Efficient Embedding of Deterministic Test Data

Mudassar Majeed[1], Daniel Ahlström[1], Urban Ingelsson[1], Gunnar Carlsson[2] and Erik Larsson[1]

[1]Department of Computer and Information Science, Linköping University, Sweden
[1]Email: {mudma,x09danah,urbin,erila}@ida.liu.se

[2]Ericsson AB BU Networks SE-164 80 Stockholm, Sweden
[2]Email: gunnar.carlsson@ericsson.com

***Abstract*—Systems with many integrated circuits (ICs), often of the same type, are increasingly common to meet the constant performance demand. However, systems in recent semiconductor technologies require not only manufacturing test, but also in-field test. Preferably, the same test set is utilized both at manufacturing test and in-field test. While deterministic test patterns provide high fault coverage, storing complete test vectors leads to huge memory requirements and inflexibility in applying tests. In an IEEE 1149.1 (Boundary scan) environment, this paper presents an approach to efficiently embed deterministic test patterns in the system by taking structural information of the system into account. Instead of storing complete test vectors, the approach stores only commands and component-specific test sets per each unique component. Given a command, test vectors are created by a test controller during test application. The approach is validated on hardware and experiments on ITC'02 benchmarks and industrial circuits show that the memory requirement for storing the test data for a system is highly related to the number of unique components.**

***Keywords*-In-field test, Embedded boundary scan test, System test, Multiple identical cores, Test controller.**

I. Introduction

Electronic systems, printed circuit boards (PCBs), with an increasing number of integrated circuits (ICs) are increasingly common [1]. PCBs include an increasing number of integrated circuits where a high number of ICs are of the same type. For example, Ericsson have telecommunication systems with 36 ICs on each PCB. Many of the 36 ICs are identical. The ICs are distributed as follows: 4 ICs of type A, 8 ICs of type B, 8 ICs of type C and 16 ICs of type D. Manufacturing of ICs and PCBs is unfortunately far from perfect and all IC as well as the complete PCBs must be tested carefully. Until recently, manufacturing test only was sufficient to ensure fault-free electronic systems. ICs are tested using automatic test equipments and IEEE 1149.1 (Boundary scan) was developed to enable testing of PCBs.

However, the supply voltage and the transistor size have been scaled down in recent designs to achieve higher performance and low power dissipation. This has led to a higher number of electronic systems failing in the field. Therefore the maintenance cost has increased in terms of monitoring, diagnosing and repairing systems. To reduce the cost, there is a trend towards automated in-field testing. An obstacle is the high test data volumes needed to efficiently test electronic systems.

Test data compression has been proposed to reduce the ATE memory requirements at IC test [2] [3] [4]. A disadvantage of test data compression for in-field test is the need of an ATE and the low diagnostic capability. To make in-field testing possible, Built-in Self Test (BiST) solutions [5] including use of JTAG are used where either pseudo-random test patterns are generated within the system or pre-generated deterministic test patterns (with high fault coverage) are stored in system memory. In terms of test execution time and fault coverage, deterministic test patterns tend to be more effective than pseudo-random test patterns, as it was noted in [6] that the fault coverage that is achieved with pseudo-random test patterns on large industrial designs can be as low as 65%. Further, pseudo-random patterns have the disadvantage of low diagnostic resolution.

In the context of testing ICs with multiple modules (blocks of logic), relevant previous studies were conducted in [7] and [8]. In [7], test vectors are broadcasted to multiple cores of the same type and the test responses are compared with each other that makes it unable to test a single component. Furthermore, test data is stored in the ATE memory and the approach in [7] is meant for manufacturing testing. In [8], one of multiple identical processor cores is selected for test and test vectors are applied and evaluated while the other cores are in operation mode. The test vectors for considered type of the processor cores are stored once on a hard drive. The approach is not applicable to systems with JTAG.

This paper considers a system consisting of ICs that are mounted on a PCB and connected with a JTAG IEEE 1149.1 (Boundary scan) interface for manufacturing test and in-field test. However, the test data volume (the amount of bits that require system memory for storage) can be significant, in the range of Gigabytes [6] (not compressed). Thus, for systems where storing deterministic test patterns is required for in-field test, the test data volume requires costly memory and the research presented in this paper aims to reduce this cost. The novelty of the approach presented in this paper is that it does not assume all the components (ICs) of the same type and investigates how the idea of reducing memory requirements by utilizing structural information can be applied in the context of an embedded test controller including on-the-fly manipulation of test data for the system in field. The key contribution of this work is an embedded test controller called Concatenator that can accept the external commands from a test manager to automate the test application and test evaluation process. The Concatenator incorporates the structural information of the system and provides the flexibility to test any combination

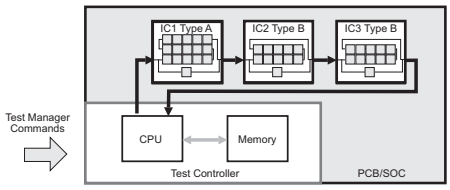
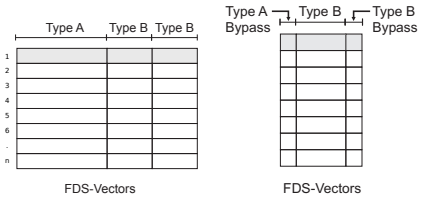


Fig. 1. An embedded system with a test controller



(a) Test all components (b) Test IC2

Fig. 2. FDS-vectors stored in memory

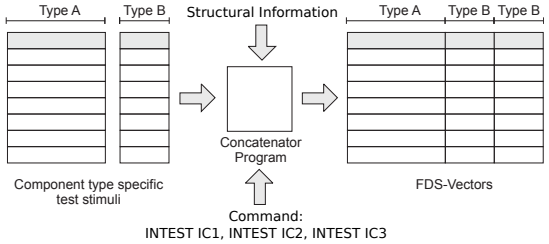


Fig. 3. Test data concatenation for testing all the components.

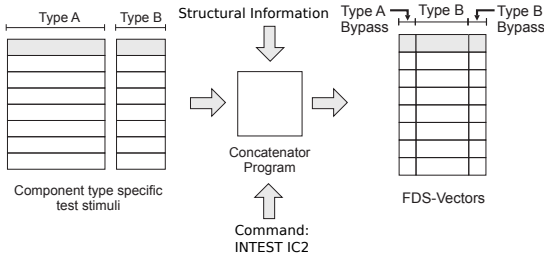


Fig. 4. Test data concatenation for testing IC2 only.

of components within a system (PCB) and reduces the memory requirements to store the test data volume. We have used the existing on board infrastructure (JTAG) to transport the test data to the components.

II. Basic Idea of Concatenation

This paper considers PCBs that may contain several mounted ICs. An example of such a system is shown in Figure 1. It contains three components named IC1, IC2 and IC3. Here, IC1 is of type A. IC2 and IC3 are both of type B. A serial interface following the IEEE 1149.1 Standard is employed to transport test data to and from the components, as is indicated by the black arrows in Figure 1. The IEEE 1149.1 Standard defines several instructions e.g., INTEST (to test the IC) and BYPASS (to bypass the IC).

Testing the system in Figure 1 is done by performing four steps. 1) Scan in a 35 bits fully-defined scan-chain test vector (FDS-vector hereinafter) of stimulus 2) Test stimuli application. 3) Scan out test response (i.e. an FDS-vector of 35 bits). 4) Compare test responses to expected responses. Several iterations of this procedure are required to adequately test the components.

The FDS-vector can be very long, tens of thousands of bits, when testing a system with several large components. Furthermore, it can happen that tens of thousands of FDS-vectors are required. To test all the components in the system, the set of stimuli FDS-vectors that is stored in memory is shown in Figure 2(a). Each box shows a test stimulus for the component of the type shown at the top of the column of that box (Figure 2(a)). Each row shows a complete FDS-vector for the components in the system (Figure 1) that are selected for the test. To test another combination of the components in the system (for example testing only IC2 of type B) another set of FDS-vectors is required to be stored in the memory as shown in Figure 2(b).

As seen from Figure 2(a) and Figure 2(b), to achieve the flexibility to test any combination of components in the system, it is required to store different tests. Notice that the test stimuli for the component of type B are stored twice in each FDS-vector (shown in Figure 2(a)). Furthermore, the same test data for type B component is stored in another test as shown in Figure 2(b). Hence the flexibility in applying the tests is achieved at the cost of memory.

The basic idea behind the concatenation approach in this work is to store only the component-type specific test patterns in the memory and run a dedicated program (Concatenator test controller hereinafter) in the on-system CPU that receives external test command from the test manager. Depending upon the external test command, the Concatenator reads the component specific test patterns and manipulates those test patterns to generate FDS-vectors on-the-fly. By manipulation of test patterns, we mean that the test stimuli and test response vectors of the stored component-type specific test patters must be concatenated into FDS-vectors according to the order of the components on the scan-path. For this it is required that structural information about the system is available. The concatenation of test stimuli is illustrated in Figure 3 and Figure 4. To test all the three components, the external command is INTEST IC1, INTEST IC2, INTEST IC3. Similarly, to test IC2 alone and bypass the other two components the external command is INTEST IC2. Test responses are concatenated in the same way. Notice that the same test data (shown in Figure 2(a) and Figure 2(b)) can be generated on the fly by using the component specific test patterns stored in the memory.

The structural information about the system (system description) is read by the Concatenator program and is shown in Figure 5. It provides the order of the components, their types and some information related to their JTAG circuitry.

The test data for each component that the Concatenator program reads is structured as in the example of Figure 6.

```
1  IC1 - IC2 - IC3
2  TypeA = IC1
3  TypeB = IC2 = IC3
4  TypeA DRLENGTH 15
5  TypeA IRLENGTH 10
6  TypeA BYPASS 1111111111
7  TypeA INTEST 0000011111
8  TypeB DRLENGTH 10
9  TypeB IRLENGTH 8
10 TypeB BYPASS 11111111
11 TypeB INTEST 00001111
```

Fig. 5. System description data

```
TypeA INTEST1 4
100100100100100 010101011010101
011011011011011 101010100101010
101101110111101 0000000111111111
010010001000010 111111100000000
TypeB INTEST1 2
0101010101 0110110110
1010101010 1001001001
```

Fig. 6. Component-type specific test data

The test data consist of component-type specific test sets (INTEST1 for type A component and INTEST1 for type B components), which are named with a component-type, a test id and the number of test patterns in the test set (Figure 6). In Figure 6, each test pattern is on a separate line and consists of a test stimulus and expected response.

The Concatenator test controller is able to manipulate the test data on-the-fly so that only component-type specific tests require memory and it is able to accept commands. It automatically handles the application of tests with different number of test patterns and reconfigures the JTAG scan-path accordingly. The following section will describe experiments to show the capabilities of the Concatenator test controller.

III. Experiments and Results

We have performed experiments to determine the amount of memory that can be saved using the concatenation approach presented in the paper. Experiments are performed with ITC'02 SOC benchmarks [9] and industrial circuits from [10] respectively. The components of the ITC'02 benchmarks and the circuits in [10] are treated as ICs and specified in terms of number of test patterns and scan-chain lengths as shown in Table I and Table II respectively. The left most column (refers to both Table I and Table II) shows the name given to the circuit. In the second column, the scan-chain length for each circuit is shown and this is followed by a column showing the number of test patterns for the each circuit. The test data volume required to test each circuit is shown in the right most column. The sizes of the circuits in Table II are within the range of approximately 50K gates to 1.4M gates. In our experiments with the circuits from [10], we have taken each circuit (Table II) as a component and compiled system designs using these components. In each of the system designs, one of the eight components is multiplied several times as shown in Table III.

Two simplifications are done for the purpose of the experiments. First simplification is that the Concatenator test controller (referred as Concatenator program hereafter) is run on a PC instead of an embedded processor on a PCB. We used an FPGA as a system under test and connected it with the PC. This way we have verified that the output (set of FDS-vectors) produced by the Concatenator test controller can be applied on a real circuit.

TABLE I
ITC'02 SOC benchmarks with Test Data Volume

Circuit	Length of Test Patterns (bits)	Number of Test Patterns	Test Data Volume (KBs)
f2126-1	8875	334	362
f2126-2	5328	422	274
f2126-3 *	502	103	6
f2126-4 *	502	103	6
Total	-	962	648
p93791-1	6942	409	347
p93791-2	153	11	0.2
p93791-3	24530	218	653
p93791-4	790	187	18
p93791-5	4562	391	218
p93791-6 *	9669	194	229
p93791-7 *	9669	194	229
p93791-8	6602	216	174
p93791-9	5180	210	133
p93791-10	7598	416	386
p93791-11	7772	234	222
p93791-12	3063	916	343
p93791-13	6684	172	140
Total	-	3768	3092
q12710-1	3517	852	365
q12710-2	11971	1314	1920
q12710-3 *	5335	1223	797
q12710-4 *	5335	1223	797
Total	-	4612	3879

TABLE II
Industrial Circuits with Test Data Volumes

Circuit	Length of Test Patterns (bits)	Number of Test Patterns	Test Data Volume (MBs)
ckt-1	12256	3768	5.51
ckt-2	22216	2636	6.98
ckt-3	9628	4927	5.65
ckt-4	43414	1528	7.91
ckt-5	26970	4899	15.75
ckt-6	80000	2859	27.27
ckt-7	20000	18027	42.98
ckt-8	110000	18142	237.9

We used Equation 1 to calculate the percentage reduction in memory requirements for each scenario. In Formula 1, $Size_{DF}$, $Size_{TDF}$, $Size_{ODF}$ and $Size_{Program}$ denote the size of description file, test data file, output data file and the Concatenator program respectively and RMR denotes reduction in memory requirements.

$$\% \; RMR = 100 \cdot \left(1 - \frac{Size_{DF} + Size_{TDF} + Size_{Program}}{Size_{ODF}}\right) \quad (1)$$

The experiments conducted for the designs using the components from Table II produced the results shown in Table IV. The columns with the heading Reduction in MR in Table IV show the percentage reduction in memory requirements (MR) for the designs. In the column marked D1, ckt-1 is included X

TABLE III
EXAMPLE DESIGNS

Design	Circuits used in the Design
D1x2	ckt-1,ckt-1,ckt-2,ckt-3,ckt-4,ckt-5,ckt-6,ckt-7,ckt-8
D2x2	ckt-1,ckt-2,ckt-2,ckt-3,ckt-4,ckt-5,ckt-6,ckt-7,ckt-8
D8x2	ckt-1,ckt-2,ckt-3,ckt-4,ckt-5,ckt-6,ckt-7,ckt-8,ckt-8

TABLE IV
DESIGNS RESULTS

	% Reduction in MR								
X	D1	D2	D3	D4	D5	D6	D7	D8	Avg
1	0	0	0	0	0	0	0	0	0
2	2	2	2	2	4	7	11	41	9
3	3	4	3	4	8	13	20	58	14
4	4	6	5	6	12	19	27	68	18
5	6	7	6	8	15	24	33	74	21
6	7	9	7	10	18	28	38	78	24
7	9	11	9	12	21	32	42	81	27
8	10	12	10	14	24	35	46	83	29
19	22	26	23	29	45	58	69	92	45
20	23	27	24	30	46	60	70	93	46

times (X is shown in the left most column) whereas the other seven circuits are used only once. Similarly, in the column D2, ckt-2 is multiplied X times and other circuits are used only once and so on.

In Table IV, it can be observed that the test data volume reduction depends on the circuit that is included in multiple instances and the number of multiplications. The reason why the reduction in memory requirements depend on the multiplied circuit (columns D1 to D8) is that the circuits themselves have different test data volume (Table II).

The result for D8 in the second row (Table IV) shows that, if a component (with high test data volume) in a system is instantiated only two times, the percentage reduction in memory requirements can be achieved up to 41%. For X = 6, the average percentage reduction in memory requirements is 24%. Further results show that if the multiplication of cores within the systems is increased in the future, then the approach presented in this paper will be even more beneficial in terms of reduction in memory requirements.

The results for the ITC'02 benchmarks are shown in Table V. The first column shows the name of the SOC designs. The second column shows the memory requirements for storing FDS-vectors as it would be done without the proposed concatenation approach. The third column shows the total memory requirements with the proposed concatenation approach. The last column shows the percentage reduction in memory requirements. For SOC p93791 the total test data volume needed to test were 3092 KBs and there were two components of the same type, each used 229 KBs. This lead to a reduction of 7.5%. For SOC q12710 the total test data volume was 3879 KBs and there were two components of the same type with 797 KBs of test data volume each. Total reduction of memory requirements was 26.4%. For SOC f2126 the total test data volume needed was 648 KBs, two components of the same type with 6 KBs of test data volume each. total reduction of memory requirement 1%.

TABLE V
ITC'02 BENCHMARK RESULTS

SOC design	MR* (KBs)	MR* using proposed approach (KBs)	% Reduction in MR*
f2126	648	641	1.0%
p93791	3092	2860	7.5%
q12710	3879	2855	26.4%

* Memory Requirements

IV. CONCLUSION

The development in semiconductor technologies enables manufacturing of systems with multiple identical ICs but enforces in-field test as manufacturing test only is not sufficient. It is most desirable to make use of the same test during production test and in-field test; hence, embed the high-quality deterministic tests. However, as systems are increasingly advanced, higher test data volumes are needed, leading to larger memory requirements. In this paper, we propose an embedded test controller that only requires to store test data for each unique component. The test controller creates on-the-fly boundary scan (IEEE 1149.1) tests for given commands. The advantage of the approach is that only one test per unique component is required. Further, the approach allows flexibility in defining test commands such that different combinations of tests can be applied. We have implemented the test controller and verified it on real hardware and made experiments on ITC'02 benchmarks and industrial circuits. The proposed scheme shows significant reduction in memory requirements.

REFERENCES

[1] I. W. Group, "International Technology Roadmap for Semiconductors (ITRS)," On the ITRS web site, 2009, http://www.itrs.net/home.html.

[2] A. El-Maleh, E. Khan, and S. A. Zahir., "A Geometric-Primitives-Based Compression Scheme for Testing Systems-on-a-Chip," in *VTS*, Aug. 2001, pp. 54–59.

[3] K. J. Balakrishnan and N. A. Touba, "Matrix-Based Test Vector Decompression Using an Embedded Processor," in *DFT*, Nov. 2002, pp. 159–165.

[4] A. Jas and N. A. Touba, "Using an Embedded Processor for Efficient Deterministic Testing of Systems-on-a-Chip," in *ITC*, Oct. 1999, p. 418.

[5] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "Logic BIST for large industrial designs: Real issues and case studies," in *ITC*, 1999, pp. 358–367.

[6] D. Das and N. A. Touba, "Reducing Test Data Volume Using External/LBIST Hybrid Test Patterns," in *ITC*, Oct. 2000, pp. 115–122.

[7] Y. Wu and P. MacDonald., "Testing ASICs with Multiple Identical Cores," *TCAD*, pp. 327–336, 2003.

[8] Li, Yanjing and Makar, Samy and Mitra, Subhasish, "CASP: concurrent autonomous chip self-test using stored test patterns," in *DATE*, Mar. 2008, pp. 885–890.

[9] E. J. Marinissen, V. Iyengar, K. Chakrabarty, "A Set of Benchmarks fo Modular Testing of SOCs," in *ITC*, Oct. 2002, pp. 519–528.

[10] Z. Wang and K. Chakrabarty., "Test data compression for IP embedded cores using selective encoding of scan slices," in *ITC*, Nov. 2005, pp. 581–590.