Combined Test Data Compression and Abort-on-Fail Testing

Erik Larsson
Embedded Systems Laboratory
Department of Computer Science
Linköpings Universitet
Sweden
contact: erila@ida.liu.se

Abstract -- **The increasing test data volume needed for the testing of System-on-Chips (SOCs) leads to high Automatic Test Equipment (ATE) memory requirement and long test application times. Scheduling techniques where testing can be terminated as soon as a fault appears (abort-on-fail) as well as efficient compression schemes to reduce the ATE memory requirement have been proposed separately. Previous test data compression architectures often make use of Multiple Input Signature Response Analyzers (MISRs) for response compression. Therefore, abort-on-fail testing and diagnostic capabilities are limited. In this paper, we propose an SOC test architecture that (1) allows test data compression, (2) where clock cycle based as well as pattern-based abort-on-fail testing are allowed and (3) diagnostic capabilities are not reduced. We have performed experiments on ISCAS designs.**

I. Introduction[1]

It is due to the technology development becoming common practice to design and test system-on-chips (SoCs) in a modular fashion. In a core-based design environment, pre-designed and pre-verified blocks of logic, often called modules or cores, are integrated to a system that is placed in a single chip, making it a system chip. The advantage with modular-design is that complex systems can be designed in a reasonable time. From a test perspective, the advantage is that each module has its dedicated test vectors, and can be tested as a separate unit. However, the test data volume needed to test these system chips is increasing. It is therefore becoming difficult to make the test data fit the memory of the Automatic Test Equipment (ATE), and further, the high test data volume leads to long test application times.

Several approaches to compress test data have therefore been proposed [3], [4], [5], [6]. These schemes make use of the fact that there is a high number of unspecified bits, so called don't care bits, in the test data volume; typically only 1%-5% of all bits are specified. The compression schemes fill the don't care bits in a suitable manner and then make use of a compression algorithm. Different compression schemes are used for test stimuli compression. Chandra and Chakrabarty make use of Golumb coding [3] and Frequency-Directed-Run-Length (FDR) Codes [4]. Ichihara *et al.* explore the use of Statistical Coding, Volkerink *et al.* investigate the use of Packet-based techniques [6] and Li and Chakrabarty make use of dictionaries with Fixed-Length Indices [8].

Common for the approaches is that test stimuli is compressed off-chip and decompressed on-chip while test responses are compressed on-chip and analyzed off-chip. A common solution to on-chip test response compression is the usage of Multiple Input Signature Response Analyzer (MISR). A MISR merges each produced response with the previous responses and the final result is shifted out when all test stimuli have been applied.

The main disadvantage with a MISR-based architecture is that the whole test sequence has to be applied before it is known if a fault is present or not. If the first vector detects a fault, it is not known until the whole sequence has been applied. Also, diagnostic capability is limited as only a signature that contains the signature is produced. If a fault is detected it is difficult to backtrack to find which vector detected the fault.

Iyengar *et al*. [9], Goel and Marinissen [11], Larsson *et al.*[10] have all proposed test scheduling techniques to minimize the test application time. Larsson *et al*. [2] showed that abort-on-fail testing (termination as soon as a fault appears) on module-level can save significant in test application time. Ingelsson *et al.* [1] showed that even further test time reduction can be achieved if testing is allowed to be aborted on clock-cycle and pattern basis. However, clock-cycle and pattern based abort-on-fail testing is not applicable when a MISR compresses the test response.

In this paper we propose an SOC test architecture that allows combined test data compression and abort-on-fail testing. The architecture allows off-line compression of test stimuli and expected test response. The advantage is that compression schemes with high compression capability can be used both for test stimuli and test response while using abort-on-fail testing.

II. Test Architectures for Test Data Compression

Given are a system consisting of a set of modules and test data for each module. The test data consists of test stimuli and expected test responses. Traditionally, the test stimuli and the expected test responses are stored in the ATE. The test stimuli

[1]. The research is partially supported by the Strategic Integrated Electronic Systems Research (STRINGENT) programme.

are applied to the system under test and the produced test responses are compared with the expected test responses.

An example system with three cores tested with an ATE is shown in Figure 1. The ATE feeds test stimuli to the SoC and the produced test responses are compared with the expected test response in the ATE.

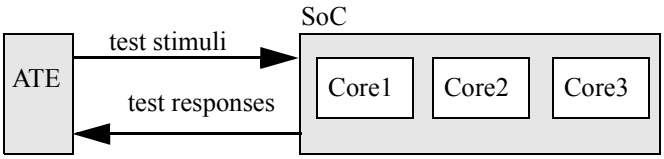


Figure 1: Example system.

A. MISR-based Compression Architecture

The problem with testing as in Figure 1 is that the test data volume is increasing rapidly; hence it is becoming problematic to make the test data volume fit the ATE. Compression techniques take advantage of the fact that the test data volume contains a high number of don't care bits (the bit can be either 1 or 0 as it does not affect the outcome of the test). The don't care bits are set in a way that favor high test data compression. Actually, as the test architecture used in these approaches is based on MISRs, only test stimuli are compressed off-line. The produced test responses are compressed on-chip in a dedicated MISR. Figure 2 shows the conceptual view for the example system (Figure 1) in a compression-based approach, and Figure 3 highlights Core2. Figure 3 shows that compressed test stimuli stored in the ATE are sent to the SoC and decompressed by decompression logic. The decompressed test stimuli are applied to the core and the produced test responses are compressed into the MISR. At the end of the test application, the signature in the MISR is shifted out and after analysis it comes clear if the module is faulty or fault-free.

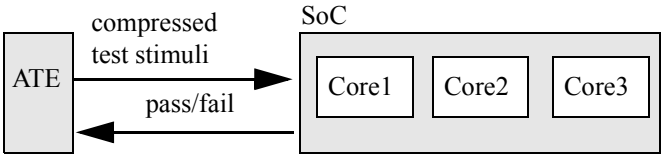
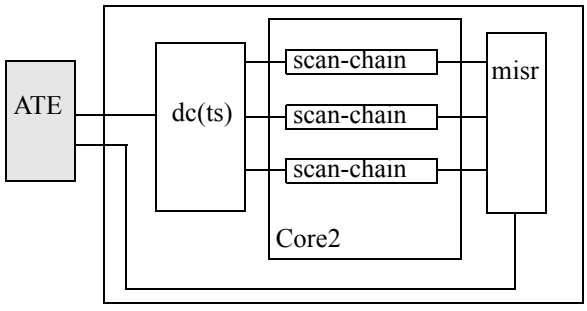


Figure 2: Conceptual view of a MISR-based test compression architecture.



dc:decompression

Figure 3: Core 2 high-lighted in the MISR-based architecture (Figure 2).

The major advantage with a MISR for test response compression is that expected test responses can be compressed on-chip; hence the responses do not have to be stored in the ATE.

Among the disadvantages are that simulation is required to design the MISR, and once the MISR is designed, the test data volume is fixed, the order test stimuli are applied is fixed, the outcome of the test is only known at the end of the testing. And, diagnostic capability is limited.

MISR-design requires simulation in order to fill the don't care bits in the expected test response. Assume that core 2 (Figure 3) is a module with 3 scan-chains each of length 3. The test stimuli and the expected test responses are then of 12 bits each. Assume that Core 2 is tested with three vectors only, given as follows:

vector, {test stimuli}, {expected test response}
1, { 1xx xx0 xxx} { x0x xxx x11}
2, { xx1 xx1 xxx} { 1xx 0xx xxx}
3, { 0x0 xxx xxx} { xxx 1xx xxx}

The don't care bits (x) in the test stimuli can be filled in an arbitrary way. Two valid examples for vector 1 are {111 110 000} and {100 000 000}. However, the two different fills result most likely in different expected test response. And, the expected test response is needed in order to design the MISR; hence, after the x-filling of the test stimuli, simulation must be used.

The MISR is designed in such a way that it in operation collects and merges the produced test response with the current contents in the MISR. The order in which the MISR is designed for must be kept; it is not possible to change the order in which the stimuli are applied and the whole sequence must be applied in order to get the final response. This prevents abort-on-fail testing where the testing is terminated as soon as a fault is detected. Further, as the whole sequence must be applied, diagnostics is problematic. The signature is shifted out and to trace the fault, back-tracking must be used, which is problematic.

B. Proposed SOC test architecture

The proposed SOC test architecture is shown in Figure 4. Figure 4 highlights, for the design in Figure 1, the decompression logic and the output logic for core 2. In the architecture, the test stimuli are compressed off-line and stored in the ATE as in previous compression approaches. The proposed scheme shows similarities with the technique proposed by Nahvi and Ivanov [7]. However, in the technique by Nahvi and Ivanov test stimuli and expected test responses are sent to the system but no compression is allowed and no masking scheme is used. Ollivierre *et al.* [12] explore sending test stimuli and expected responses to the system. A masking register is used to unmask the responses. The focus is to finding a minimal number of reloads of the masking register. In current proposal, a dedicated mask is used for each response.

Different from previous approaches are that the expected test responses and mask data are also compressed off-line and stored in the ATE. For test application, compressed test stimuli, compressed expected test responses and the compressed mask

data are all transported into the system under test. The test stimuli are decompressed and applied to the system. The expected test responses are decompressed and compared with the produced test responses. The mask data are decompressed, and used to determine which bits in the expected test responses and the produced test responses that should be compared. After any shift-out clock cycle, produced test stimuli and expected test stimuli are compared, and a pass/fail signal is generated. The pass/fail bit is generated immediately, in contrast to the MISR-based architecture; hence time saving abort-on-fail testing can be used.

Assume that core 2 (Figure 3) is a module with 3 scan-chains each of length 3. The test stimuli and the expected test responses are then of 12 bits each. Assume that core 2 is tested with three vectors only, given as follows:

vector, {test stimuli}, {expected test response}
1, {1xx xx0 xxx} {x0x xxx x11}
2, {xx1 xx1 xxx} {1xx 0xx xxx}
3, {0x0 xxx xxx} {xxx 1xx xxx}

In the proposed scheme, test stimuli and test response are filled according to any filling scheme. The mask data are used to determine where care bits in the expected response are placed. A "1" in the mask data means that the bit is a care bit (not a don't care bit) and a "0" indicate that the bit is a don't care bit. The following is the result to be compressed according to an arbitrary filling scheme:

vector, {stimuli}, {expected response} {mask data}
1, {111 110 000} {000 000 011} {010 000 011}
2, {111 111 111} {111 000 000} {100 100 000}
3, {000 000 000} {000 111 111} {000 100 000}

1) The output logic

The input to the output logic are the *produced test response* (*pr*), the decompressed *expected test response* (*er*), and the decompressed mask (*m*) (see Figure 4).

For example, assume that: *pr* = {100 010 111}, *er* = {000 000 011}, *m* = {010 000 011}.

Note, that *pr* and *er* are different; hence one could assume a fault. However, there is no fault as the original expected test response is {x0x xxx x11}. The mask (*m*) determines which of the bits that should be compared. In the example above, bit 2 (counting from left-side), 8 and 9 should be compared. Excluding all other bits than these three gives the following:

pr = {-0- --- -11}, *er* = {-0- --- -11}, and then *pr*=*er*.

If *er*=*pr* and *m*=1 the output should be 1 to indicate fault-free. However, if *er*≠*pr* and *m*=1 the output should be "0" to indicate a fault. In all cases when *m*=0, the output should be "1". It indicates that the bit is not important. The function is:

$$output = m \cdot (\overline{er} \cdot pr + er \cdot \overline{pr})$$

The implementation of the logic function is shown for one bit in Figure 5, and for each of the four output bits at core 2 in Figure 6. Note that the evaluation chain can be chained, which means that only a single output is needed. Furthermore, scan-elements can be inserted at each bit-output to ease diagnostic. If a fault is detected, the contents in added scan-elements can be shifted out and analyzed.

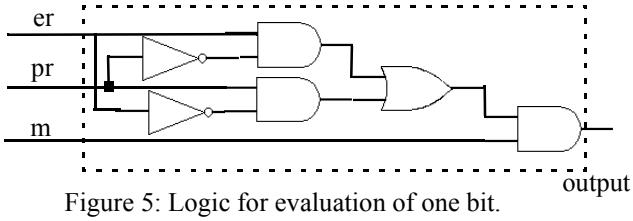
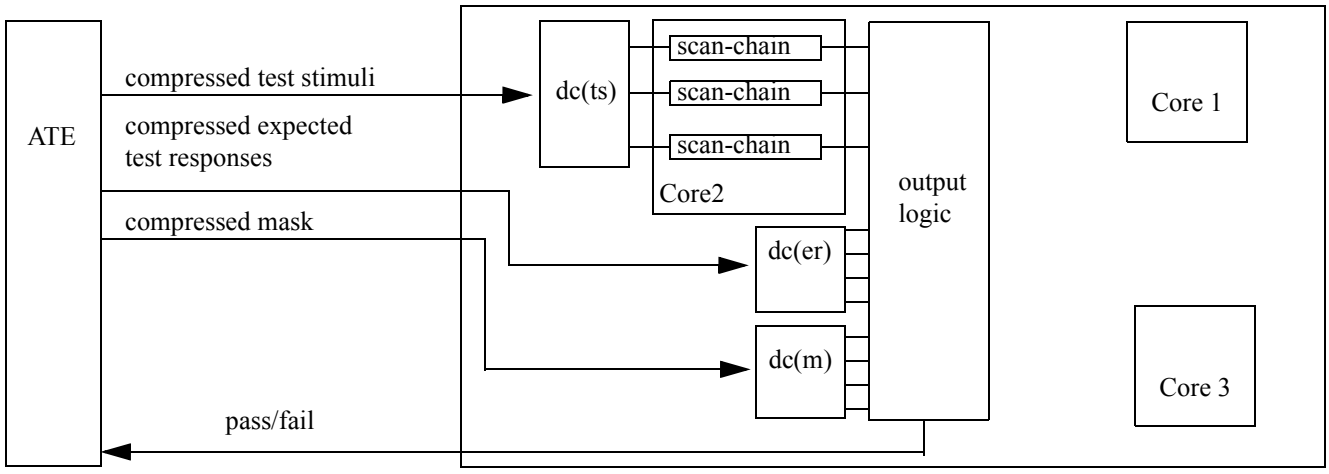


Figure 5: Logic for evaluation of one bit.

III. Experimental Results

We have taken ISCAS design and implemented a compression technique based on the Fixed-Length Indices proposed by Li and Chakrabarty [8]. We have compressed the test data volume assuming that the above proposed architecture is used. The results are presented in Table 1. For each module, we list the compression ratio on the test stimuli, test responses, and the mask data. We have not compared our scheme with MISR-based approaches as these approaches only compress the test



dc(ts):decompression of test stimuli
dc(er):decompression of expected test responses
dc(m):decompression of mask

Figure 4: Proposed Test Architecture.

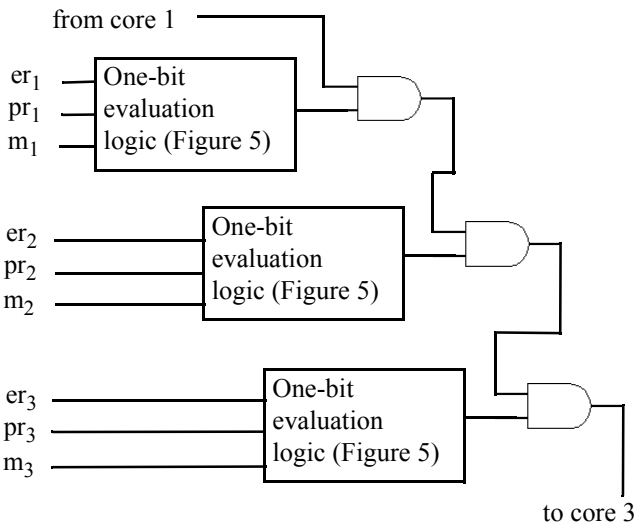


Figure 6: Logic for evaluation at core 2 in Figure 4.

stimuli.

TABLE 1. Compression ratio for the ISCAS designs.

Designs	Compression ratio (%)		
	test stimuli	test responses	mask data
c7552	36.6	26.5	8.4
s838	59.0	57.7	55.5
s9234	55.9	56.5	29.6
s38584	64.0	62.8	34.4
s13207	80.2	76.5	56.0
s15850	60.5	60.0	39.4
s5378	60.6	58.7	39.4
s35932	46.6	89.8	86.0
s38417	62.4	58.9	6.7

IV. 8.Conclusions

The increasing test data volume is becoming a major obstacle in System-on-Chip (SOC) test design as it leads to long test application times and high Automatic Test Equipment (ATE) memory requirements. Test scheduling techniques and test compression schemes have been proposed in order to solve the test time minimization problem respectively the test data volume problem separately. The test time can be efficiently reduced by using an abort-on-fail scheme where testing is terminated as soon as a fault is detected. It has been shown that allowing test termination on clock-cycle and pattern-base are efficient for test time reduction. However, compression architectures are generally based on Multiple Input Signature Analyzers (MISR) for on-chip compression of the test response; hence per pattern and per cycle termination is not possible as the response is shifted out from the MISR only at the end of the testing when all test stimuli have been applied. A further disadvantage with a MISR-based architecture is that diagnostics is cumbersome. A key factor in order to combine test scheduling and test data volume compression is the SOC test architecture. In this paper, we propose an SOC test architecture that allows abort-on-fail testing with termination per clock-cycle or per pattern. As the architecture allows such termination, it is suitable for diagnostics as well. We have performed illustrative experiments on the ISCAS designs.

References

[1] U. Ingelsson, S. Goel, E. Larsson, and E. J. Marinissen, "Test Scheduling for Modular SOCs in an Abort-on-Fail Environment", *Proceedings of European Test Symposium (ETS'05)*, Tallinn, Estonia on May 22-25, 2005.

[2] E. Larsson, J. Pouget, and Z. Peng, "Defect-Aware SOC Test Scheduling", *Proceedings of VLSI Test Symposium (VTS'04)*, Napa, CA, USA, April 2004.

[3] A. Chandra and K. Chakrabarty, "System-on-a-Chip Test Data Compression and Decompression Architectures Based on Golomb Codes", *Transactions on CAD of IC and Systems*, pp. 355-367, Vol. 20, No. 3, 2001.

[4] A. Chandra and K. Chakrabarty, "Frequency -Directed-Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression", *Proceedings of VLSI Test Symposium (VTS)*, pp. 42-47, 2001.

[5] H. Ichihara, A. Ogawa, T. Inoue, and A. Tamura, "Dynamic Test Compression Using Statistical Coding", *Proceedings of Asian Test Symposium (ATS)*, pp. 143-148, Kyoto, Japan, November 2001.

[6] E. H. Volkerink, A. Khoche, and S. Mitra, "Packet-based Input Test Data Compression Techniques", *Proceedings of International Test Conference (ITC)*, pp. 154-163, Baltimore, MD, USA, October 2002.

[7] Mohsen Nahvi and Andre Ivanov, "Indirect Test Architecture for SoC Testing", Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23 , Issue 7, July 2004, pp. 1128 - 1142.

[8] L. Li and K. Chakrabarty, "Test Data Compression Using Dictionaries with Fixed-Length Indices", VLSI Test Symposium (VTS), 27 April - 1 May 2003, Napa Valley, CA, USA, pp. 219-224.

[9] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores", Journal of Electronic Testing: Theory and Applications (JETTA), 18(2):213-230, April 2002.

[10] E. Larsson, K. Arvidsson, H. Fujiwara and Z. Peng, "Efficient Test Solutions for Core-based Designs", Transaction on Computer-Aided Design of Integrated Circuits and Systems.Volume: 23, Issue:5, May 2004, pages:758 - 775.

[11] S. K. Goel and E. J. Marinissen, "SOC Test Architecture Design for Efficient Utilization of Test Bandwidth", ACM Transactions on Design Automation of Electronic Systems (TODAES), 8(4):399-429, October 2003.

[12] S. Ollivierre, A. B. Kinsman, and N. Nicolici, "Compressed Embedded Diagnosis of Logic Cores", *International Conference on Computer Design (ICCD'04)*, 11-13 October 2004, San Jose, CA, USA, pages 534-539.