed to select a subset of the remaining vectors. The unclocked vector that covers the most undetected faults is chosen first, and those faults are dropped. The cycle is repeated, choosing an unclocked vector that detects

Table 1: FreezeFrame vs. Squeeze

Circuit	PI	Total Faults	Unt	HITEC + Squeeze [5]			HITEC + FreezeFrame					
				Det	Vec	Time	Det	Vec	uc vec	cuc vec	tot vec	Time
s298	3	308	26	265	133	16.7m	265	120	5	0	**120**	16.9m
s344	9	342	11	329	62	3.94m	329	58	12	0	**58**	4.53m
s349	9	350	13	334	53	3.67m	334	52	8	0	**52**	7.33m
s382	3	399	9	**359**	**485**	38.7m	354	342	0	0	342	52.8m
s386	7	384	70	314	128	22.2s	314	99	16	5	**104**	43.5s
s400	3	426	17	**375**	**454**	42.5m	373	356	0	0	356	52.7m
s444	3	474	24	**418**	**598**	43.0m	411	495	0	0	495	1.59h
s526	3	555	23	**359**	**84**	5.05h	358	97	8	0	97	5.17h
s641	35	467	63	404	**72**	32.3s	404	51	38	23	74	1.77m
s713	35	581	105	476	56	33.5s	476	41	27	15	56	1.33m
s820	18	850	36	814	462	4.63m	814	438	27	4	**442**	7.36m
s832	18	870	52	818	438	5.93m	818	416	15	2	**418**	8.74m
s1196	14	1242	3	1239	219	1.69m	1239	86	133	108	**194**	5.40m
s1238	14	1355	72	1283	232	1.84m	1283	95	142	109	**204**	5.48m
s1423	17	1515	14	**1047**	**137**	8.99h	993	136	5	1	137	9.30h
s1488	8	1486	41	1444	408	17.7m	1444	378	14	2	**380**	19.9m
s1494	8	1506	52	1453	**359**	11.2m	1453	374	23	4	378	13.6m
s3271	26	3270	5	**3253**	**521**	33.9m	3248	554	0	0	554	1.33h
s3330	40	2870	123	**2117**	**258**	10.4h	2114	256	71	1	257	11.2h
s3384	43	3380	1	**3066**	**118**	5.29h	3046	91	77	2	93	6.15h
s4863	49	4764	22	**4629**	**249**	2.23h	4627	236	44	9	245	4.24h
s5378	35	4603	224	**3245**	**198**	18.2h	3233	212	53	3	215	19.8h
s6669	83	6684	0	6663	135	29.2m	**6671**	149	156	0	**149**	1.59h
s35932	35	39,094	3984	**35,084**	**178**	2.44h	35,023	176	14	1	177	11.4h

Most compact test set having maximal fault coverage is highlighted in bold font
uc vec: Original number of unclocked vectors **cuc vec**: Number of compacted unclocked vectors

the most undetected faults, until there are no unclocked vectors remaining that cover any undetected faults. In general, essential selection followed by greedy selection yields a very good approximation to the absolute smallest set cover [12].

5 Results

FreezeFrame was implemented, and experiments were carried out on an HP-UX 9000/770/100 with 256MB RAM. Faults detected by the unclocked vectors in the beginning of the test set are usually easy-to-detect faults that will also be detected by the later clocked vectors. Even if many unclocked vectors are added near the beginning of the test set, most will be eliminated after static compaction. When FreezeFrame reaches test sequences later in the test set, there will be fewer unclocked vectors generated because there are fewer propagated fault effects due to the reduced fault list and because most of the faults left are hard-to-detect faults. In order to be efficient in both space and time, we allowed only 8 unclocked vectors to be added to each clocked vector. The starting population size for the GA is 32, and the test generation runs in 3 passes through the fault list. The time limits for the three passes are 0.5, 5, and 50 seconds per fault.

Results for FreezeFrame are shown in Table 1 and compared to Squeeze for ISCAS89 sequential benchmark circuits that are initializable using 3-valued logic. The number of primary inputs (PI), total faults, and untestable faults (Unt) are shown for each circuit, along with the number of detected faults (Det), number of clocked vectors (Vec), and execution time for both Squeeze and FreezeFrame. In addition, the number of original unclocked vectors (uc vec), compacted unclocked vectors (cuc vec), and total vectors (tot vec) are shown for FreezeFrame. The most compact test set having maximal fault coverage is highlighted in bold font. The overall FreezeFrame execution time is longer compared to that of Squeeze partly because of the additional unclocked vector test generation and static compaction.

In Table 2, FreezeFrame is compared to other previous approaches. Squeeze performs better in most circuits than other approaches. Therefore, it is justifiable to compare FreezeFrame to Squeeze. FreezeFrame's shorter test set achieves a comparable fault coverage for most of the circuits studied. In s641, s1494, s3271, and s5378, Squeeze still produces a shorter test set.

Generally, the fault coverage using FreezeFrame is

Table 2: Comparison to Previous Approaches

Circuit	SEQCOM [1]		COINS(*) [2]		[3]		BRF(2) [4]		Squeeze [5]		FreezeFrame	
	Det	Vec	Det	Vec	Det	Vec	Det	Vec	Det	Vec	Det	Vec
s386	314	131	314	154	306	380	-	-	314	128	314	104
s641	404	80	404	136	401	121	404	74	404	72	404	74
s713	-	-	-	-	467	142	476	71	476	56	476	56
s820	-	-	722	235	-	-	409	87	814	462	814	442
s832	-	-	-	-	-	-	494	101	818	438	818	418
s1196	1232	238	1235	307	1239	332	1239	228	1239	219	1239	194
s1238	-	-	-	-	1283	348	1283	244	1283	232	1283	204
s1423	-	-	-	-	-	-	1298	397	1047	137	993	137
s1488	1444	358	1444	566	-	-	1182	88	1444	408	1444	380
s1494	-	-	-	-	-	-	1075	84	1453	359	1453	378
s5378	-	-	-	-	-	-	3368	294	3245	198	3233	215

equivalent to Squeeze's. However, the fault coverage is sometimes lower for FreezeFrame, due to the non-determinism of the algorithms. In s6669, more faults are detected by FreezeFrame than Squeeze. In 9 of the ISCAS89 circuits, the test set is smaller compared to Squeeze and gives the same fault coverage. This indicates that FreezeFrame can reduce the test set size to reduce test application time while maintaining the same fault coverage. The FreezeFrame test set tends to be smaller because with the addition of unclocked vectors, more faults are detected earlier.

6 Conclusion

Since test application time is an important factor in reducing the VLSI chip testing cost, shorter test sets are preferred. Still, high fault coverage should be maintained. The traditional approach to producing shorter test sets is through compaction. In our work, we developed a package called FreezeFrame that uses an approach introduced in [6], which involves freezing the clock. In FreezeFrame, we use the HITEC test generator and Squeeze dynamic compaction procedures to produce the initial test set, along with a new algorithm to generate unclocked vectors to be applied while the clock is frozen, and IcePick to statically compact the unclocked vectors. FreezeFrame produces more compact test sets than other approaches while maintaining a comparable fault coverage for several of the ISCAS89 sequential benchmark circuits.

Acknowledgment

We would like to thank Xiaoming Yu for assisting with the implementation and generating results.

References

[1] I. Pomeranz and S. M. Reddy, "On generating compact test sequences for synchronous sequential circuits," *Proc. European Design Automation Conf. (EURO-DAC)*, pp. 105–110, 1995.

[2] I. Pomeranz and S. M. Reddy, "Dynamic test compaction for synchronous sequential circuits using static compaction techniques," *Proc. Int. Symp. Fault-Tolerant Computing*, pp. 53–61, 1996.

[3] A. Raghunathan and S. T. Chakradhar, "Dynamic test sequence compaction for sequential circuits," *Proc. Int. Conf. VLSI Design*, pp. 170–173, 1996.

[4] T. J. Lambert and K. K. Saluja, "Methods for dynamic test vector compaction in sequential test generation," *Proc. Int. Conf. VLSI Design*, pp. 166–169, 1996.

[5] E. M. Rudnick, and J. H. Patel, "Putting the squeeze on test sequences," *Proc. Int. Test Conf.*, pp. 723–732, Nov. 1997.

[6] M. Abramovici, K. B. Rajan, and D. T. Miller, "FREEZE!: A new approach for testing sequential circuits," *Proc. Design Automation Conf.*, pp. 22–25, 1992.

[7] T. M. Niermann, and J. H. Patel "HITEC: A test generation package for sequential circuits," *Proc. of the European Conference on Design Automation*, pp. 214–218, Feb. 1991.

[8] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," *Reading, MA: Addison-Wesley*, 1989.

[9] E. M. Rudnick, J. H. Patel, G. S. Greenstein, and T. M. Niermann, "Sequential circuit test generation in a genetic algorithm framework," *Proc. Design Automation Conf.*, pp. 698–704, 1994.

[10] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, The Institute of Electrical and Electronics Engineers, Inc., New York, 1990.

[11] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. on Computers*, vol. C-30, no. 3, pp. 215–222, March 1981.

[12] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, 1984.