

Test Compaction of Logic Blocks by Sharing of Transparent Scan Sequences Using Walsh Code Algorithm

A.S.Janapriya¹ S.Sivaganesan²

¹ME-VLSI Design ²Assistant Professor

^{1,2}Department of Electronics & Communication Engineering

^{1,2}Kalaignar Karunanidhi Institute of Technology, Affiliated To Anna University, Coimbatore, Tamil Nadu, India

Abstract— An arbitrary design implemented into a field-programmable gate array (FPGA). FPGA contains many logical blocks. An approach provides transparent scan to share tests among different logic blocks whose primary inputs and outputs are included in scan chains even if the blocks have different numbers of state variables. The transparent-scan sequences based on tests for one logic block could detect faults in other logic blocks, with different numbers of state variable. It uses n number of test configuration instead of 2n number of test configuration by walsh code algorithm. Transparent scan enhances the ability to produce a compact test set for a group of logic blocks. The procedure obtains a set of transparent-scan sequences for a group of logic blocks from compacted test sets for the logic blocks in the group. From this set, it selects a subset that detects all the target faults, which are detected by the complete set.

Key words: Full-scan circuits, test compaction, test generation, transparent scan, Field-programmable gate array (FPGA), testing

I. INTRODUCTION

SRAM-BASED field programmable gate arrays (FPGAs) are 2-D arrays of logic blocks and programmable switch matrices, surrounded by programmable input/output (I/O) blocks on the periphery. Transparent scan test to test the logic blocks in the FPGA. In the logic block, scan-select and scan-chain inputs of a scan circuit are considered as inputs of the sequential circuit in the same way as the primary inputs, and the scan-chain outputs are considered as outputs in the same way as the primary outputs. Faults are allowed to be detected during all the clock cycles of a transparent-scan sequence. In general, fault coverage is computed by sequential fault simulation of the transparent-scan sequence.

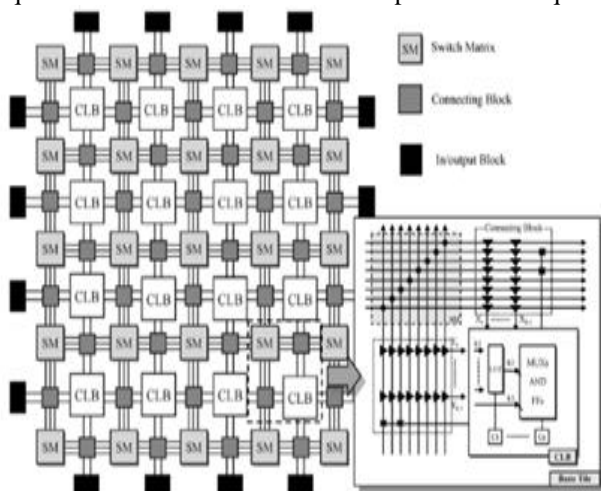


Fig. 1: SRAM-based FPGA architecture

Transparent-scan sequences are allowed to assign arbitrary values to the scan-select input. This resulted in arbitrary sequences of scan and functional clock cycles. An initial transparent-scan sequence can be obtained from a conventional scan-based test set, and it can follow its application precisely cycle by cycle.

In this case, the scan-select and scan-chain input sequences are such that the conventional test set is applied to the circuit by applying the transparent scan sequence. When a logic block is embedded in a design, access to its primary inputs and primary outputs, as well as its state variables, for the purpose of test application may be available only serially through scan chains. Fig.1 illustrates such a logic block B_i with a single scan chain. The scan chain is marked with a dashed line.

The scan-select input of B_i is denoted by $SCSEL_i$, its scan-chain input by $SCINP_i$, and its scan-chain output by $SCOUT_i$. Under the model of Fig. 1, a transparent-scan sequence for B_i specifies values only for $SCSEL_i$ and $SCINP_i$. The output sequence specifies only the corresponding values of $SCOUT_i$. Thus, the input values are brought to all the inputs of the combinational logic serially, and the output values of all the outputs of the combinational logic are observed serially.

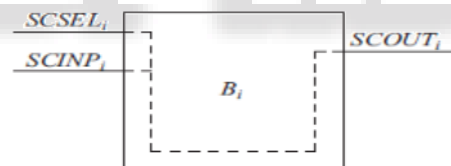


Fig 2: Logic block

Every logic block B_i follows the model of Fig. 1 with serial access to all the inputs and all the outputs of its combinational logic. A transparent-scan sequence for B_i is obtained from every test in a conventional single-cycle scan-based test set of B_i . The transparent-scan sequence follows the application of the test precisely, and it is not modified. The scan-select and scan-chain input sequences of the transparent-scan sequences based on it are not modified, the scan-chain input sequences satisfy the same constraints as the conventional tests from which they were obtained. The original configuration of the used logic blocks is preserved, whereas the configuration of the global interconnects and unused logic blocks are changed to test all used logic blocks.

II. TRANSPARENT SCAN

A transparent-scan sequence T_i, j for a logic block B_i that follows the model of Fig. 1 specifies two values at every clock cycle, the value of the scan-select input $SCSEL_i$, and the value of the scan-chain input $SCINP_i$. The test vector at clock cycle u of T_i, j is denoted by $T_i, j(u)$. The value of the

scan-select input under $T_{i,j}(u)$ is denoted by $T_{i,j}(u, 0)$, and the value of the scan-chain input is denoted by $T_{i,j}(u, 1)$. For a functional clock cycle, $T_{i,j}(u, 0) = 0$. For a scan clock cycle, $T_{i,j}(u, 0) = 1$.

For illustration, the next example is based on a logic block B_i with four flip-flops in its scan chain. Let $S_i = \{0011, 0101, 1001\}$ be a conventional single-cycle scan-based test set for B_i .

u	$T_{i,0}(u)$	$T_{i,1}(u)$	$T_{i,2}(u)$
0	11	11	11
1	11	10	10
2	10	11	10
3	10	10	11
4	0x	0x	0x
5	1x	1x	1x
6	1x	1x	1x
7	1x	1x	1x
8	1x	1x	1x

Table 1: Transparent scan sequences

The transparent-scan sequences shown in Table I apply these tests to the circuit. Considering $s_i, 0 = 0011$, clock cycles 0 to 3 of $T_{i,0}$ are scan clock cycles, and they have $T_{i,0}(u, 0) = 1$ for $0 \leq u \leq 3$. These clock cycles are used for loading the test 0011 into the scan chain. The values of the scan-chain input, $T_{i,0}(u, 1)$ for $0 \leq u \leq 3$, correspond to this test assuming that scan chains are shifted to the right. Clock cycle 4 is a functional clock cycle, with $T_{i,0}(4, 0) = 0$. This clock cycle is used for capturing the circuit response to 0011 in the scan chain. The scan-chain input value $T_{i,0}(u, 1)$ can be determined arbitrarily, and it is marked with an "x" in Table I. Clock cycles 5 to 8 are scan clock cycles with $T_{i,0}(u, 0) = 1$ for $5 \leq u \leq 8$. They allow the response of the circuit to 0011 to be scanned out and observed. The values of the scan-chain input $T_{i,0}(u, 1)$, for $5 \leq u \leq 8$, can be determined arbitrarily. They can be used for overlapping the test with the next test. For example, a two-pattern broadside test for a logic block with k_i state variables would have two functional clock cycles between two scan subsequences of length k_i .

For the transparent-scan sequences considered in this paper, it is also possible to use combinational fault simulation instead of sequential fault simulation. This can be achieved by applying the following process.

- 1) The present state for the functional clock cycle can be computed without logic or fault simulation based on the values of the scan-chain input during the scan clock cycles that define the scan-in operation at the beginning of the test.
- 2) Combinational fault simulation is required for the functional clock cycle.
- 3) Based on the fault effects that are propagated to the flipflops during the functional clock cycle, and the number of scan clock cycles for the scan-out operation at the end of the test, it is possible to compute which fault effects will reach an output.

III. TEST CONFIGURATION GENERATION PROCEDURE

Test generation under the proposed approach, which eliminates the distinction between scan operations and

application of primary input vectors, can be done as follows. The circuit for which test generation is carried out is Cscan. This circuit has two extra primary inputs compared to the original circuit C: the scan-in input scan_inp, and the scan-select input scan_sel. It also has an extra primary output, the scan output scan_out. The procedure will produce a test sequence where scan_sel and scan_inp are used as conventional primary inputs, and fault effects may be observed on scan_out. An example of such a test sequence is shown in Table 2.

	a_1	a_2	a_3	a_4	scan_sel	scan_inp
0	0	0	1	0	0	0
1	1	1	0	1	0	0
2	0	0	1	0	0	0
3	0	0	0	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	0	0
7	0	0	0	0	1	0
8	0	0	0	1	0	0
9	1	0	0	0	0	0
10	0	0	0	1	0	0
11	0	0	0	0	0	1
12	0	0	0	1	0	0
13	0	0	0	0	1	0
14	0	0	0	0	1	1
15	0	0	0	1	0	0
16	1	0	0	0	1	0
17	0	0	0	1	0	0
18	0	0	0	0	1	1
19	0	0	0	0	1	0
20	0	0	0	0	0	0
21	0	1	0	0	0	0
22	0	0	1	0	0	0
23	1	0	0	1	0	0
24	0	0	0	0	0	0

Table 2: Test sequence for s 27scan

This sequence was generated for s 27scan, which is the scan version of ISCAS-89 benchmark circuit s 27. The circuit has four primary inputs labeled a_1, a_2, a_3, a_4 . It has three state variables. It is interesting to note that scan is applied for a single time unit at time unit 5, 7 and 16. In addition, it is applied for two consecutive time units at time units 13, 14 and 18, 19. Thus, all the scan operations are limited scan operations with one and two shifts of the scan chain, and there is never a complete scan operation that takes three shifts of the scan chain.

A. Walsh Code Algorithm:

- 1) Activating all possible faults (stuck-at, open, and pairwise bridging faults) for M nets (wires) can be performed using only $\lceil \log_2(M+2) \rceil$ test vectors.
- 2) These vectors are columns of binary representations of numbers 1 to M using $\lceil \log_2(M+2) \rceil$ bits and called Walsh codes. This concept used for interconnect testing.

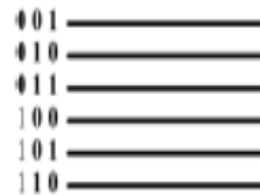


Fig. 3: Logarithmic test set to activate all faults for six wires.

In order to detect all faults in the fault list, faults must be sensitized using a set of single-term functions and test vectors shown in fig.2. These single-term functions are implemented in all LUTs used in the user design. The single-term functions implemented in the user LUTs correspond to a test configuration which detect the interconnect faults sensitized in that test configuration. The objective is to come up with a minimum number of test configurations such that all faults in the fault list are sensitized and, hence, detected in at least one test configuration. Testing for bridging faults has always been a challenging issue, particularly for ASICs. This is mainly due to the fact that finding an appropriate fault list for bridging faults is not as straightforward as that for stuck-at faults. The number of all possible single stuck-at faults in a circuit is linear with the size of the circuit whereas the number of all pairwise bridging faults is quadratic with the size of the circuit. Activating all possible faults (stuck-at, open, and pairwise bridging faults) for M nets performed using only $\lceil \log_2(M+2) \rceil$ test vectors. These vectors are columns of binary representations of numbers 1 to using bits and called Walsh codes.

IV. THE STATIC COMPACTION

The method is to effect static compaction by a few repetitions of the selection of essential matrices and discarding of identified redundant matrices.

A. Lemma 1:

With reference to a given set of faults in a circuit C , any test set T without essential matrices has at least one test matrix t that is redundant with respect to the set $T = T \setminus \{ t \}$.

B. Proof:

Suppose that at a cycle of the repetition, the test matrices $T_1, T_2, \dots, T_i, \dots, T_N$ are remaining. Assume there are no essential matrices in the remaining subset of test matrices. Thus, any fault in the cover of the subset of the remaining test matrices that is detectable by the matrix r_j is also detectable by another matrix T_k (i.e. $k \neq j$) in the same subset. Thus, at least, T_j is redundant.

- 1) Test matrices are applied one-by-one to simulate CUT.
- 2) At the application of each test matrix, deduction rules are used to trace detectable faults (under the single fault assumption) from the primary outputs backward to the primary inputs (see appendix).

V. TEST COMPACTION PROCEDURE

This section describes a test compaction procedure that accepts a set of conventional scan-based test sets S_0, S_1, \dots, S_{n-1} for logic blocks B_0, B_1, \dots, B_{n-1} . The procedure translates the tests into a set T of transparent-scan sequences. It then selects a subset of these sequences that is sufficient for detecting all the target faults that are detected by S_0, S_1, \dots, S_{n-1} . As before, the number of state variables of B_i is denoted by k_i , for $0 \leq i < n$. The set of faults that S_i detects in B_i is denoted by F_i . The set of target faults is $F = \cup_{i=0}^{n-1} F_i$. In addition, $S_i = \{si,0, si,1, \dots, si,mi-1\}$. For $0 \leq i < n$ and for $0 \leq j < mi$, the procedure translates si, j into a transparent-scan sequence Ti, j as

described in the previous section. It then adds Ti, j to T . It is possible to use a set covering procedure in order to select a subset of T that detects all the faults in F .

However, a set covering procedure requires information about all the sequences from T that detect every fault in F . In the context of transparent scan, the two steps proceed as follows.

Step 1 selects a subset $Tsel1 \subseteq T$ by identifying faults from F that are detected by unique sequences in T . If a fault $f \in F$ has only one transparent-scan sequence $Ti, j \in T$ that detects it, Ti, j must be included in the selected subset of transparent scan sequences. In this case, Ti, j is included in $Tsel1$. Step 2 selects additional transparent-scan sequences as necessary to produce a subset $Tsel2$ that detects all the faults in F . The details of the two steps are described next. Step 1 performs two-detection fault simulation of the faults in F under the transparent-scan sequences in T . The number of detections of a fault $f \in F$ is equal to the number of sequences from T that detect the fault. The two-detection fault simulation procedure drops a fault from further simulation after it finds two transparent-scan sequences in T that detect the fault. It stores the number of times a fault $f \in F$ is detected during this process in a variable denoted by $ndet(f)$. It stores the index of the first transparent-scan sequence that detects a fault $f \in F$ in a variable denoted by $first(f)$. If, at the end of this process, a fault $f \in F$ has $ndet(f) = 1$, the sequence with index $first(f)$ must be selected. This sequence is included in $Tsel1$.

The expectation is that the sequences in $Tsel1$ will detect most of the faults in F . $Tsel1$ is guaranteed to detect a fault f with $ndet(f) = 1$. For a fault f with $ndet(f) = 2$, there are two or more options for transparent-scan sequences in T that detect it. Such a fault is likely to be detected by $Tsel1$. With a small number of faults that are not detected by $Tsel1$, Step 2 uses fault simulation with fault dropping to select additional sequences so as to detect all the faults in F .

Step 2 starts by assigning $Tsel2 = Tsel1$. It performs fault simulation with fault dropping of F under the transparent-scan sequences in $Tsel2$ in order to remove from consideration faults that are already detected. It then performs fault simulation with fault dropping of F under $T - Tsel2$. If a fault $f \in F$ is detected by a transparent-scan sequence $Ti, j \in T - Tsel2$, the procedure adds Ti, j to $Tsel2$. For ease of implementation, fault simulation during steps 1 and 2 is performed for one logic block at a time. Thus, the procedure does not need all the logic blocks to be stored simultaneously. In addition, when a logic block B_i is simulated in Step 1 or 2, the transparent-scan sequences that are based on its tests are simulated first, followed by the transparent-scan sequences that are based on the tests of the other blocks. This ensures that faults are detected and dropped from consideration as early as possible.

For illustration, the next example considers a group of logic blocks where B_0 is ISCAS-89 benchmark s27, B_1 is ITC-99 benchmark s208. The numbers of state variables for these circuits are $k_0 = 8$, and $k_1 = 7$. The numbers of conventional scan based tests for these circuits are $m_0 = 5$, and $m_1 = 12$. This results in 36 transparent-scan sequences, which are included in T . The numbers of single stuck-at faults are 32 and 70 respectively. The results of two-detection fault simulation for seven faults of each circuit are shown in Table II. For a fault $fi,k \in Fi$.

VI. SIMULATION RESULTS

A. Walsh Code Method:



Fig. 4: Walsh Code method

It shows the combined benchmark circuit of s27-s208 using walsh code algorithm.

B. Comparison Table:

Circuit	Existing method			Proposed
	S27	S208	S27-S208 Combined	S27-S208 Walsh
Test Vector	100	100	100	80
Fault coverage	50	69	50+69	69+69

Table 3: Comparison of test vector and fault coverage using walsh code algorithm

Table 3 shows comparison of test vector and fault coverage using walsh code algorithm. The proposed technique uses walsh code algorithm method with reduced number of test vectors and also it provide high fault coverage compare to the existing method.

VII. CONCLUSION

Walsh code algorithm used in the test compaction, it minimizes the total number of test vectors used in the circuit. This project describes a test compaction procedure under transparent scan for groups of logic blocks whose primary inputs and outputs are scanned. Using walsh code algorithm it reduces 2^n number of test vector sequence to n number of test vector. Experimental results showed that transparent-scan sequences based on tests for one logic block could detect faults in other logic blocks, with different numbers of state variables. This allowed a reduced number of transparent-scan sequences to be used for the group. Transparent-scan sequences of logic blocks with higher numbers of state variables typically detected faults of logic block with smaller numbers of state variables. This was the main contributor to the reduction in the number of

transparent scan sequences for the group using walsh code algorithm.

REFERENCES

- [1] Abramovici.M, Breuer, and Friedman.D (1995) Digital Systems Testing and Testable Design. Piscataway, NJ, USA: IEEE Press, Vol. 22, No. 10, pp. 1443–1449.
- [2] Barnhart.C, Brunkhorst.V, Distler.F, Farnsworth, and Koenemann (2001) ‘OPMISR: The foundation for compressed ATPG vectors,’ in Proc. Int. Test Conf.,Vol. 20, No. 10, pp. 748–757.
- [3] Boateng.B, Konishi.H, and Nakata.T.M. (2001) ‘A method of static compaction of test stimuli,’ in Proc. Asian Test Symp.,Vol. 18, No. 10, pp. 137–142.
- [4] Chang and Lin (1992) ‘Test set compaction for combinational circuits,’ in Proc. Asian Test Symp.,Vol. 21, No. 8, pp. 20–25.
- [5] Dimopoulos.M and Linardis.P (2003) ‘Accelerating the compaction of test sequences in sequential circuits through problem size reduction,’IEEETrans. Comput.-Aided Design, Vol. 22, No. 10, pp. 1443–1449.
- [6] Dimopoulos and Linardis (2004) ‘Efficient static compaction of test sequence sets through the application of set covering techniques,’ in Proc. Design, Autom. Test Eur. Csonf.,Vol. 17, No. 5, pp. 194–199.
- [7] El-Maleh and Osais (2003) ‘Test vector decomposition-based static compaction algorithms for combinational circuits,’ACM Trans. DesignAutom. Electron. Syst., Vol. 8, No. 4, pp. 430–459.
- [8] Flores P.F, Neto H.C, and Marques-Silva (1999) ‘On applying set covering models to test set compaction,’ in Proc. Great Lakes Symp.VLSI,Vol. 13, No. 4, pp. 8–11.
- [9] Goel.P and Rosales.B.C (1979) ‘Test generation and dynamic compaction of tests,’ in Proc. Test Conf., Vol. 15, No. 7, pp. 189–192.
- [10]Hamzaoglu and Patel.J.H (1998) ‘Test set compaction algorithms for combinational circuits,’ in Proc. Int. Conf. Comput.-Aided Design,Vol. 11, No. 8, pp. 283–289.
- [11]Hochbaum.D.S (1996) ‘An optimal test compression procedure for combinational circuits,’IEEE Trans. Comput.-Aided Design, Vol. 15, No. 10, pp. 1294–1299.
- [12]Kajihara.S, Pomeranz.I, Kinoshita.K, and Reddy.S.M (1995) ‘Cost-effective generation of minimal test sets for stuck-at faults in combinational logic circuits,’IEEE Trans. Comput.-Aided Design, Vol. 14, No. 12, pp. 1496–1504.
- [13]Koenemann, Barnhart.C, Keller, Snethen, Farnsworth, and Wheeler (2001) ‘A SmartBIST variant guaranteed encoding,’ in Proc. AsianTest Symp.,Vol. 21, No. 3, pp. 325–330.
- [14]Lee.K, Chen.J, and Huang.C (1998) ‘Using a single input to support multiple scan chains,’ in Proc. Int. Conf. Comput.-Aided Design, Vol. 12, No. 2, pp. 74–78.