An Embedded Core DFT Scheme to Obtain Highly Compressed Test Sets

Abhijit Jas, Kartik Mohanram, and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
E-mail: {jas, kmram, touba}@cat.ece.utexas.edu

Abstract

This paper presents a novel design-for-test (DFT) technique that allows core vendors to reduce the test complexity of a core they are trying to market. The idea is to design a core so that it can be tested with a very small number of test vectors. The I/O pins of such a "designed for high test compression" (DFHTC) core are identical to the I/O pins of an ordinary core. For the system integrator, testing a DFHTC core is identical to testing an ordinary core. The only difference is that the DFHTC core has a significantly smaller number of test vectors resulting in less test data as well as less test time (fewer scan vectors). This is achieved by carefully combining a parallel "test per clock" BIST scheme inside the core with the normal external testing scheme using a tester. The BIST structure inside the core generates weighted pseudo-random test vectors which detect a large number of faults in the core. Results indicate that such DFHTC cores have a significantly smaller number of test vectors than their ordinary counterparts thereby greatly reducing test time and test storage.

1. Introduction

Core-based design and reuse is emerging as a new paradigm for the design of integrated circuits. System integrators construct a system-on-a-chip using pre-designed and pre-verified cores as building blocks. System integrators can purchase cores from various core vendors. This creates a competitive environment where multiple core vendors are trying to sell cores with similar functionality. As the complexity of systems-on-a-chip continues to increase, the difficulty and cost of testing such chips is escalating rapidly [Chandramouli 96], [Zorian 98]. One characteristic of a core that emerges as an important distinguishing factor is test complexity. Given two cores with similar functionality, the core that can be thoroughly tested with the smallest amount of test data and the simplest tester program has a significant competitive advantage because it reduces manufacturing test costs.

In this paper, a novel design-for-test (DFT) technique that allows core vendors to reduce the test complexity of the core they are trying to market is presented. The idea is to create a DFHTC core which can be tested with a significantly smaller number of test vectors compared to the ordinary core (as illustrated in Fig. 1). The I/O pins of the DFHTC are identical to the I/O pins of an ordinary core. There is a "scan data in" pin (SDI), "scan data out" pin (SDO), and a "scan enable" (SE) pin to control the scan chain (as illustrated in Fig. 2). For the system integrator, testing a DFHTC core is identical to testing an ordinary core. The only difference is that the DFHTC core has a significantly smaller number of test vectors than that of its ordinary counterpart resulting in less test data as well as less test time (fewer scan vectors). The tester program that is required for testing a DFHTC is no different than that required for testing an ordinary core. Thus, from the system integrator's point of view, a DFHTC core is identical to an ordinary core in all respects except that it has a much smaller test set. Internally, however, the actual number of test vectors that are applied to the core is much larger. The additional test vectors are weighted pseudo-random vectors that are generated internally inside the core and is completely transparent to the system integrator.

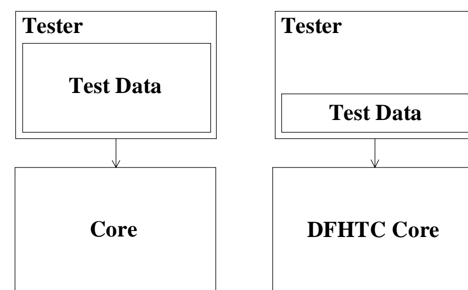

Figure 1. Concept of a DFHTC Core

The problem of reducing the test data and test time for cores has been attacked from several different angles in recent literature [Aerts 98], [Jas 98], [Rajski 98], [Sugihara 98]. Scan chain architectures for core-based

designs that maximize bandwidth utilization are presented in [Aerts 98]. A technique for compression/decompression of scan vectors using cyclical decompressors and run-length coding is described in [Jas 98]. Since a DFHTC core is fully compatible to an ordinary core, all techniques for optimal testing of cores with scan can be applied in the same manner to DFHTC cores. Another approach for reducing test time is to use built-in self-test (BIST). A modular BIST approach that allows sharing of BIST control logic among multiple cores is presented in [Rajski 98].

Designing a core with BIST is an alternative to designing a DFHTC core. However, there are several advantages to developing a DFHTC core:

- It is non-trivial to achieve high fault coverage with BIST alone. Inserting test points to improve fault coverage degrades performance. In many cases, it may be undesirable to modify the function logic.
- Pure BIST requires long test lengths which can add to tester socket time (i.e., the time that the chip sits in the tester socket).
- Developing a tester program for handling a core with BIST may be more complicated for the system integrator if all the other cores are conventional scan designs that are externally tested.
- A DFHTC core is compatible with all other cores with scan chains, hence the same test integration methodologies and tools can be used.
- By combining weighted pseudo-random vectors with deterministic test vectors a DFHTC core achieves a high fault coverage with a very small number of deterministic test vectors.

For these reasons, the system integrator may prefer a DFHTC core to one with BIST.

A novel technique for combining BIST and external testing across multiple cores is described in [Sugihara 98]. However there are several disadvantages with this technique. It requires multiple test sets for each core with each test set having different components for BIST and external testing. It requires a considerable amount of hardware for scheduling the tests for the different cores. Moreover, since it is based on running BIST simultaneously for all but one of the cores on the chip, it suffers from large power dissipation during testing. This paper presents a simpler and more practical approach for combining external testing and BIST.

This paper is organized as follows: Section 2 discusses the implementation details of the scheme. Section 3 describes a procedure for obtaining a highly compressed set of scan vectors that provides the desired fault coverage. Experimental results are shown in Sec. 4. Section 5 is a conclusion.

2. Implementing a *DFHTC* Core

Having described the concept of a DFHTC core, now the details of how it is designed will be discussed. It is best explained with an example. Figure 2 shows a DFHTC core. The test vectors are applied to the core by a number of *pseudo-random pattern generators* (PRPGs) which collectively produce one test vector per clock cycle. The PRPGs are formed by adding logic to the scan chain so that it can be configured as multiple PRPGs during testing. Note that these PRPGs are transparent outside the core. The controller provides the interface to the outside world (i.e., the tester) and has a *serial-in parallel-out* shift register inside it. The size of this shift register is the same as that of the PRPGs. The tester shifts in the scan vectors into this shift register from which they are transferred to the PRPGs by the controller and applied to the CUT. Each scan vector may be conceptually thought of as being composed of a number of b bit blocks. Thus if there are N such blocks, then the total length of a scan vector is bN. Thus a "test per scan" approach (as is the case with external tester driven testing) would imply a test vector is applied once each bN clock cycles. In our scheme, we utilize these scan shift clock cycles which are otherwise wasted to apply additional test vectors to the CUT. Ideally we should be able to apply bN tests to the CUT in bN clock cycles. In the following paragraph we explain how such a "test per clock" scheme can be implemented in the "test per scan" framework without modifying the external interface of the core.

As explained earlier, the test vectors are actually applied to the CUT through the PRPGs in parallel thus implementing a "test per clock" scheme. At the very beginning, the PRPGs are initialized with some starting seed (as is done in BIST). The collection of all these seeds in the PRPGs forms the first test vector that gets applied to the CUT. Now the tester starts shifting data to the core. As it starts shifting in the first scan vector, the PRPGs are also started in autonomous mode. These PRPGs generate pseudo-random test vectors at the inputs of the CUT. The output response of the CUT is compacted in a *multiple input signature register* (MISR). The feedback line of the MISR forms the "scan data out" (SDO) pin. At every clock cycle when the tester shifts in one bit of the scan vector, it can scan out one bit through the SDO pin. This makes the output response of the DFHTC core look like that of an ordinary core. Using a MISR introduces the possibility of losing fault coverage due to aliasing. This can be avoided by either doing fault simulation with the MISR when generating the test vectors, or by choosing the size of the MISR so that the probability of aliasing is sufficiently low.

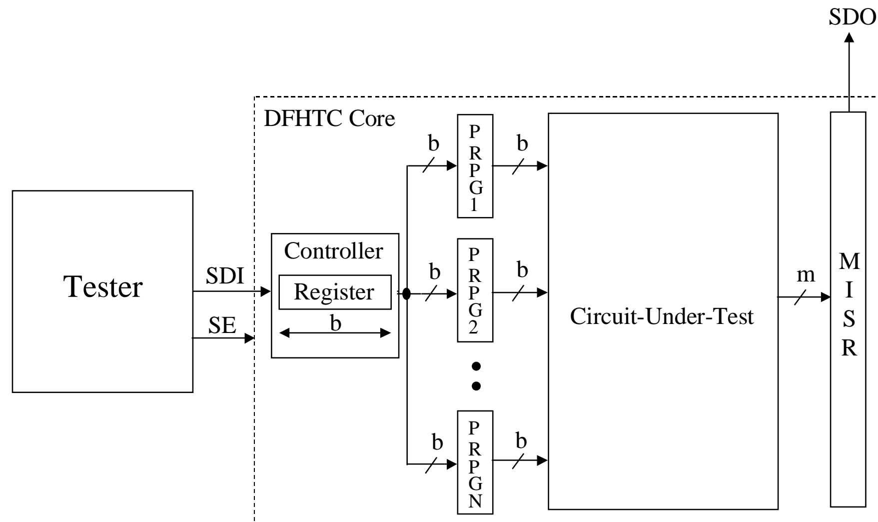

Figure 2. Test Architecture of a DFHTC Core

t_1	1111 1000 0011
t_2	0111 0100 0001
	0011 0010 1000
	0001 1001 0100
•	**1000** 1100 0010
	1000 0110 1001
•	**1000** 1011 1100
	1000 0101 0110
•	**1000 1010** 1011
	1000 1010 0101
	1000 1010 1010
t_{12}	**1000 1010** 1101
$\mathbf{t_{13}}$	**1000 1010 1110**

Figure 3. Example of Internally Generated Weighted Pseudo-Random Test Vectors

As soon as the controller receives the first *b* bits of a scan vector, it loads it in parallel into the first PRPG (the PRPGs may need to be modified to a small extent to have this parallel loading capability if they do not already have one) and "locks" it. A locked PRPG does not produce new patterns at every clock cycle but continues to hold the same pattern which it held last before it was locked (this can be done either by gating out the clock or disabling an enable signal). However, the rest of the PRPGs are "free", and they continue to run while the tester continues to shift in the next *b* bits of the scan vector. Thus, during these *b* clock cycles, *b* pseudo-random vectors get applied to the CUT, all of which have the same pattern in the set of inputs being fed by the first PRPG. Hence these pseudo-random vectors are "weighted" in a certain set of inputs which continues to be held at a fixed pattern. Note that these weights have actually been derived from an external deterministic scan vector and thus have a greater probability of detecting yet undetected faults. When the tester has received the second set of *b* bits of the scan vectors, it loads them in parallel into the second PRPG and locks it. Thus during the next *b* cycles, two of the PRPGs are locked while the rest are free and continue to produce more pseudo-random vectors which are more weighted than those in the earlier *b* clock cycles (as these have two PRPGs locked). This process continues until the last set of *b* bits of the scan vector is received by the controller. As this last set gets loaded into the *N*-th PRPG, we have the desired scan vector ready to be applied to the core. Once this test vector is applied to the CUT, all the PRPGs are unlocked again. The whole process is started all over again, i.e., while the first set of *b* bits of the next scan vector are being shifted into the controller by the tester all the free PRPGs run and produce a "test per clock". Thus for every *bN* bits long deterministic scan vector we are actually applying *bN* weighted pseudo-random test vectors which detect a large number of the remaining faults in the circuit. As shown in the experimental results section later in the paper, this greatly reduces the number of test vectors which are needed to be stored on the tester. Also, note that all these additional weighted pseudo-random vectors are more likely to detect new faults than the usual equiprobable pseudo-random vectors produced by a PRPG as their weights are part of an ATPG generated test vector. This phenomenon has been observed in [Pomeranz 93], [AlShaibi 94], [Touba 95], and [Tsai 97].

Fig. 3 illustrates the scheme of internally generating the weighted pseudo-random vectors which get applied to the CUT during the course of normal external testing. The highlighted portions of the test vectors have been shifted into the core from outside (i.e., from the tester) and the PRPG has been locked. The other portions of the test vectors have been internally generated by running the PRPG in autonomous mode.

The scheme described above has several other variations which in some cases produces a better compression. In the scheme described above, the PRPGs are locked in a fixed sequence i.e., the i-th PRPG is always locked before the j-th PRPG if $i < j$. One alternative would be to implement a *round-robin* scheme where for the first deterministic scan vector the first PRPG is locked first followed by the second, third and so on; for the second deterministic test vector the second PRPG is locked first followed by the third, fouth and so on eventually ending with the first. Another approach would be to implement a *select* scheme. In this scheme, we prefix every scan vector with an s bit codeword which indicates to the controller which locking order is to be followed when shifting in that particular scan vector (there can be $k=2^s$ fixed predetermined locking orders). However, all these other schemes make the controller more complex and consequently increase hardware overhead. Hence deciding which scheme to implement involves a trade-off between the amount of compression and the hardware overhead.

3. Constructing a Highly Compressed Scan Vector Test Set

In this section, an algorithm is described that the core vendor can use to obtain a highly compressed test set for the core based on the proposed scheme. Fig. 4 gives a flow chart for the algorithm. In the algorithm, it is assumed that we have a CUT with bN inputs (N PRPGs each b bits wide).

The first step is to generate b pseudo-random test vectors by simulating the PRPGs. These are the test vectors that are initially applied to the core when all the PRPGs are free. Fault simulation is done to drop all the faults that are detected by these b pseudo-random test vectors. An ATPG tool is then used to target some undetected fault f in the fault-list. The resulting test vector t obtained from ATPG affects the next $b(N\text{-}1)$ test vectors that are applied to the CUT (as explained in Sec. 2). These test vectors are computed by locking the appropriate PRPGs in turn and simulating the remaining free PRPGs to generate the $b(N\text{-}1)$ weighted pseudo-random test vectors. Fault simulation is performed to drop all the faults that are detected by test vector t and all of the $b(N\text{-}1)$ weighted pseudo-random test vectors. If the fault-list is not empty at this stage, all the above steps are repeated. This continues until the fault-list becomes empty (i.e., all detectable faults are detected) or the desired fault coverage has been achieved.

The final set of test vectors that the core vendor gives the system integrator consists of only the deterministic test vectors generated in the ATPG step of the above algorithm.

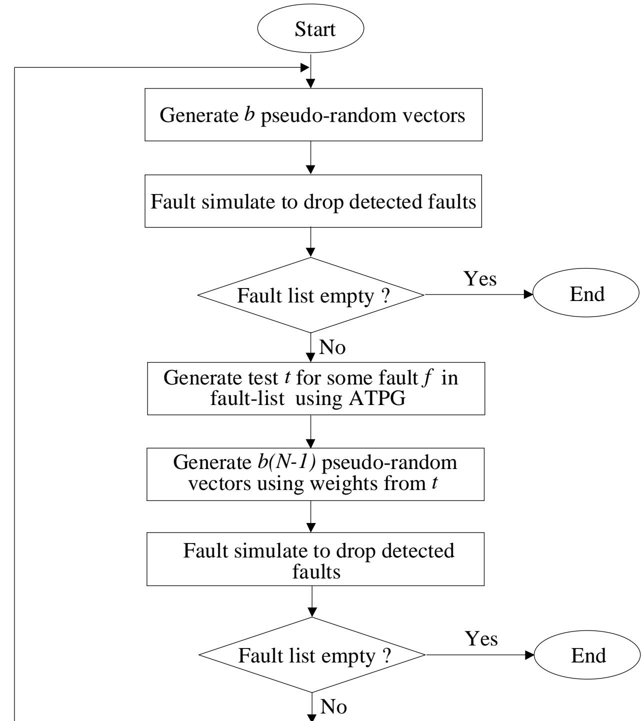

Figure 4. Flowchart for Generating Highly Compressed Test Set

4. Experimental Results

Experiments were performed for the largest ISCAS benchmark circuits [Brglez 89]. Table 1 shows the results comparing a DFHTC core test set with that of an ordinary core. The fault coverage in both cases is 100% of detectable faults. For the ordinary core, the following are shown: the number of test vectors with ATPG and static compaction, and the total amount of test data (that must be stored on the tester). For the DFHTC core, the following are shown: the number of PRPGs, the number of test vectors which results from using our scheme, and the total amount of test data. Lastly, the percentage reduction in the number of scan cycles required for

Table 1. Results Comparing Test Sets of DFHTC and Ordinary Core Using Same ATPG Software

Circuit Name	Scan Size (bits)	Normal Core		DFHTC			% Reduction in Test Data
		Num. of Test Vectors	Test Data (bits)	Num. PRPGs	Num. of Test Vectors	Test Data (bits)	
s5378	199	181	72038	4	49	19502	72.9
				8	48	19104	73.5
s9234	247	198	97812	4	85	41990	57.1
				8	78	38532	60.6
s13207	700	266	372400	4	33	47200	87.6
				8	28	39200	89.5
s15850	611	153	186966	8	56	68432	63.4
				16	60	73320	60.1
s38417	1664	270	898560	8	156	505856	42.2
				16	142	472576	47.4
s38584	1464	213	623644	8	33	93696	84.5
				16	32	96624	85.0

testing each circuit is shown. The percentage is computed as follows:

[(Original Scan Test Data) - (DFHTC Core Scan Test Data)] / (Original Scan Test Data) x 100

As can be seen from the results, the number of test vectors required to test the DFHTC core is substantially less than that of its ordinary counterpart. Consequently, the total amount of test data is reduced and the number of scan cycles for testing the circuit is also reduced. As can be seen, the number of PRPGs has a relatively small effect on the amount of compression.

In terms of area overhead, the scheme requires a shift register, controller, and additional logic to configure the scan chain into PRPGs. We synthesized the controllers and found that they ranged from 204 to 257 two-input gate equivalents for the benchmark circuits shown in Table 1.

5. Conclusion

A DFHTC core reduces test time and tester memory requirements. It provides some nice advantages compared with using a pure BIST approach because it is fully compatible with ordinary cores, i.e., it has the same test I/O pins and can use the same tester program. Hence, the system integrator does not need to treat a DFHTC core any different than other cores.

A DFHTC core will reduce test costs for the system integrator. Thus, core vendors may find DFHTC cores a means to achieve a competitive advantage in selling their cores.

The general idea of combining BIST and external testing for cores is an attractive one. A hybrid test approach allows the test time and test storage to be reduced as with BIST, and it provides a high fault coverage without having to insert test points as with external testing. The technique described here is one possible approach for combining BIST and external testing, but there are opportunities for further research in this area.

Acknowledgements

This material is based on work supported in part by the National Science Foundation under Grant No. MIP-9702236, and in part by the Texas Advanced Research Program under Grant No. 1997-003658-369.

References

[Aerts 98] Aerts, J., and E.J. Mariniseen, “Scan Chain Design for Test Time Reduction in Core-Based ICs,” *Proc. of International Test Conference*, pp. 448-457, 1998.

[AlShaibi 94] AlShaibi, M.F., and C.R. Kime, “Fixed-Biased Pseudorandom Built-In Self-Test for Random Pattern Resistant Circuits,” *Proc. of International Test Conference*, pp. 929-938, 1994.

[Brglez 89] Brglez, F., D. Bryan, and K. Kozminski, “Combinational Profiles of Sequential Benchmark Circuits,” *Proc. of International Symposium on Circuits and Systems*, pp. 1929-1934, 1989.

[Chandramouli 96] Chandramouli, R., and S. Pateras, “Testing Systems on a Chip,” *IEEE Spectrum*, pp. 42-47, Nov. 1996.

[Jas 98] Jas, A., and N.A. Touba, “Test Vector Compression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs,” *Proc. of International Test Conference*, pp. 458-464, 1998.

[Pomeranz 93] Pomeranz, I., and S.M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 7, pp. 1050-1058, Jul. 1993.

[Rajski 98] Rajski, J., and J. Tyszer, "Modular Logic Built-In Self-Test for IP Cores," *Proc. of International Test Conference*, pp. 313-321, 1998.

[Sugihara 98] Sugihara, M., H. Date, and H. Yasuura, "A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem," *Proc. of International Test Conference*, pp. 465-472, 1998.

[Touba 95] Touba, N.A., and E.J. McCluskey, "Transformed Pseudo-Random Patterns for BIST," *Proc. of VLSI Test Symposium*, pp. 410-416, 1995.

[Tsai 97] Tsai, K.H., S. Hellebrand, J. Rajski, and M. Marek-Sadowska, "STARBIST: Scan Autocorrelated Random Pattern Generation," *Proc. of 34th Design Automation Conference*, pp. 472-477, 1997.

[Zorian 98] Zorian, Y., E.J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," *Proc. of International Test Conference*, pp. 130-143, 1998.