International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Efficient Hardware Utilization for Functional Broadside Test to Achieve High Fault Coverage

P. Durga Venkata Prasad[1], K. Tirumala Rao[2]

[1]P.G Student of Department of Electronics and Communication Engineering, NCET, Jupudi, Ibrahimpatnam JNTUK, Kakinada, Andhra Pradesh, India

[2]Assistant Professor of Department of Electronics and Communication Engineering, NCET, Jupudi, Ibrahimpatnam JNTUK, Kakinada, Andhra Pradesh, India

Abstract: ***Functional broadside tests are two-pattern scan-based tests that avoid over testing by ensuring that a circuit traverses only reachable states during the functional clock cycles of a test. On-chip test generation has the added advantage that it reduces test data volume and facilitates at-speed test application. This paper shows that on-chip generation of functional broadside tests can be done using simple hardware, and can achieve high transition fault coverage forte stable circuits. With the proposed on-chip test generation method, the circuit is used for generating reachable states during test application. This alleviates the need to compute reachable states off-line***

Keywords: Built-in test generation, functional broadside tests, reachable states, transition faults.

1. Introduction

Over testing due to the application of two-pattern scan-based tests was described in [1]-[3]. Over testing is related to the detection of delay faults under nonfunctional operation conditions. When an arbitrary state is used as a scan-in state, a two-pattern test can take the circuit through state-transitions that cannot occur during functional operation. As a result, slow paths that cannot be sensitized during functional operation may cause the circuit to fail [1]. In addition, current demands that are higher than those possible during functional operation may cause voltage drops that will slow the circuit and cause it to fail [2]-[3]. In both cases, the circuit will operate correctly during functional operation.

Functional broadside tests [9] ensure that the scan in state is a state that the circuit can enter during functional operation, or a reachable state. As broadside tests [4], they operate the circuit in functional mode for two clock cycles after an initial state is scanned in. This results in the application of a two-pattern test. Since the scan-in state is a reachable state, the circuit goes through state transitions that are guaranteed to be possible during functional operation. Delay faults that are detected by the test can also affect functional operation. This alleviates the type of over testing described in [1]-[3].Test generation procedures for functional and pseudo-functional scan-based tests were described in [5]-[13].

The procedures generate test sets for application from an external tester. Functional scan-based tests use only reachable states as scan-in states. pseudo functional scan _based test useful functional constraints avoid unreachable states that are captured by constraints. This work considers the on-chip (or built-in) generation of functional broadside tests. On-chip test generation reduces the test data volume and facilitates at-speed test application. On-chip test generation methods for delay faults, such as the ones described in [14] and [15], do not impose any constraints on the states used as scan-in states. Experimental results indicate that an arbitrary state used as a scan-in state is unlikely to be a reachable state. The on-chip test generation method from [16] applies pseudo-functional scan-based tests. Experimental results indicate that pseudo-functional tests or not sufficient for avoiding unreachable states as scan-in states. The on-chip test generation process described in this work guarantees that only reachable states will be used.

Under the proposed on-chip test generation method, the circuit is used for generating reachable states during test application. This alleviates the need to compute reachable states or functional constraints by an off-line process as in [5]-[13] and [16]. The underlying observation is related to one of the methods used in [9] for external test generation, and is the following. If a primary input sequence A is applied in functional mode starting from a reachable state, all the states traversed under Aare reachable states. Any one of these states can be used for the application of a functional broadside test. By generating on-chip and ensuring that it takes the circuit through a varied set of states, the on-chip test generation process is able to achieve high transition fault coverage using functional broadside tests based on A.

When the circuit-under-test is embedded in a larger design, its primary inputs may be driven by other logic blocks that are part of the same design. In addition, the primary inputs of the circuit-under-test include any external inputs of the design that drive the circuit-under-test. The primary outputs of the circuit-under-test may drive other logic blocks, or they may be primary outputs of the complete design. For simplicity this paper assumes that primary inputs can be assigned any combination of values. The paper is organized as follows. Section II Gives an overview of the on-chip generation and application of functional broadside tests. Section III Describes the details. Section IV Presents experimental results demonstrating the achievable fault coverage.

2. Overview

This section gives an overview of the proposed method. This paper assumes that the circuit is initialized into a known state before functional operation starts. Initialization may be achieved by applying a synchronizing sequence as in [5]-[7] and [9]-[12], by asserting a reset input as in [8] and [13], or by a combination of both. We denote the initial state of the circuit by s_r with s_r as the initial state for functional operation, the set of reachable states consists of every state s_i such that there exists a primary input sequence that takes the circuit from s_r to s_i. Since s_i can be entered during functional operation starting from s s_r, s_i is a reachable state. It is possible to obtain reachable states on-chip by placing the circuit in state s_r and applying a primary input sequence A=a(0)a(1)...a(L−1) of length Lin functional mode.

The circuit can be brought into state s_r by using a scan-in operation, or by using its initializing sequence. Lets(u) be the state that the circuit reaches at time unit u under A, for 0≤u≤L. We have that s(0)=s_r. In addition, s(u) is a reachable state for 0≤u≤L.Therefore, every states(u) can be used as a basis for a functional broadside test <s(u), a1, a2>, where s(u)plays the role of a scan-in state. As in a broadside test,a1

And a_2 are primary input vectors that are applied in two consecutive functional clock cycles starting from s(u) using a slow and a fast clock, respectively. Every subsequence of length two of A defines a functional broadside test t(u)=<s(u), a(u), a(u+1)>.By using a(u) and a(u+1) from A, it is possible to avoid the need for a different source for these primary input vectors during on-chip test generation.

For illustration we consider s27 with initial state s_r =000. The circuit is shown in Figure 1. A primary input sequence for the circuit is shown in Table I. For every time unit u, Table I shows the state s(u) and the primary input vector a(u). The other columns of Table I will be explained later. Table I yields the functional broadside tests t(0)=<000, 1001, 1000>,t(1)=<010, 1000, 1100>,...,t(14)=<101, 1000, 1111>.

The proposed on-chip generation method of functional broadside tests is based on placing the circuit in the initial state s_r, applying a primary input sequence A, and using several of the functional broadside tests that can be extracted from A in order to detect target faults. Next, we discuss how the application of A is affected by the need to observe fault effects created by a test t(u)=<s(u),a(u),a(u+1)>.At time unit u the circuit is in state s(u). Applying a(u) and a(u+1) in functional mode will result in the application of t(u). A fault can be detected in one of two ways. (1) Based on the primary output vector z(u+1) obtained in response to a(u+1), if this vector is different from the expected fault free primary output vector.(2)Based on the final state s(u+1) of the test,

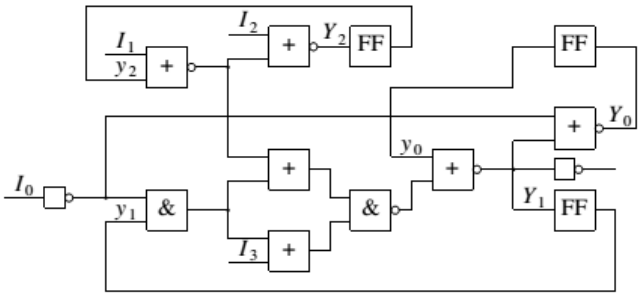


Figure 1: .s27

TABLE I. Primary input sequence fors27

u	$s(u)$	$a(u)$	RS_1	RS_2
0	000	1001	0011	3
1	010	1000	1010	2
2	100	1100	1100	1
3	101	1101	1111	2
4	101	1001	0011	3
5	101	0110	0110	0
6	000	1100	0100	3
7	101	1011	1011	0
8	100	1001	0011	3
9	100	1100	0110	2
10	101	1001	1001	2
11	101	1001	0011	1
12	101	1100	1110	2
13	101	1001	1011	2
14	101	1000	0000	2
15	101	1001	0011	2

If this state is different from the expected fault Free State. In the context of built-in self-test, z(u+1) ands(u+2) need to be captured by an output response compactor such as a MISR[17]. In the case of s(u+2), the state needs to be scanned out and shifted into the output response compactor over a number of clock cycles equal to the length of the longest scan chain. The circuit then needs to be brought back to states(u+2) in order to continue the test application process under A. Bringing the circuit back to states(u+2) can be done by using circular shift of s(u+2) [17]. As s (u+2) is scanned out, it can also be scanned in. If s(u+2) is faulty, the output response compactor will capture the fault effect, and observation of the final signature will indicate that a fault is present. If s(u+2) is fault free, the remaining tests based on A will be applied as required. The application of overlapping tests based on A requires special hardware. To avoid it, this paper focuses on subsets of non-overlapping tests of the form {t (u_0), t (u_1)... t (u_k−1)}, whereu_i+1<u_i+1for 0≤i<k−1.

3. On-Chip Generation of Functional Broadside Tests

This section describes the on-chip generation method for functional broadside tests based on the concepts discussed in Section II. It first describes the generation of the sequence A. It then describes the selection of tests that will be applied based on A. Finally it describes the overall on-chip test generation process. A.The Primary Input Sequence A The simplest way to generate a primary input sequence Aon-chip

is to use a random source such as an LFSR. However, a random sequence A may bring the circuit from the initial state s_r into a limited set of reachable states. This can be explained by the effect observed in [18] and referred to as repeated synchronization. According to [18], a primary input cube c synchronizes a sub set of state variables S(c) if the following conditions are satisfied. Let c be applied to the primary inputs when the circuit is in the all-unspecified present-state. Suppose that this results in a next-state s. The state variables whose values are specified ins are included in S(c).In the example of s27 shown in Figure 1, the primary input cube $i_0i_1i_2i_3$=0xxx applied in present-state $y_0y_1y_2y_3$=xxx results in the next-stateY0Y1Y2=0xx, synchronizing state variable y0.

In addition, the primary input cube $i_0i_1i_2i_3$= xx1x applied in present-state$Y_0y_1y_2$=xxx results in the next-state$y_0y_1y_2$=xx0, synchronizing state variabley2.A primary input cube c with a small number of specified values is likely to appear repeatedly in a random primary input sequence A. When this happens, the state variables in S(c) assume the same values repeatedly under A. This may prevent the circuit from entering certain reachable states, and limit the ability of the functional broadside tests extracted from A to detect target faults. Repeated synchronization was avoided in [18] by using a software procedure.

Here, we need a process that will be implemented using on-chip hardware. To design such hardware, we first use a software procedure to compute primary input cubes that synchronize one or more state variables. The procedure focuses on primary input cubes with single specified values since such cubes are most likely to appear repeatedly in a random primary input sequence. For a circuit with n primary inputs,I0,I1,..., In−1, it considers every primary input cube c_j, v where primary input I_j assumes the value v, for $0 \le j < n$ and $v \in \{0,1\}$, and the other primary inputs are unspecified. For c $_{j,}$v it computes the set of synchronized state variables $S(_{cj,}v)$. It then combines all the primary input cubes into a single cube c=c(0) (1)..(n−1) as follows. (1) If |S(cj,0)| <|S(cj,1)|, setting Ij =0 causes fewer next-state variables to be specified. The procedure sets c(j)=0. (2) If |S(cj,0)| >|S(cj,1)|, setting Ij =1causes fewer next-state variables to be specified. The procedure sets c(j)=1. (3) If |S(cj,0)|=|S(cj,1)|, the procedure setsc(j)=x.

In the example of s27 this procedure yields S(c0,0)={y0},S(c2,1)={y2}, and S(c j,a)=φ in all the other cases. Therefore, it yields c=1x0x.We u sec in the hardware generation of A=a(0)a(1)...a(L−1) as follows. A random source called RS1 is used for generating a sequence of n-bit random primary input vectors. The length of the sequence is a constant L. A second random source called RS2 is used for generating random numbers in the range $[0,2^p-1]$, for a constant p. At a time unit u whereRS2>0, a(u)is modified such that it is equal toc for all the primary inputs where c is specified. For illustration we consider the random primary input sequence for s27 shown in Table I under column RS1. Column RS2 shows the values produced by the second random source RS2 using p=2. We have c=1x0x for this circuit. Therefore, whenRS2>0 at time unit u, a(u) is modified by assigning a 1 to input 0 and a 0 to input 2. The values of the other inputs do not change. The resulting primary input sequence A is the one shown under column a(u) of Table I. Considering a circuit with n primary inputs and a primary input cube c=c(0)c(1)...c(n−1), a single (n+p)-bit LFSR can be used for generating A.

The logic required is shown in Figure 2. The LFSR is shown at the top of the figure. The value in cell j of the LFSR is denoted by lf s_r(j). Then leftmost cells of the LFSR are used for generating a random n-bit primary input sequence. The p rightmost cells of the LFSR are used for generating numbers in the range [0,2p−1]. Ap-input OR gate is used for generating a signal denoted by mod. When mod=0, or if c(j)=x, the value of primary input I_j is equal to l s f r(j). When mod=1, if c(j)≠x, the value of primary input I j is equal to c(j). Figure 2 shows three types of inputs. (1) For Ij1, c(j1)=x. In this case,Ij1=lfsr(j1). (2) For Ij2, c(j2)=0. In this case, Ij2=mod′. Lf s_r(j2)+mod. c(j2)= mod′. Lf s_r(j2). (3) For Ij3,c(j3)=1. In this case,Ij3=mod′. Lf s_r(j3)+mod. c(j3)=mod′. Lf s_r(j3)+mod=lf s_r(j3)+mod.

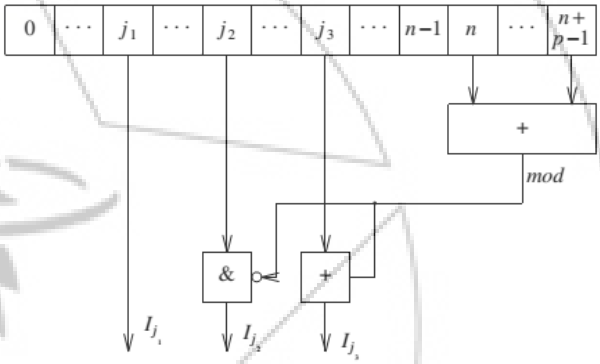


Figure 2: On-chip generation of A

In general, if there are N primary inputs Ij with c(j)≠ x, the implementation of Figure 2 requires an (n+p)-bit LFSR, ap-input OR gate, N two-input AND or OR gates, and at most N inverters.

B. Test Selection

Next, we describe the selection of the functional broadside tests that will be applied based on a primary input sequence A. Let F be the set of target faults. In general, the goal is to select a subset of time units U={u0, u1,..., uk−1} and apply a test set T(U)={t(u_i):u_i∈U}, t(u_i)={s(u_i),a(u_i),a(u_i+1)}, $_{such}$ that the following conditions are satisfied. (1) To ensure that the tests are non-overlapping,u_i+1 <u_i+1for $0 \le i < k-1$.(2) It should be possible to produce the subset U on-chip efficiently. (3) The test set T(U) based on U should detect as many of the faults in F as possible. (4) U should be as small as possible. The simplest way to satisfy the first two conditions is to include in U all the even or all the odd time units. We denote the resulting subsets of time units by U even and U odd, respectively. With an even L, U even ={0, 2, 4,...,L−2} and U odd ={1, 3, 5,...,L−3}.Test application based on U even (U odd) can proceed as follows. A counter denoted by CNT=CNT(0)CNT(1)...CNT(m−1), where m=log2L, counts from 0 to L−1 to indicate the current time unit u of A. For even time units CNT(m−1)=0, and for odd time units CNT(m−1)=1. Thus, a single inverter or buffer

producing the function apply= CNT(m−1)′ or apply= CNT(m−1) is needed to identify even (odd) time units. When apply=1,t(u) is applied as a two-pattern test to the circuit. Application of t(u) includes

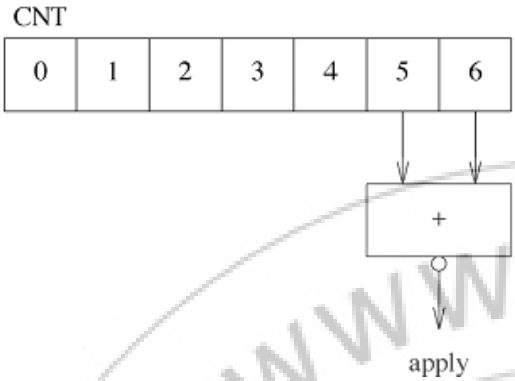


Figure 3: On-chip test selection.

Output response compaction of the primary output vector under the second pattern, and output response compaction of the final state of the test. The counter CNT is stopped during the scan out operation of the final state until the same state is restored through circular shift as discussed in Section II.

C. Selecting Seeds

The on-chip test generation hardware described so far has Parameters l, d and sel. These parameters determine the Primary input sequence, and the tests that will be applied Based on it. Keeping and sel constant in order to keep the hardware fixed, there is flexibility only in determining the seed of the LFSR. Different seeds yield different primary input sequences and different tests. Therefore, it is possible to increase the fault coverage by using several different seeds. To select seeds for a circuit it is possible to use an approach similar to the one used for test data compression [21].

Using a symbolic seed, it is possible to compute a primary input sequence and the subset of tests based on it, and then solve equations based on functional broadside tests that are known to detect target faults. The approach used in this paper avoids deterministic test generation to identify effective functional broadside tests by considering random seeds. A set of seeds is selected using the following process. Let be the set of target faults. For , the process selects a random seed denoted by . It computes the primary input sequence obtained with this seed. It also finds the subset of tests that will be applied based on .Both and are determined by the parameters and sel. In particular, is fixed by sel for all the values of . The process performs fault simulation of under and removes the detected faults from . If any faults are detected, it stores in a set denoted by SEED. It continues to consider additional seeds until the last seeds do not improve the fault coverage, for a constant . Computation of seeds requires fault simulation with fault dropping of under tests for every seed. As additional seeds are selected, the number of faults in is reduced, and the fault simulation effort decreases.

Let the set of selected seeds= {$seed_0$,$seed_1$.. $seed_{k-1}$}, be , with corresponding sequences A0,A1,A2,A_{K-1} and subsets of applied tests $T_0(U)$,T1(U)….$T_{K-1}(U)$. It is possible that all the faults $T_i(U)$ detected by will also be detected by $T_{i+1}(U)$,….$T_{K-1}(U)$. In this case $seed_i$, does not need to be used as part of the on-chip test generation process. The process of seed selection identifies and removes such sequences as follows. During the selection of the seeds, if a fault is detected by a test from , the process sets It also sets for . After all the seeds are selected, for , the process checks whether all the faults detected by are also detected by subsets such that and . This requires fault simulation of every fault such that under subsets such that and . If all the faults with are detected by other subsets, the process updates the variable for every such fault to reflect another subset based on which is detected. It also sets. At the end, the process removes from SEED every seed for which.

D. Selecting Parameter

The parameters of the on-chip test generation hardware, L, d and sel, are selected as follows. The value of mod determines the probability of avoiding repeated synchronization by modifying a primary input vector of a(u)based on the cube . Experimental results for benchmark circuits indicate that mod=3 that or 4 is suitable for all the circuits considered, and that decreasing or increasing mod beyond these values does not produce a higher fault coverage. We use when it is sufficient for matching or exceeding the fault coverage reported in [19], and mod=4 when mod=3 is not sufficient for this purpose. It was necessary to use mod=4 only for one circuit. Increasing and can potentially increase the fault coverage.

Increasing l and d increases the number of available tests, and increasing reduces the dependencies between the values of the primary inputs. Increasing sel can potentially decrease the fault coverage since it decreases the number of tests that will be applied to the circuit. Considering the values suitable for different circuits and the effects of increasing and decreasing these values, the values found to be effective are, or 8192,or ,and or 512. The value of determines the number of LFSR bits to be . This number is limited to be at most 300. The following process is applied in order to select the values of these parameters for a given circuit. Setting and , the process considers and 16. For every value of such that it selects seeds as described in the previous subsection using .It finds a fault coverage denoted by. Of all the values of, it selects the lowest one that yields the highest fault coverage. This is the value of selected for the circuit. Next, the process considers and 8192. For every value of it considers . For every value of and sel it selects seeds as described in the previous subsection using .It finds a fault coverage denoted by. The process continues to increase sel for each value of only as long as the fault coverage does not go down, or until sel reaches its upper bound of 512.Of the solutions obtained with, the following ones are reported.

Table 4: On-Chiptestgeneration

circuit	mod	d	L	sel	seq	tests app	eff	f.c.	gates	LFSR
s641	-	-	-	-	38	508	124	81.02	41	38
s641	3	4	1024	8	17	2176	138	81.02	4	140
s641	3	4	8192	2	4	16384	144	81.02	4	140
s1423	-	-	-	-	46	2659	234	83.27	48	20
s1423	3	16	2048	2	34	34816	269	83.70	3	272
s1423	3	16	2048	4	39	19968	260	83.35	3	272
s1423	3	16	8192	2	26	106496	269	83.63	3	272
s5378	-	-	-	-	96	2108	383	73.50	102	38
s5378	3	4	1024	4	48	12288	457	74.38	7	140
s5378	3	4	8192	2	25	102400	485	75.47	7	140
s35932	-	-	-	-	23	415	231	87.21	24	38
s35932	3	4	2048	2	4	4096	236	87.21	2	140
s35932	3	4	4096	512	56	448	230	87.21	2	140
s38584	-	-	-	-	312	13524	1755	66.30	313	15
s38584	3	8	1024	2	214	109568	1925	66.82	2	96
s38584	3	8	1024	4	251	64256	1872	66.46	2	96
s38584	3	8	2048	2	222	227328	1982	67.02	2	96
b04	-	-	-	-	34	326	160	84.54	35	15
b04	4	10	8192	2	4	16384	189	84.54	2	120
b04	4	10	8192	512	38	608	156	84.54	2	120
b08	-	-	-	-	24	374	88	81.87	25	13
b08	3	4	4096	256	36	576	89	81.87	2	40
b08	3	4	8192	2	4	16384	114	81.87	2	40
b09	-	-	-	-	16	183	53	77.29	17	5
b09	3	16	1024	32	15	480	61	77.29	2	32
b09	3	16	8192	2	7	28672	67	77.29	2	32
b10	-	-	-	-	22	240	102	81.72	23	15
b10	3	4	2048	128	31	496	102	81.72	2	48
b10	3	4	8192	4	3	6144	130	81.72	2	48
b11	-	-	-	-	27	540	143	75.79	28	11
b11	3	4	1024	8	24	3072	160	75.90	2	32
b11	3	4	4096	2	9	18432	173	76.99	2	32
b11	3	4	8192	2	12	49152	185	77.38	2	32
b14	-	-	-	-	179	7213	840	77.12	180	36
b14	3	8	1024	4	191	48896	1056	80.17	2	264
b14	3	8	2048	2	168	172032	1094	81.72	2	264
b14	3	8	8192	2	177	724992	1221	83.43	2	264
b20	-	-	-	-	407	8786	1873	81.97	408	36
b20	3	8	1024	4	403	103168	2331	84.33	2	264
b20	3	8	4096	2	349	714752	2774	85.88	2	264

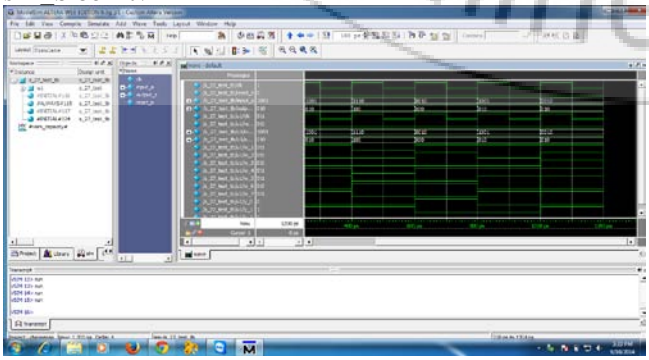
1) If the fault coverage does not exceed that of [19] for any solution, we report on the following solutions.
a) The solution with the highest fault coverage and the Lowest number of seeds.
b) The solution with the highest fault coverage and the Lowest number of applied tests.

2) If the fault coverage of at least one solution exceeds that Of [19], we report on the following solutions.
a) The solution with the highest fault coverage.
b) The solution with fault coverage higher than that of [19] and the lowest number of seeds.
c) The solution with fault coverage higher than that of [19] and the lowest number of applied tests.

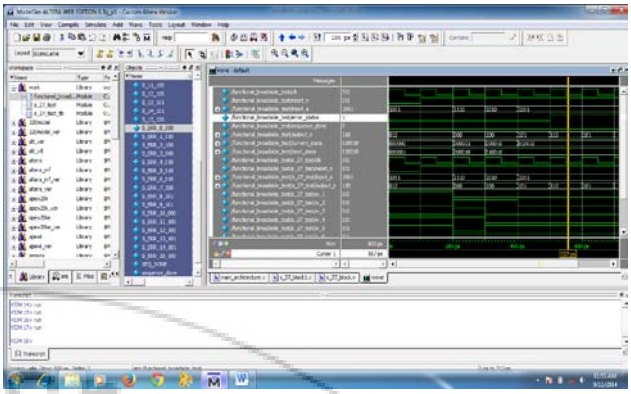
One of these solutions is expected to be the most appropriate for the circuit. In addition to these solutions there are others with intermediate numbers of seeds and applied tests, which are not reported

4. Experimental Results

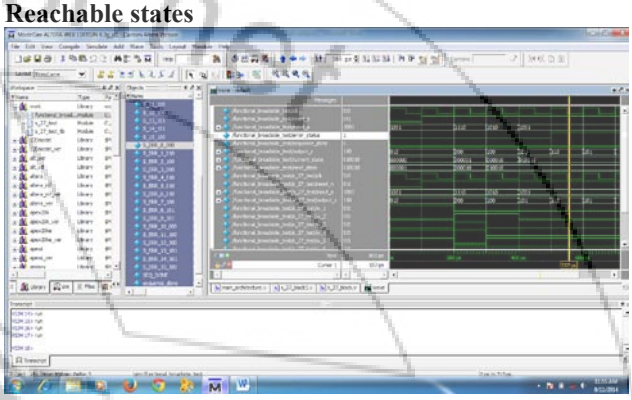
s27_block1.v



main_architecture.v(Un Reachable states)



Reachable states



5. Conclusion

This paper described an on-chip test generation method for Functional broadside tests. The hardware was based on the application of primary input sequences starting from a known reachable state, thus using the circuit to produce additional reachable states. Random primary input sequences were modified to avoid repeated synchronization and thus yield varied sets of reachable states. Two-pattern tests were obtained by using pairs of consecutive time units of the primary input sequences. The hardware structure was simple and fixed, and it was tail oared to a given circuit only through the following parameters: 1) the length of the LFSR used for producing a random primary input sequence; 2) the length of the primary input sequence; 3) the specific gates used for modifying the random primary input sequence; 4) the specific gate used for selecting applied tests; and 5) the seeds for the LFSR. The on-chip generation of functional broadside tests achieved high transition fault coverage for testable circuit When you submit your paper print it in two-column format, including figures and tables [1]. In addition, designate one author as the “corresponding author”. This is the author to whom proofs of the paper will be sent. Proofs are sent to the corresponding author only [2].

References

[1] A. Bonnaccorsi, “On the Relationship between Firm Size and Export Intensity,” Journal of International Business Studies, XXIII (4), pp. 605-635, 1992. (journal style)
[2] R. Caves, Multinational Enterprise and Economic Analysis, Cambridge University Press, Cambridge, 1982. (book style)

[3] M. Cl e rc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization," In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), pp. 1951-1957, 1999. (conference style)
[4] H.H. Crokell, "Specialization and International Competitiveness," in Managing the Multinational Subsidiary, H. Etemad and L. S, Sulude (eds.), Croom-Helm, London, 1986. (book chapter style)
[5] K. Deb, S. Agrawal, A. Pratab, T. Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithms for Multi objective Optimization: NSGA II," KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000. (technical report style)
[6] J. Geralds, "Sega Ends Production of Dreamcast," vnunet.com, para. 2, Jan. 31, 2001. [Online]. Available: http://nl1.vnunet.com/news/1116995. [Accessed: Sept. 12, 2004]. (General Internet site)

Author Profile

P. Durga Venkata Prasad pursuing the M.Tech degree in VLSI SYSTEM DESIGN from Nimra college of Engineering &Technology, Received B.Tech degree in ECE from Akula Sri Ramulu College of engineering and Technology.

K. Tirumala Rao Received the M.Tech degree in VLSI SYSTEM DESIGN from Avanthi Institute of Engineering and Technology,Narsipatnam, B.Tech degree in ECE at Nimra college of engineering and Technology. He has total Teaching Experience (UG and PG) Of 3 years. He has guided and co-guided 3P.G and U.G students.