IEEE the 8th Asian Test Symposium (ATS'99), pp. 263-268, Nov. 1999.

New DFT Techniques of Non-Scan Sequential Circuits with Complete Fault Efficiency

Debesh Kumar Das
Dept. of Comp. Sc. and Engg.
Jadavpur University
Calcutta-700 032, India
debeshd@hotmail.com

Satoshi Ohtake Hideo Fujiwara
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5, Takayama-Cho, Ikoma, Nara 630-0101, Japan
{ohtake, fujiwara}@is.aist-nara.ac.jp

Abstract:

As opposed to scan schemes, a non-scan DFT allows at-speed testing. This paper suggests three techniques on non-scan DFT of sequential circuits. The proposed techniques guarantee 100% fault efficiency by using combinational ATPG tool. In all techniques, an additional circuit called CRIS is proposed to reach unreachable states on the state register of a machine. The second and third techniques use an additional hardware called differentiating logic (DL), that uniquely identifies a state appearing in a state register. The design of DL is universal, i.e., not dependent on the circuit structure. Hardware overhead of DL and CRIS is lower than that of full scan. Test generation and application time are found to compare favorably with those of earlier designs.

1. Introduction

To achieve a good design-for-testability (DFT) technique, the designers must have following goals – 1) to decrease test generation time, 2) to decrease test application time, 3) to have high fault efficiency*, 4) to achieve at-speed testing, and 5) hardware overhead should also be less in the designs. One approach in DFT designs is scan technique. In full scan technique [1,2], the test generation problem of a sequential circuit is reduced to that of a combinational one and use of combinational ATPG guarantees complete fault efficiency. Partial scan [3,4] offers low hardware overhead than full scan, but as it uses sequential test generation methods, high fault efficiency cannot be achieved. However, scan techniques fail to provide at-speed testing. To avoid the problems of scan techniques, non-scan approaches are proposed in [6-9]. In [6], some flip-flops are controlled by multiplexers. In [7], DFTs are designed using locally available lines. In [8], non-scan design was targeted only to remove equivalent and isomorph redundancy. In all these approaches, though testing time may be improved, complete fault efficiency cannot be achieved. Non-scan DFT approach with complete fault efficiency using combinational ATPG is first proposed in [9].

This paper suggests three new non-scan DFT techniques for sequential circuits. In the proposed techniques, test sequences for different faults in a sequential machine are found by generating test patterns by a combinational ATPG tool used on combinational part of the machine and use of such ATPG tool guarantees complete fault efficiency. As each test pattern generated by this ATPG tool consists of values on primary inputs as well as state registers, some test patterns may consist some values that can never be reached by state transitions from reset states. To reach such values on state registers (invalid states), we propose a technique to append an extra logic called circuit to reach invalid states (CRIS) with the original machine. Among the three techniques, the first one requires k additional observable points (k is the number of flip-flops in the circuit). Use of one more additional circuit called as Differentiating Logic (DL) greatly reduces the number of additional observable points in second and third techniques. The DL part of the proposed additional hardware is universal (i.e., independent of the original machine). To increase the testability for PLA-based machines, the work in [11] appends an additional hardware. However, hardware overhead in [11] is higher and it depends on the original machine. Three proposed techniques, the method in [9] and full scan technique are compared on benchmarks. First two techniques have low hardware overhead. First and third techniques have low test application time. Test application time of the second technique is larger in comparison to those of first and third, but it is less than that of full scan. Test length and hardware overhead are found to compare favorably with those of scan and previous non-scan approaches.

*The ratio of number of faults detected or proved redundant by a test algorithm to the total number of faults in a circuit is known as *fault efficiency*.

2. Preliminaries

The general model of a synchronous sequential machine is shown in Fig. 1, consisting of a combinational circuit (CC) and a state register (SR). The machine has n primary input (PI) fed by Binary variables $x_1, x_2,\ldots, x_n$. The outputs [inputs] $y_1, y_2,\ldots, y_k$ [$Y_1, Y_2,\ldots, Y_k$] of k memory elements of SR define the present [next] state of the machine. The behavior of the machine is described by the state transition diagram (STG). We assume that the machine has a reset state. Given an input, *transition from a state* S_i means the state transitions and change in primary output (PO) lines, if that input is applied in the machine with state S_i. If a fault f in a sequential machine changes the transitions from a state S_i, then to detect f, we have to first initialize the machine to state S_i (called *initialization state* for f). To do this, we have to apply a sequence of vectors (known as *justification sequence* [5] of S_i), application of which to the machine in the reset state, changes the state to S_i. However, there may not exist any such justification sequence for a state S_i, as there may exist some states in the machine that are unreachable or cannot be reached in sufficient time (hard to reach) from the reset state. But some of these may be needed for initialization for testing. A state that cannot be reached or which is hard to reach from reset state is known as an *invalid state,* else it is a *valid state*. The list of invalid and valid states in a machine can be known from STG of the machine. To detect a fault, after the application of justification sequence, we to apply another sequence of vectors known as differentiating sequence. A *differentiating sequence* [5] for a pair of states S_j and S_k, in a sequential circuit is a minimal length sequence of input vectors, such that the output response obtained by applying the sequence when the circuit is initially in S_j, is different from that obtained when the circuit is initially in S_k.

Let us extract the combinational circuit from a sequential machine, by replacing inputs [outputs] of SR by pseudo primary outputs (PPOs) [pseudo primary inputs (PPIs)]. Then this combinational circuit is known as the *combinational test generation model (CTGM)* of the sequential machine. For example, CTGM of the machine of Fig. 1 is shown in Fig. 2. Given the CTGM of a sequential machine, we try to generate the test vectors for this CTGM. Obviously each test vector is an ordered (n+k)-tuple, corresponding to n PIs and k PPIs. A state in a machine is called a *test state,* if it appears in PPI lines of any test vector of CTGM of the machine. A state of a machine which is a test state and also a valid [an invalid] state is known as *valid test state* [*invalid test state*]. In general, the number of test states is much smaller compared with the total number of states, 2^k.

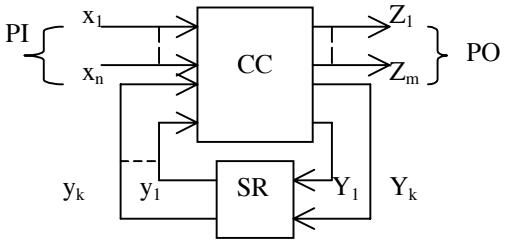

Fig. 1: The general model of a sequential machine

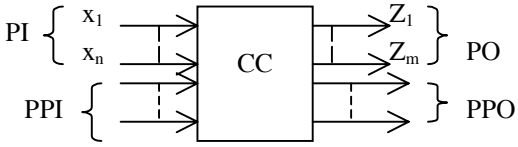

Fig. 2: CTGM of the machine of Fig. 1

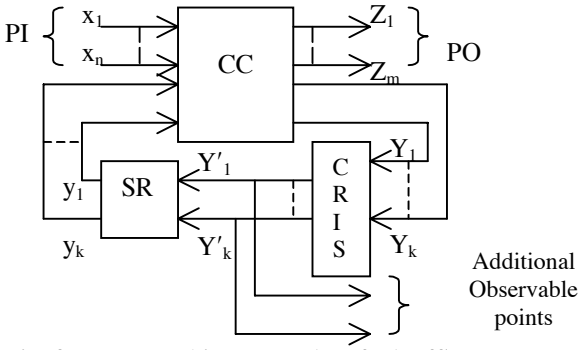

Fig. 3: DFT to achieve complete fault efficiency (Technique 1)

3. New DFT designs

From STG of the machine, first we find the set of valid and invalid states. Then we use combinational ATPG algorithm to find the set of test vectors of the CTGM. If such a test vector is a valid test state, then this state can be reached from the reset state. But if it is an invalid test state (i.e, it is unreachable from the reset state), the value of PPIs cannot be set to the SR using state transitions of the machine. The problem of state initialization to an unreachable state poses a major problem in the test generation of sequential circuits. In our designs, we adopt a new technique to reach these unreachable states. Notice that to test a circuit, we need not reach all invalid states, reaching only to invalid test states are sufficient.

3.1 The first technique

In our design to set the invalid test states to the SR, we append an extra logic called as CRIS (circuit to reach invalid states) to the original machine that generates all invalid test states of the machine. The DFT scheme is shown in Fig. 3. CRIS has the inputs as the next state lines of the original machine. In a similar approach recently [9], an additional circuit was also used to reach these invalid test states, where primary inputs are used as inputs to the extra logic.

(a) Designing CRIS: Let V [S_{ITS}] denotes the set of valid [invalid test] states in the machine. Then, any state $S_i \in V$ [S_{ITS}] can [cannot] appear in next state lines by proper [any] transition from reset state. CRIS makes also the appearance of S_{ITS} at the inputs to SR. For this, CRIS takes PPOs (Y_1, Y_2, …, Y_k) as the inputs, and produces (Y'_1, Y'_2,…, Y'_k) as inputs to SR using some control inputs. The output of CRIS is the same as input when control inputs are at logic 0, and when one or more of them is 1, it produces some invalid test state. Optimization of number of control inputs and hardware of CRIS is an open problem. Here, we follow a heuristic approach. For each state $S_i \in S_{ITS}$, we first find how S_i can be produced from each state $S_j \in V$. For example, say an invalid test state 0101 can be produced from a valid state 0011 by complementing 2nd and 3rd bits of 0011. This can be done by ORing Y_2 and ANDing Y_3 with a control input C, as shown in Fig. 4. For each state $S_i \in S_{ITS}$, we find how S_i can be produced from any state in V in this manner. Among these different possible productions, we implement one that with minimum hardware. If different invalid test states require same bit to be complemented, we use the same control line. If any line requires both ANDing and ORing, we replace the gate by XOR.

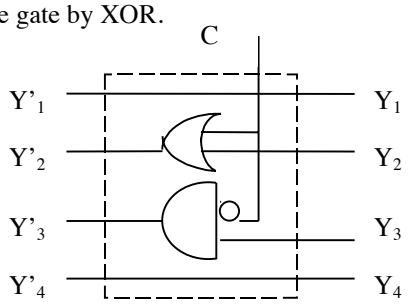

Fig.4: An example of CRIS

(b) Number of control inputs and hardware overhead of CRIS: Theoretically, the number of such control lines requirement can be maximum k, and that happens when there is only one valid state in the machine and there are at least 2^{k-1} invalid test states. But practically, as number of invalid test states is much smaller (can be at most the number of test states) in comparison to total number ($=2^k$) of states, and that is not very high in comparison to number of valid states, control lines requirement and hardware overhead cannot be high. By our heuristic approach, benchmarks result depicts significantly low overhead.

(c) Testing of CRIS: To detect a fault, the machine is initialized to its test state by justification sequence and controlling control inputs of CRIS. As all next state lines are observable, a state reached at next state lines after justification sequence can be identified. Thus, any fault in CRIS is also detected.

(d) Short test application time: When an initialization state S_i for a fault is properly reached in present state lines of SR, hold mode is activated where the state of the machine is kept at S_i, independently of the inputs at PIs. As several faults may have same test state S_i, for all such faults test vectors are applied consecutively holding the machine at state S_i. Moreover, with the observation of next state lines, the length of differentiating sequence is always null. Use of CRIS to reach unreachable states, use of no differentiating sequences and use of hold mode to avoid the repeated application of same justification sequence highly reduce test application time.

(e) Short test generation time and complete fault efficiency: Use of combinational ATPG tool decreases test generation time. Use of this tool and use of CRIS to ensure the machine to reach any test state make fault efficiency to be 100%.

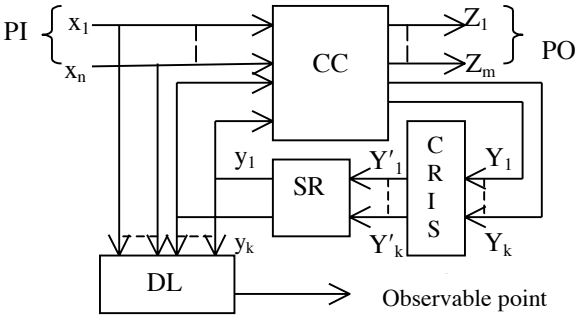

Fig. 5: DFT with complete fault efficiency and less observable pints (Technique 2)

3.2 The second technique

The drawback of the first technique is that it requires k additional observable points. To reduce the number of observable points, we use one more additional circuit known as differentiating logic (DL). The complete scheme is shown in Fig. 5.

(a) Design of (DL): Two cases need to be considered.

Case 1: $k \leq n$: In this case, DL has one output, given by $F = x_1y_1 + \bar{x}_1\bar{y}_1 + x_2y_2 + \bar{x}_2\bar{y}_2 + \ldots\ldots + x_ky_k + \bar{x}_k\bar{y}_k$. The circuit to realize F is shown in Fig. 6. The function F has a unique property. For every combination of (y_1, y_2,…, y_k), $y_i \in (0,1)$, the sub-function contains a unique pattern in x_is, such that for a pattern (y_1, y_2,…, y_k) at PPIs, if we apply a pattern X at PIs with (x_1, x_2,…, x_k) = ($\bar{y}_1$, $\bar{y}_2$,…, $\bar{y}_k$), we get the output of DL as 0, and for any other pattern at PI the output is at logic 1. It implies that if the machine reaches a state $S_i(y_1, y_2,\ldots, y_k)$, then by applying a single input pattern, obtained by complementing each bit of (y_1, y_2,…, y_k), this state can be uniquely identified. That is, differentiating sequence of any two states is of unit length.

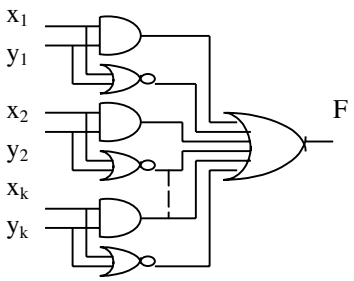

Fig. 6: Differentiating Logic (DL)

Example 1: The K-map for k=3 is shown in Fig. 7. Variables x_i's (y_i's) are used to label the map horizontally (vertically). A horizontal line in k-map corresponds to a state. Note that any state can be uniquely identified by a single input pattern. For example, a state $(y_1,y_2,y_3) = (010)$, can be uniquely identified by the vector $(x_1,x_2,x_3)=(101)$.

Case 2: k > n: DL has r (= $\lceil k/n \rceil$) outputs, and each output line realizes F_i ($1 \leq i \leq r$) s.t., $F_{j+1} = x_1 y_{jn+1} + \bar{x}_1 \bar{y}_{jn+1} + x_2 y_{jn+2} + \bar{x}_2 \bar{y}_{jn+2} + \ldots\ldots + x_a y_{jn+a} + \bar{x}_a \bar{y}_{jn+a}$ where a = n for $(0 \leq j < r-1)$, and a=k-(r-1)n for j=r-1. If a is found to be 1, then we replace F_{j+1} by y_{jn+1}.

(b) DL is universal: Design of DL is dependent only on the number of PIs and flip-flops in the circuit, i.e., it is universal, not dependent on the circuit structure. Thus, any fault in DL does not interfere with the original circuit behavior.

(c) Use of Hold mode: It is used to identify a state. If a state $(y_1,y_2,\ldots,y_k)$ is expected at present state lines, we activate hold mode and apply an input for case 1 (input sequence for case2) at PIs such that $x_i = \bar{y}_i \; \forall$ i $(1 \leq i \leq n)$. If the output of DL is 0, then the state of the machine is identified as the expected state.

(d) Techniques to achieve low test application time: As differentiating sequence is of length r = $\lceil k/n \rceil$, test application time is greatly reduced, which is n in case of full scan. To decrease it further, we adopt following technique. Say, to detect a fault, the machine is initialized to a state S_i. Now, application of the test vector may change the state to S_j. If S_j is a test state, we use it as an initialization state of another fault. If S_j is not a test state or there is no other fault left to be detected with initialization state S_j, then we attempt to initialize the machine to any other test state. Compared with full scan, our method results low test application time in benchmarks.

(e) Testing of CRIS and DL: Any fault in DL or CRIS can be detected, by observing the output of DL.

(f) Hardware overhead: It equals to (2k+r) gates, where is less than that of full scan for r < n-1.

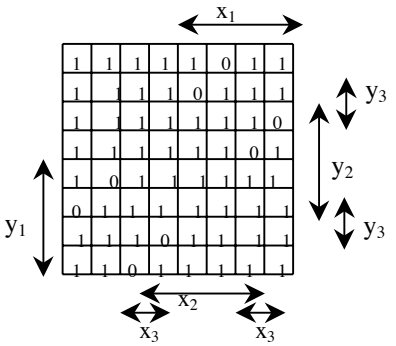

Fig. 7: K-map of DL for k=3 and n ≥ 3

3.3 The third technique

Drawback of second technique is that as observable points use present state lines, we cannot use the same justification sequence for different faults having same initialization state. To avoid this, the third technique is proposed, where a register R is used to load the values of the next state lines and outputs of R are fed into DL. The complete scheme is shown in Fig. 8. Use of hold mode is similar to that of first technique.

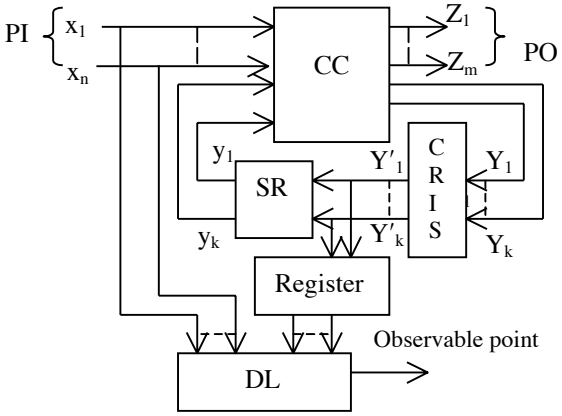

Fig. 7: DFT Design of Technique 3

4. Experimental Results

General performance of the DFT Design can be described as in Table 1. Rows "scan", "ATS-98", "case1", "case2" and "case3" represent full scan, the method in [9], technique-1, technique-2 and technique-3 respectively. O(ISG) and O(CRIS) indicate the overhead of invalid state generator (ISG) in the paper of [9] and that of CRIS of this paper respectively. It is found experimentally that O(CRIS) < O(ISG). O(CRIS) was found to be maximum of two two-input gates in MCNC benchmarks. The value c denotes the number of control inputs needed for CRIS and r equals to $\lceil k/n \rceil$, where n and k are the number of PIs and flip-flops in the machine respectively. In most cases of benchmarks, r is found to be 1 and c is 1, except in two cases, where it is found to be 2.

Table 1: Overall comparison

Method	overhead		TG	TA	At-speed
	Pin	Area	time	time	
scan	3	3k gates	low	high	Not possible
ATS-98	k+2	O(ISG)	low	low	possible
T-1	k+1+c	O(CRIS)	low	low	possible
T-2	r+1+c	O(CRIS)+ (2k+r) gates	low	higher than T-1, less than scan	possible
T-3	r+1	O(CRIS)+ (9k+r) gates	low	low	possible

Table 2: STG characteristic

name	#PIs	#POs	#States	#FFs
bbara	4	2	10	4
bbsse	7	7	16	4
bbtas	2	2	6	3
beecount	3	4	7	3
cse	7	7	16	4
dk14	3	5	7	3
dk15	3	5	4	2
dk16	2	3	27	5
dk17	2	3	8	3
ex1	9	19	20	5
ex2	2	2	19	5
ex3	2	2	10	4
ex4	6	9	14	4
ex5	2	2	9	4
ex6	5	8	8	3
ex7	2	2	10	4
keyb	7	2	19	5
kirkman	12	6	16	4
lion	2	1	4	2
lion9	2	1	9	4
mc	3	5	4	2
opus	5	6	10	4
planet	7	19	48	6
planet1	7	19	48	6
pma	8	8	24	5
s1	8	6	20	5
s1488	8	19	48	6
s1494	8	19	48	6
s208	11	2	18	5
s27	4	1	6	3
s298	3	6	218	8
s386	7	7	13	4
s420	19	2	18	5
s510	19	7	47	6
s820	18	19	25	5
s832	18	19	25	5
sand	11	9	32	5
sse	7	7	16	4
styr	9	10	30	5
tav	4	4	4	2
tbk	6	3	32	5
tma	7	6	20	5
train11	2	1	11	4
train4	2	1	4	2

Experimental results on benchmarks are also shown. Benchmark specifications are shown in Table 2. AutoLogic II (Mentor Graphics) tool synthesizes the circuits from MCNC benchmarks [10]. Columns "name", "#PIs", "#POs", "#states", "#FFs" denote the name, the number PIs, POs, states, and flip-flops of the original sequential machines respectively. In benchmark results, we show only those cases when number of inputs (n) > 1. For n=1, we apply only the first technique of our DFT designs.

Table 3: Hardware/pin overhead

	Hardware Overhead (*gates*)					Pin Overhead				
name	scan	ATS98	case1	case2	case3	scan	ATS98	case1	case2	case3
bbara	12	12	1	10	38	3	6	6	3	3
bbsse	12	12	1	10	38	3	6	6	3	3
bbtas	9	10	1	6	27	3	5	5	4	4
beecount	9	9	1	8	29	3	5	5	3	3
cse	12	0	0	9	37	3	5	5	2	2
dk14	9	9	1	8	29	3	5	5	3	3
dk15	6	0	0	5	19	3	3	3	2	2
dk16	15	31	1	11	46	3	7	7	5	5
dk17	9	0	0	5	26	3	4	4	3	3
ex1	15	15	1	12	47	3	7	7	3	3
ex2	15	34	2	12	47	3	7	8	6	6
ex3	12	20	1	11	39	3	6	6	4	4
ex4	12	12	1	10	38	3	6	6	3	3
ex5	12	20	1	11	39	3	6	6	4	4
ex6	9	0	0	7	28	3	4	4	2	2
ex7	12	20	2	12	40	3	6	7	5	5
keyb	15	15	1	12	47	3	7	7	3	3
kirkman	12	0	0	9	37	3	5	5	2	2
lion	6	0	0	5	19	3	3	3	2	2
lion9	12	20	1	11	39	3	6	6	4	4
mc	6	0	0	5	19	3	3	3	2	2
opus	12	12	1	10	38	3	6	6	3	3
planet	18	18	1	14	56	3	8	8	3	3
planet1	18	18	1	14	56	3	8	8	3	3
pma	15	15	1	12	47	3	7	7	3	3
s1	15	15	1	12	47	3	7	7	3	3
s1488	18	18	1	14	56	3	8	8	3	3
s1494	18	18	1	14	56	3	8	8	3	3
s208	15	15	1	12	47	3	7	7	3	3
s27	9	9	1	8	29	3	5	5	3	3
s298	24	165	1	20	76	3	10	10	5	5
s386	12	12	1	10	38	3	6	6	3	3
s420	15	15	1	12	47	3	7	7	3	3
s510	18	18	1	14	56	3	8	8	3	3
s820	15	15	1	12	47	3	7	7	3	3
s832	15	15	1	12	47	3	7	7	3	3
sand	15	0	0	11	46	3	6	6	2	2
sse	12	12	1	10	38	3	6	6	3	3
styr	15	15	1	12	47	3	7	7	3	3
tav	6	0	0	5	19	3	3	3	2	2
tbk	15	0	0	11	46	3	6	6	2	2
tma	15	15	1	12	47	3	7	7	3	3
train11	12	16	1	11	39	3	6	6	4	4
train4	6	0	0	5	19	3	3	3	2	2

Table 3 shows hardware and pin overhead. Hardware overhead of first technique is lowest and significantly small. Hardware overhead of both first and second technique is smaller than that of full scan. The third technique needs more hardware as an additional register of k flip-flops (k= # of flip-flops) are used. We have considered 7 gates per flip-flop in third technique. In the techniques 2 & 3, number of gates are decreased by 3 from that given in the formula of Table 1, if the remainder in dividing k by n be 1. Pin overhead of proposed second and third techniques are same and in most cases it equals to that of full scan technique which is always 3. The first technique, requires more number of pins and it is same as that in the method of [9]. Test generation and application time for different methods are shown in Table 4. A combinational/sequential test generation tool

TestGen (Sunrise) is used. Results show that test generation time is almost equal in five cases. Test application time is highest in case of full scan method. This time is almost equal in the method of [9], first technique and third technique. Second technique requires larger test application time in comparison to those of first and third techniques, but this time is short in comparison to that of full scan.

Table 4: Test generation/application time

	Test Generation Time (*sec.*)					Test Application Time (*cycles*)				
name	scan	ATS98	case1	case2	case3	scan	ATS98	case1	case2	case3
bbara	0.32	0.32	0.36	0.39	0.37	339	88	86	171	89
bbsse	0.58	0.58	0.60	0.63	0.63	399	101	106	250	105
bbtas	0.06	0.06	0.05	0.06	0.06	71	27	29	43	54
beecount	0.19	0.19	0.21	0.19	0.26	199	60	63	69	79
cse	1.08	1.08	1.16	1.13	1.19	584	144	142	323	153
dk14	0.13	0.13	0.15	0.15	0.16	199	60	69	56	68
dk15	0.10	0.10	0.05	0.09	0.12	110	42	37	35	35
dk16	0.35	0.35	0.43	0.41	0.46	593	148	151	408	376
dk17	0.06	0.06	0.09	0.09	0.14	127	47	46	74	90
ex1	4.58	4.58	5.07	5.18	5.14	1679	323	335	838	340
ex2	0.27	0.27	0.29	0.24	0.35	461	119	107	196	307
ex3	0.12	0.12	0.12	0.14	0.17	249	72	61	84	133
ex4	0.31	0.31	0.32	0.33	0.29	294	76	77	212	83
ex5	0.11	0.11	0.10	0.15	0.15	244	71	64	90	116
ex6	0.46	0.46	0.45	0.50	0.47	243	70	71	69	69
ex7	0.08	0.08	0.13	0.18	0.12	194	60	61	33	123
keyb	4.90	4.90	5.18	5.20	5.47	1481	282	307	634	316
kirkman	13.88	13.88	12.79	12.89	12.76	2409	499	499	2796	508
lion	0.07	0.07	0.09	0.05	0.07	56	24	23	23	22
lion9	0.21	0.21	0.21	0.19	0.20	259	69	67	180	139
mc	0.06	0.06	0.06	0.06	0.10	44	20	19	22	23
opus	0.39	0.39	0.36	0.42	0.43	394	106	105	289	110
planet	4.14	4.14	4.13	4.25	4.38	1539	400	392	2002	407
planet1	4.28	4.28	4.02	4.18	4.12	1539	400	392	2002	407
pma	1.30	1.30	1.36	1.38	1.40	971	204	204	428	208
s1	5.19	5.19	5.14	5.23	5.05	1283	269	291	889	288
s1488	20.27	20.27	20.35	20.66	20.88	3051	615	660	5571	647
s1494	21.60	21.60	22.07	21.87	20.91	3065	645	677	4379	650
s208	9.35	9.35	7.85	8.11	7.80	1535	289	313	1645	319
s27	0.24	0.24	0.28	0.35	0.38	191	59	66	44	74
s298	82.40	82.40	79.58	83.91	86.98	9746	2429	2516	25924	5031
s386	1.23	1.23	1.31	1.27	1.29	554	131	144	404	134
s420	9.42	9.42	7.70	8.38	7.83	1439	273	305	1896	305
s510	1.06	1.06	1.13	1.22	1.21	930	195	194	1899	201
s820	16.09	16.09	16.39	16.24	16.26	2273	450	507	3010	484
s832	17.25	17.25	17.63	17.95	18.01	2225	438	516	2443	525
sand	9.26	9.26	8.87	8.72	9.12	1547	308	301	691	324
sse	0.61	0.61	0.58	0.60	0.58	399	101	106	250	105
styr	5.60	5.60	5.44	5.52	5.51	1385	296	319	793	309
tav	0.24	0.24	0.24	0.29	0.24	119	45	42	52	46
tbk	51.75	51.75	49.86	49.64	50.14	4469	793	780	1864	783
tma	0.85	0.85	0.88	0.90	0.99	623	151	157	252	162
train11	0.23	0.23	0.26	0.23	0.24	264	74	83	76	140
train4	0.05	0.05	0.05	0.05	0.07	41	19	18	12	21

5. Conclusions

The paper suggests three new techniques on non-scan DFT. As state initialization is a major problem in testing of sequential circuits, it solves that problem by using an additional hardware called as CRIS (circuit to reach invalid states). It is found experimentally that hardware overhead of CRIS is also low. The techniques use combinational ATPG tool to find the test sequences of the machine. Among the three techniques, hardware overhead of the first technique is the lowest, but it requires k additional observable points. To decrease the number of observable points, a notion of differentiating logic (DL) is proposed in technique 2. Even with the use of this DL, hardware overhead is less than that of full scan. Use of this DL increases test application time in comparison to that of first technique, but this time is less than that of full scan. To achieve the test application time, same as that of first technique, an additional register is used in third technique. The novelty of these techniques is that they guarantee complete fault efficiency with at-speed testing. Hardware overhead, test generation time and test application time compare favorably with those of earlier designs. Future work includes the application of the same technique for larger circuits.

Acknowledgement: Debesh Kumar Das is supported by JSPS-INSA fellowship during the preparation of this work. Satoshi Ohtake is under JSPS research fellowship. This work was supported in part by Semiconductor Technology Academic Research Center (STARC) under the Research Project and in part by the Ministry of Education, Science, Sports and Culture, Japan under Grant-in-Aid for Scientific Research B(2) (no.09480054). Authors would like to thank Toshimitsu Masuzawa, Tomoo Inoue, and Michiko Inoue of Nara Institute of Science and Technology for their helpful discussion.

References:

1. H. Fujiwara, *Logic Testing and Design for Testability,* The MIT Press, 1985.
2. M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design,* W. H. Freeman & Co., New York 1990.
3. S. T. Chakradhar, A. Balkrishnan and V. D. Agrawal, "An exact algorithm for selecting partial scan flip flops," *Proc. DAC-94,* pp. 81-86.
4. P. S. Parikh and M. Abramovici, "A cost based approach to partial scan," *Proc. DAC-93.*
5. S. Devadas and K. Keutzer, "A unified approach to the synthesis of fully testable sequential machines," *TCAD,* vol.10, pp. 39-50, 1991.
6. V. Chickermane, E. M. Rudnick, P. Banerjee and J. H. Patel, "Non-scan design-for-testability techniques for sequential circuits," *Proc. DAC-93,* pp. 236-241.
7. I. Pomeranz and S. M. Reddy, "Design for testability for sequential circuits using locally available lines," *Proc. DATE-98,* pp. 983-984.
8. D. K. Das and B. B. Bhattacharya, "Testable design of non-scan sequential circuits using extra logic," *Proc. ATS-95,* pp. 176-182.
9. S. Ohtake, T. Masuzawa and H. Fujiwara, "A non-scan DFT method for controllers to achieve complete fault efficiency," *Proc. ATS-98.*
10. S. Yang, "Logic synthesis and optimization benchmarks user guide," *Technical Report 1991-IWLS-UG-Saeyang, Microelectronics Center of North Carolina.*
11. S. T. Chakradhar, S. Kanjilal, V. D. Agrawal, "Finite state machine synthesis with fault tolerant test function," *Journal of Electronic Testing: Theory and Applications,* pp. 57-69, 1993.