Combinational Profiles of Sequential Benchmark Circuits

Franc Brglez*, David Bryan, Krzysztof Koźmiński

Microelectronics Center of North Carolina
P.O.Box 12889, 3021 Cornwallis Rd, Research Triangle Park, NC 27709

Abstract

This paper presents a set of 31 digital sequential circuits described at the gate level. These circuits extend the size and complexity of the ISCAS'85 set of combinational circuits and can serve as benchmarks for researchers interested in sequential test generation, scan–based test generation, and mixed sequential/scan–based test generation via partial scan techniques. Although all the benchmark circuits are sequential, synchronous, and use only D-type flip–flops, additional interior faults and asynchronous behavior can be introduced by substituting some or all of the flip–flops with their appropriate functional models. The standard functional model of a D flip–flop provides a reference point independent of the faults particular to the flip–flop implementation. In this paper, a testability profile of the benchmarks in the full–scan mode configuration is discussed.

Introduction

The advances in automatic test pattern generation (ATPG) algorithms for scan–based systems [1,2] prompted an exchange of benchmark circuits [3] that were distributed to special session participants during ISCAS'85 [4]. These benchmarks are characterized in more detail in [5,6]. Since their introduction, the benchmarks have been widely distributed and have been used not only for the purpose of evaluating the performance of ATPG algorithms but also in the areas of simulation, fault simulation, testability analysis, formal verification, logic synthesis, technology mapping and layout synthesis. The ISCAS'85 benchmarks contain faults for which no tests can be found and proving them redundant with reasonable computational effort remains a challenge even today. The task of proving *all* untestable faults in these benchmarks to be redundant has been accomplished for the first time only recently [7].

For ISCAS'89, we have prepared a new set of benchmarks that can be used for evaluating not only full scan–based algorithms for test pattern generation, but also to identify opportunities and limitations for sequential test pattern generation algorithms, with or without the assistance of partial scan. Most significantly, these benchmarks extend the range of current ISCAS'85 benchmarks both in their size as well as complexity. While all benchmark netlists are by default expanded for synchronous circuit behavior, asynchronous behavior and additional faults can be introduced by appropriate modeling of D flip–flop primitives as discussed later in the paper.

Following our call for contributions in September 1988, the benchmarks were assembled from various netlist formats received from industrial and university sources from the USA and abroad. We also synthesized several of the benchmarks automatically, as a part of the on–going research at MCNC.

The first part of the paper presents the benchmarks, their essential circuit parameters, and any structural characteristics that were passed on to us. A testability profile is determined for the full–scan mode configurations of all circuits in order to establish the basis for comparisons of scan–, partial scan–, and sequential–based methods in terms of fault coverage and test pattern generation cost. The second part of the paper discusses the netlist details, the current distribution format, and describes the method used at MCNC to produce several different levels of IC representations, including the ones distributed with the rest of the benchmark set, from a single high–level description. Our target for the future is to have a consistent set of different–level representations of identical circuits that will help to coordinate the test pattern generation efforts on the switch–level, gate–level, and higher levels of abstraction, and to make possible fair comparisons between test pattern generation algorithms operating on different levels.

Benchmark characteristics and profiles

We present a total of 31 circuit benchmarks; their essential statistics are shown in Table 1. We adopt the naming convention similar to the ISCAS'85

---

* Also with Bell Northern Research, Research Triangle Park, NC 27709.

(217) 333-4789 Further information can be obtained from Dr. W. Kenneth Jenkins.

set: *s#*. The letter *s* signifies that the circuit is synchronous *s*equential; the number that follows represents the number of interconnect lines among the circuit primitives. Note that the double of this number also represents the upper bound on the size of the single stuck–at fault list. The actual size of the fault list reported in Table 1 has been obtained by a simple fault collapsing method outlined in the next section.

Circuit Name	# of Primary Inputs	# of Primary Outputs	# of D–type flipflops	# of AND/OR/ NOT Gates	# of Collapsed Faults
s27	4	1	3	10	32
s208	11	2	8	96	215
s298	3	6	14	119	308
s344	9	11	15	160	342
s349	9	11	15	161	350
s382	3	6	21	158	399
s386	7	7	6	159	384
s400	3	6	21	162	424
s420	19	2	16	196	430
s444	3	6	21	181	474
s510	19	7	6	211	564
s526n	3	6	21	194	553
s526	3	6	21	193	555
s641	35	24	19	379	467
s713	35	23	19	393	581
s820	18	19	5	289	850
s832	18	19	5	287	870
s838	35	2	32	390	857
s953	16	23	29	395	1079
s1196	14	14	18	529	1242
s1238	14	14	18	508	1355
s1423	17	5	74	657	1515
s1488	8	19	6	653	1486
s1494	8	19	6	647	1506
s5378	35	49	179	2779	4603
s9234	19	22	228	5597	6927
s13207	31	121	669	7951	9815
s15850	14	87	597	9772	11727
s35932	35	320	1728	16065	39094
s38417	28	106	1636	22179	31180
s38584	12	278	1452	19253	36305

Table 1. Benchmark circuit characteristics

No schematic diagrams are provided for these benchmark circuits. Also, the functional descriptions are not available for most of the circuits. The following is a summary of what we presently know about the circuits:

- s349 is a 4–bit multiplier,
- s298, s400, s444, and s526 are traffic light controllers,
- s9234, s13207, s15850, s38417, s38584 are real–chip based and rely on partial scan,
- s386, s510, s953, and s1494 are controllers, synthesized from a high level description,
- s1238 is a combinational circuit with randomly inserted flip–flops,
- s208, s420, and s838 are digital fractional multipliers [8], synthesized hierarchically from a high level description,
- s298, s208, s713, s641, and s832 are based on PLD devices,
- s344, s382, s526n, s641, s820, s1196, s1488 have been re–synthesized from s349, s400, s526, s713, s832, s1238, s1494, after removing all redundancies in full–scan mode [9]

A full–scan mode testability profile of these benchmarks is given in Tables 2, 3, and 4.

Circuit Name	Random Pattern Fault Coverage	Overall test		CPU Usage		Length of the IC Test
		Useful Patterns	Final Coverage	Random	DTPG	
s27	100	7	100	0.3		31
s208	100	34	100	0.9		314
s298	100	36	100	0.6		554
s344	100	25	100	0.6		415
s349	99.43	25	99.43	13	0.7	415
s382	100	36	100	0.8		813
s386	100	75	100	1.4		531
s400	98.58	35	98.58	16	1.3	791
s420	97.91	59	100	19	1.2	959
s444	97.05	37	97.05	25	2.3	835
s510	100	62	100	1.1		440
s526n	100	70	100	5.6		1561
s526	99.82	70	99.82	17	0.9	1561
s641	99.14	59	100	25	1.3	1199
s713	92.77	59	93.46	35	11	1199
s820	100	123	100	6.6		743
s832	98.39	122	98.39	33	4.7	737
s838	88.53	118	100	48	12	3926
s953	100	99	100	17		2999
s1196	99.84	146	100	37	2.2	2792
s1238	94.75	157	94.90	60	17	3001
s1423	99.08	75	99.08	51	7.6	5699
s1488	100	138	100	6.2		972
s1494	99.20	137	99.20	53	4.9	965
s5378	99.11	274	99.13	195	18	49499
s9234	89.65	474	93.47	662	558	108774
s13207	98.40	491	98.45	658	151	329639
s15850	94.06	517	96.68	905	736	309763
s35932	89.81	82	89.81	2818	30015	143506
s38417	96.74	1099	99.45	2161	1942	1800699
s38584	95.66	763	95.85	1803	3720	1110091

Table 2. Testability profile in full scan mode

Our test generation system is based on the components described in [10,11,12] and runs on a VAX-8650. The two main points highlighted in

Tables 2, 3, and 4 are: random pattern testability of the circuits, and the length of test sequences required to test for the shown fault coverage in the full scan mode. This length is a function of the number of useful test patterns, n, and number of flip–flops in the scan chain, d: $n*(d+1)+d$. The random pattern grading algorithm was run for 100,000 trial patterns, unless a 100% fault coverage was attained earlier; this occurred for circuits listed in Table 3.

Circuit Name	Number of Random Patterns for 100% coverage
s27	64
s208	3200
s298	384
s344	128
s382	480
s386	3040
s510	1184
s526n	25216
s820	17440
s953	46464
s1488	8480

Table 3. Circuits that are 100% random testable.

Circuit Name	Redundant Faults	Aborted Faults
s349	2	NONE
s400	5	1*
s444	11	3*
s526	1	NONE
s713	18	20*
s832	13	1*
s1238	68	1*
s1423	5	9†
s1494	12	NONE
s5378	37	3†
s9234	219	235
s13207	131	20
s15850	308	81
s35932	2832	1152
s38417	79	100
s38584	1235	271

Table 4. Circuits with redundant and/or aborted faults (see text for the meaning of * and †).

For random pattern resistant faults, our deterministic test pattern generator (DTPG) [12] was run with a backtrack limit of 50. The final fault coverage given in Table 2 is not always 100%; the data about faults for which DTPG found no tests are given in Table 4. Even with the backtrack limit of 50, we found that most of the faults not covered in random pattern grading were redundant. However, all of the synthesized and/or re–synthesized circuits were 100% testable in the full scan mode.

For some circuits we could prove redundancy of all faults aborted with 50 backtracks by extending the backtrack limit in our DTPG; these circuits are marked in table 4 with '*'. The total additional computation cost for these circuits was 90 seconds of CPU time. We also reduced the number of aborted faults in s1423 to 5, and in s5378 to 1; these circuits are marked with '†'. The CPU time used for these circuits was 644 seconds.

We made no attempt to optimize the overall cost of the test pattern generation process; the cost could be significantly reduced on some random pattern–resistant circuits if we chose to engage the deterministic test pattern generator earlier.

Levels of Benchmark Representation

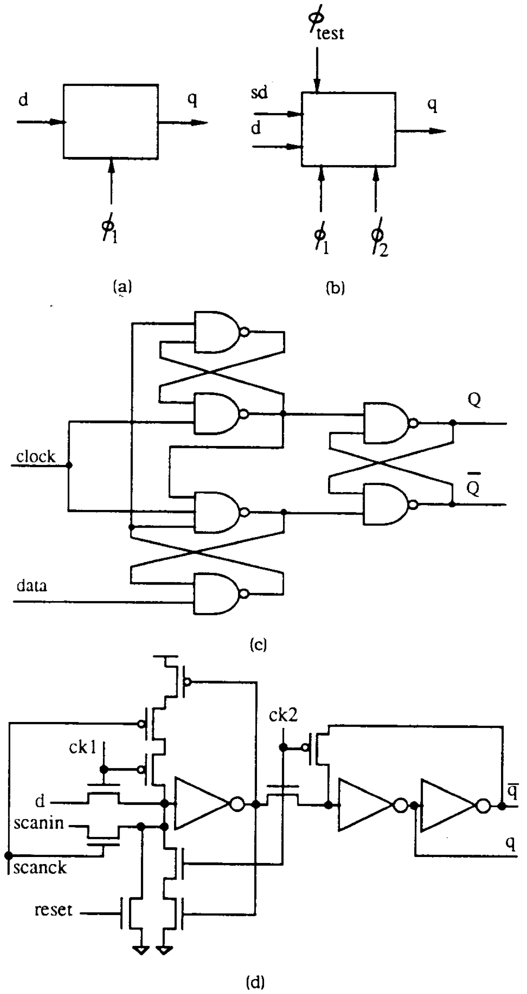
In this section we describe the current distribution format and illustrate our approach to generating different levels of benchmark representation from a single high–level description. We have used this approach not only to develop some of the testability benchmarks but also to contribute to a new set of layout benchmarks for the forthcoming workshop on module generation [13].

The benchmark circuits distributed to ISCAS'89 session participants were all described on the gate level. A netlist example of circuit S27, the simplest benchmark, is shown in Fig. 1.

```
# 4 inputs
# 1 outputs
# 3 D-type flip-flops
# 2 inverters
# 8 gates(1 ANDs +1 NANDs +2 ORs +4 NORs)
INPUT(G0)
INPUT(G1)
INPUT(G2)
INPUT(G3)
OUTPUT(G17)
G5 = DFF(G10)
G6 = DFF(G11)
G7 = DFF(G13)
G14 = NOT(G0)
G17 = NOT(G11)
G8 = AND(G14,G6)
G15 = OR(G12,G8)
G16 = OR(G3,G8)
G9 = NAND(G16,G15)
G10 = NOR(G14,G11)
G11 = NOR(G5,G9)
G12 = NOR(G1,G7)
G13 = NOR(G2,G12)
```

Fig. 1. Netlist of the s27 benchmark circuit

We chose a netlist format that is concise, self–documenting and easy to parse. The only sequential element is the D–type flip–flop with a single data input and a single uncomplemented output; we leave the choice of inserting additional outputs and/or control signals to the user. In contrast to combinational gates, expanding the flip–flops to switch–level representations offers many options. Fig. 2 illustrates four possible options that could be used to expand the flip–flops for the purposes of either test generation or fault simulation: a) functional level with a single clock, b) functional level with scan input, controlled by clock and test mode signals, c) gate–level controlled by a single clock and d) switch–level with the functionality of b) and an additional reset signal.



(a) (b)

(c)

(d)

Fig. 2 Sample of modeling options for a D flip–flop

The model shown in Fig. 2b represents the function of a scan–path flip–flop described at gate–level in [14], the model in Figure 2d is based on the scannable flip–flop in MCNC's 1.25 μm standard cell library [15]. The flip–flop models shown in Fig. 2a and Fig. 2b could also be augmented with a reset signal. In either configuration, functional models are useful to evaluate the intrinsic difficulties of test generation or fault simulation process, before introducing dependencies due to particular treatment of faults internal to flip–flops.

Since our netlist representation is neutral with respect to details of the flip–flop implementation, these benchmarks can be used by researchers who are interested in sequential test generation, scan–based test generation, or in using the concepts of partial scan [16,17]. The fault lists for the benchmark circuits are based on the single stuck–at model on the pins of each gate, including the data input and the data output of the flip–flop. There are no faults on the lines that would be user–specific such as clock lines or reset lines. The fault counts reported in Table 1 refer to the collapsed fault set consisting of one fault per each AND/OR/NAND/NOR gate input and two faults per output of any gate with a fanout greater than 2. Primary outputs and inputs to flip–flops always have two faults, and a single fault is associated with arbitrarily long chains of inverters.

We conclude this section by describing the process of synthesizing some of the benchmarks in Table 1. Note that all of these benchmarks are 100% testable in full scan mode. Our experience indicates that the typical CPU cost of test generation for synthesized circuits is less than the cost of the synthesis process itself. Synthesis of larger designs is done hierarchically in order to contain the cost; however, from the user's perspective, the compilation of the complete circuit structure is a single–step processing of a description on a mixed functional/structural level, with a host of logic synthesis tools run in the background [18,19,20,21]. The most notable advantage of this approach is that we can automatically maintain consistency of several levels of representations by deriving all of them from a single source.

The example below illustrates some of the details of synthesizing the s838 benchmark. The top–level structural description of a digital fractional multiplier [8] is shown in Fig. 3, using LOGIC–III language [21,22]. At this level, we interconnect only three modules: `counter`, `and_array` and `SUM_BLOCK`. Except for the signal type "`NODE`", all other signal types at the `CIRCUIT` level are I/O pin signals, some of which are global, such as `RESET` and `CLOCK`. During compilation, `RESET` and `CLOCK` signals are automatically connected to all D-type flip-flops. Two modules are controlled by the integer parameter `Nslices`; we synthesize our modules in increments of 4 bits.

```
CIRCUIT dfm32;
VAR
   reset : RESET;
   Phi1H, Phi2H, Phi1_test : CLOCK;
   X, Clear: INPUT;
   C: array[0..32] of INPUT;
   W, Z: OUTPUT;
   Y: array[1..32] of NODE;
   P: array[0..32] of NODE;
   Nslices: INTEGER;
BEGIN
   Nslices := 8;
   counter (Nslices,Clear,X,W,Y);
   and_array (Nslices,X,Y,P);
   SUM_BLOCK (P,C,Z);
END.
```

Fig. 3. Top level specs for s838, alias dfm32

An example of a 4-bit counter specification is shown in Fig. 4.

```
LOGIC_MODULE
cnt4 (Clear, X: INPUT;Carry: OUTPUT);
VAR
   st : array[1..4] of STATE;
   i : INTEGER;
BEGIN
   if (Clear) then
      st := 0
   else
      for i := 0 to 15 do
         begin
            if (X AND (st = i)) then
            st := i+1 ;
         end;
   Carry := 0;
   if (st = 15) then
   Carry := 1;
END.
```

Fig. 4. LOGIC-III specification of a 4-bit ripple counter.

With the help of several logic synthesis tools this specification is compiled into a netlist of standard cells shown in Fig. 5. This netlist can be used to drive a standard cell layout system [15], or can be used to considerable advantage within a hierarchical test pattern generation system [12]. For the purpose of generating the testability benchmarks supporting gate-level test pattern generation, it was flattened into gate primitives such as shown in Fig. 1.

Alternatively, the standard cells can be flattened down to the transistor level such as shown in Figs. 6 and 7. This representation can be used to drive a switch-level simulator, a switch-level verifier or a test pattern generator, as well as a layout synthesis system. Several of the benchmarks in Table 1 have been compiled into a switch-level netlist to augment the benchmark set for the forthcoming workshop on module generation [13].

```
oai222s INSI237
(I13,I233,I11,I225,I12,I221,I106)
oai21s INSI242 (I301,I170,I221,I104)
oai21s INSI243 (I171,I226,I233,I105)
ai2s INSI244 (I301,I103,I221)
oai21s INSI245 (I167,I307,I102,I225)
ai3s INSI246 (I311,I307,I170,I233)
```

Fig. 5: Sample of a standard cell-level netlist.

```
cell begin oai21s xgrid=16 ygrid=4 wire=6
...
translist
 pb b q vdd width=32 length=2 type=p
 pa2 a2 n2664 q width=29 length=2 type=p
 pa1 a1 vdd n2664 width=26 length=2 type=p
 nb b n108 gnd width=19 length=2 type=n
 na2 a2 q n108 width=18 length=2 type=n
 na1 a1 n108 q width=18 length=2 type=n
;
cell end oai21s
```

Fig. 6: A representation of a standard cell oai21s.

```
p I221 I104 Vdd 2.00 32.00 r 0 0 64.00
p I170 I242.2664 I104 2.00 29.00 r 0 0 58.00
p I190 Vdd I242.2664 2.00 26.00 r 0 0 52.00
e I221 I242.108 GND 2.00 19.00 r 0 0 38.00
e I170 I104 I242.108 2.00 18.00 r 0 0 36.00
e I190 I242.108 I104 2.00 18.00 r 0 0 36.00
```

Fig. 7: Transistor-level netlist based on oai21s cell.

The hierarchical expansions of benchmark circuits into various levels provide an opportunity for consistent comparisons of tradeoffs not only in test pattern generation but also in synthesis, verification, simulation and layout.

Benchmark Distribution

Testability benchmarks will be made available to interested researchers by June 1989. Anyone interested is invited to use the file transfer program *ftp* to login to *mcnc.org* as *anonymous*. A file named ISCAS89.INF in the *benchmark/news* directory will contain the up-to-date information about the ISCAS'89 set. Several benchmark sets, including ISCAS'85, are already available via *ftp*. Persons having difficulties with *ftp* access, should send their request to: *Benchmark Secretary, MCNC, P.O.Box*

(217) 333-4789 Further information can be obtained from **Dr. W. Kenneth Jenkins.**

12889, Research Triangle Park, N.C, or use electronic mail to *benchmarks@mcnc.org*.

Acknowledgements

Thanks are due to Balaji Krishnamurthy for organizing the special test session at ISCAS'89. We could not have assembled as diverse a set of benchmarks without contributions from several sources: while the contributors wish to remain anonymous, every benchmark user owes them considerable appreciation. We continue to receive useful feedback from the ISCAS'89 session participants who are the first users of these benchmarks. The project could not have been initiated without the understanding and direct support from MCNC. The cost of the benchmark tape distribution and administration is supported by a grant from ACM.

References

[1] P. Goel, *An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits*, IEEE Trans. Comp., Vol. C–30, No. 3, pp. 215—222, March 1981.
[2] H. Fujiwara, T. Shimono, *On the Acceleration of Test Generation Algorithms*, IEEE Trans. Comp., Vol. C–32, No. 12, pp 1137–1144, Dec. 1983.
[3] F. Brglez, H. Fujiwara, *A Neutral Netlist of 10 Combinational Benchmark Circuits and A Target Translator in FORTRAN*, Int. Symp. Circuits and Systems, IEEE, June 1985.
[4] Special Session: *Recent Algorithms for Gate-Level ATPG with Fault Simulation and Their Performance Assessment*, Int. Symp. Circuits and Systems, IEEE, pp. 663—698, June 1985.
[5] F. Brglez, P. Pownall, R. Hum, *Accelerated ATPG and Fault Grading via Testability Analysis*, Int. Symp. Circuits and Systems, IEEE, pp. 695—698, June 1985.
[6] Franc Brglez, *A Fast Fault Grader: Analysis and Applications*, Int. Test Conf., IEEE, pp. 785—794, Nov. 1985.
[7] M.H. Schulz, E. Auth, *Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques*, 18th Int. Symp. Fault–Tolerant Computing, June 1988.
[8] H.T. Nagle, Jr., B.D. Carrol, J.D. Irwin, *An Introduction to Computer Logic*, Prentice–Hall, 1975, pp. 461—463.
[9] D. Bryan, F. Brglez, *CRISP: A Redundancy Removal System for Scan–Based Logic*, Tech. Report TR89–13, MCNC, Research Triangle Park, NC, Feb. 1989.
[10] J. Calhoun, D. Bryan and F. Brglez, *Automatic Test Pattern Generation for Scan–Based Digital Logic*, Tech. Report TR87–17, MCNC, Research Triangle Park, NC, Aug. 1987.
[11] F. Brglez, D. Bryan, J. Calhoun, R. Lisanke, *A Modular Scan–Based Testability System*, Int. Conf. Computer Design, pp. 408–412, Oct. 1988.
[12] J. Calhoun, F. Brglez, *A Framework and Method for Hierarchical Test Generation*, Tech. Report TR89–06, MCNC, Research Triangle Park, NC, Jan. 1989.
[13] Physical Design Workshop, *Module Generation and Silicon Compilation*, April 30–May 3, 1989, Long Beach, CA. The benchmarks are available from Module Generation Benchmarks, MCNC, P.O. Box 12889, Research Triangle Park, NC 27709.
[14] S. Funatsu, N. Wakatsuki, T. Arima, *Test Generation Systems in Japan*, Proc. 12th Test Automation Symp., Vol. 6, pp. 114–122, 1975.
[15] *VPNR: Vanilla Place and Route System Manual*, MCNC, Research Triangle Park, NC, Dec. 1988.
[16] E. Trishler, *Incomplete Scan Path with an Automatic Test Generation Methodology*, Int. Test Conf., pp. 153–162, Nov. 1980.
[17] V.D. Agrawal, K–T. Cheng, D.D. Johnson, T. Lin, *A Complete Solution to the Partial Scan Problem*, IEEE Int. Test Conf., pp. 44–51, Sept. 1987.
[18] H.Abdalla, G. Kedem, F. Brglez, *LOGIC–II: A high Level Language for a Cell–Based Design System*, Tech. Report TR87–16, MCNC, Research Triangle Park, NC, Aug. 1987.
[19] R. Lisanke, G. Kedem, F. Brglez, *DECAF: Decomposition and Factoring for Multi–Level Logic Synthesis*, Tech. Report TR87–15, MCNC, Research Triangle Park, NC, Aug. 1987.
[20] R. Lisanke, F. Brglez, G. Kedem, *McMAP: A Fast Technology Mapping Procedure for Multi–Level Logic Synthesis*, IEEE Int. Conf. Computer Design, Oct. 1988.
[21] F. Brglez, D. Bryan, J. Calhoun, G. Kedem, R. Lisanke, *Automated Design for Testability*, IEEE Trans. Industrial Electronics, May 1989.
[22] G. Kedem, F. Brglez, *OASIS: Open Architecture Silicon Implementation System*, Tech. Report TR88–06, MCNC, Research Triangle Park, NC, Feb. 1988.