Enhancing Silicon Debug via Periodic Monitoring

Joon-Sung Yang and Nur A. Touba

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

Abstract

Scan-based debug methods give high observability of internal signals, however, they require halting the system to scan out responses from the circuit-under-debug (CUD). This is time consuming as many scan dumps may be required. In this paper, conventional scan chains that have non-destructive scan out capability are configured to operate as multiple MISRs during system operation. Information from the multiple MISRs is monitored periodically to identify erroneous behavior. A procedure for constructing the MISRs to maximize debug capability is described. A three step process is used to zero in on the first clock cycle in which an error is present with a small number of scan dumps. Moreover, a method for bypassing errors is described to permit debug in the presence of multiple bugs.

1. Introduction

As technology advances, larger and denser devices are being manufactured with shorter time to market requirements. Identifying and resolving problems in integrated circuits (ICs) are the main focus of the pre-silicon and post-silicon debug process. As indicated in the International Technology Roadmap for Semiconductors (ITRS) [ITRS 05], post-silicon debug is a major time consuming challenge that has significant impact on the development cycle of a new chip.

Trace buffers are commonly used to capture data from some select signals to aid in the debug process [Abramovici 06], [Anis 07a, 07b], [Hopkins 06], [Yang 08]. They provide at-speed signal capture capability over a number of clock cycles which enhances the observability of the internal signals. More information for silicon debug can be achieved by using compression techniques which further improve the visibility for the debug process. However, trace buffer based debug techniques have limited capability due to limited on-chip storage space. The number of signals that are observed and the number of clock cycles over which the signal information is available is limited with trace buffers. The information provided by the trace buffer may not be sufficient to find both temporal (when) and spatial (where) information for failures in the silicon using only trace buffers.

Scan chains are used to support manufacturing testing and can be reused for post-silicon debug to increase debug capability [Carbine 97], [Hopkins 06], [Gu 06]. Scan dumps give high observability of internal signals and states after the occurrence of a triggering event. Scan dumps play a key role in binary search based debug [Yen 06] for observing the state of the circuit-under-debug (CUD). Binary search based debug involves iteratively dividing the search space in half until the first cycle that the error is activated and observed is found. In [Vermeulen 02], methods for using scan chains to further increase observability were introduced. Hardware debug modules integrated with scan chains are added to a chip and provide the capability to start, stop, reactivate, or single step execute the debug process with

the scan chain values being delivered through the IEEE 1149.1 standard test access port (TAP). The drawback of binary search based debug is that it can require a large number of debug sessions to find the first failing cycle where each session requires halting the system to perform a scan dump.

Shadow flip-flops or latches are often used to provide a non-destructive scan out capability that preserves the existing system state. Many systems are fully scannable with non-destructive capability which is helpful for both test and debug [Carbine 97], [Vermeulen 02], [Kuppuswamy 04]. Note that while the system is running, shadow flip-flops or latches are not used for system operation. This fact is exploited in the work.

In this paper, we propose a new debug technique based on reusing non-destructive scan chains. The shadow flip-flops are configured to operate as multiple-input signature registers (MISRs) during system operation. The shadow flip-flops normally do not perform any function when the system operates, however, in the proposed method they are formed into multiple MISRs to enhance silicon debug capability. Compressed information from the multiple MISRs is monitored periodically with externally provided data to identify erroneous behavior. The MISRs are constructed based on structural information of the circuit to maximize their debug efficiency. A three step debug process is used to zero in on the first failing clock cycle trying to minimize the number of scan dumps. By providing high observability of the system state without the need for scan dumps, the proposed method can detect erroneous behavior far earlier than conventional debug methods can. In addition, we propose a debug method to bypass errors which can facilitate downstream debug. Because the presence of a bug may prohibit accurate downstream debug, a faulty response replacement or data masking method may be needed to assist in validating a system.

Sec. 2 describes how to configure multiple MISRs using non-destructive scan chains. Sec. 3 describes the features of the three step debug process, and Sec. 4 describes an error bypassing method, Sec. 5 shows the experimental results, and Sec. 6 concludes this paper.

2. Procedure for Configuring Multiple MISRs

Scan based debug gives greater observability than trace buffer based debug [Anis 07a, 07b], [Yang 08], however, observing the system state requires halting the system to perform the scan dump and hence is not suitable for at-speed debug [Hopkins 06]. It may take many cycles for an error in the system to propagate to a primary output where it can be observed, so there can be a long time gap between from when a bug is invoked and to when it is visible. Due to this latency, it can be time consuming to find the root cause of errors using only a scan dump based debug methodology.

As described in [Gu 02] and [Vermeulen 02], there have been several techniques proposed to reuse DFT logic for silicon debug. In this paper, conventional scan chains with non-destructive scan out capability are reused and configured to operate as a set of MISRs. A periodic checking scheme is proposed to monitor the states of a system without scan dumps. The MISRs configured from the shadow flip-flops in scan chains keep compacting the system state and linear compactors further compress the MISR signatures to greatly reduce the volume of debug data. By having only a very small amount of highly compressed data which represents the system state, it is possible to monitor this data and detect any misbehavior in the circuit much earlier than when it would normally become functionally observable at the chip pins. Structural information is used to configure the multiple MISRs in a way that helps to more rapidly diagnose the root cause of the erroneous behavior. By carefully configuring the MISR signatures, it is possible to extract spatial information about where the error is originating from.

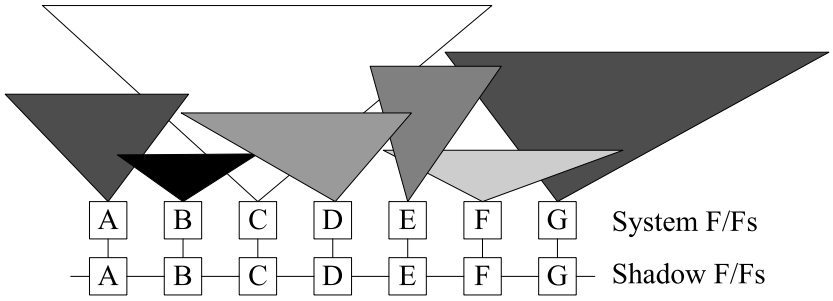

Figure 1. Scan Chain and Logic Cone

Fig. 1 shows an example of the logic cones driving the system flip-flops along with the shadow flip-flops present for non-destructive scan. The flip-flops are labeled as *A*-*G* for the illustrative purposes. By traversing the netlist, the degree of overlap between the logic cones that drive each flip-flop can be determined. The partitioning of flip-flops into multiple MISRs is based on this logic cone analysis. If only a single large MISR was constructed, it would require long wires to generate the feedbacks and may cause other issues related to the physical design. Partitioning the flip-flops into multiple MISRs addresses this problem and can also be used to spatially isolate the candidate error sites to speed up the debug process. The proposed approach is based on partitioning scan cells that are the most structurally related in the design together in the same MISR. This helps to minimize routing of the MISRs as well as maximize spatial diagnosis capability by reducing the probability that an error propagates to multiple MISRs.

In the proposed approach, the MISRs are configured using a graph that represents the degree of logic cone overlap between different flip-flops. In Fig. 2, each node corresponds to a flip-flop in Fig. 1. The weight on the edges corresponds to the amount of logic cone overlap between the logic cones of the corresponding nodes measured in terms of the number of gates that are shared between the two cones. For example, the logic cone driving flip-flop *A* overlaps with cones of *B* and *C,* and the degree of overlap is 20 and 10, respectively. MISRs are generated using an initial clustering procedure followed by an iterative merge & update procedure. The details of algorithm are described in the following subsections.

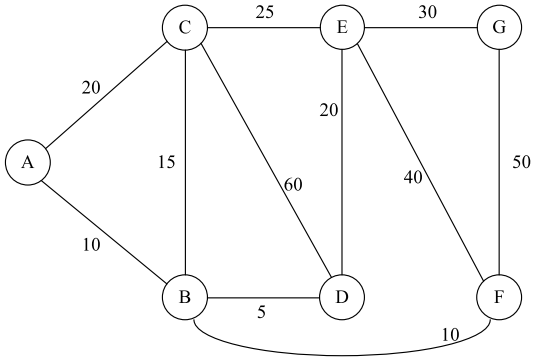

Figure 2. Graph Representation of Logic Cone Relation.

2.1. Initial Clustering Procedure

There are two inputs to the initial clustering procedure, one is the number of MISRs, *n*, and the other is the minimum size of a MISR, *m*, which determines the minimum aliasing probability. The initial clustering procedure select *n* clusters of the *m* most overlapped logic cones in each. The edge with the largest weight is selected as a

starting point for the first cluster. Because the weight represents the logic cone overlap size, the largest weight has a higher probability of error propagation to both cones assuming that the functional and electrical bugs occur with Gaussian distribution. For the same reason, if there are equal size overlaps, the flip-flop driven by the largest logic cone and its neighbor flip-flop are selected. Therefore, the edge between node *C* and *D* is selected from Fig. 3(a), and node *C* and *D* are merged to begin constructing the first MISR (dashed circle in Fig. 3(a)).

The graph is updated after the nodes are merged. Nodes which are merged generate a composite node in a graph. From Fig. 3(b), since the node *C* and *D* are merged, they form a new node and the edge weights are updated. Additional nodes are added to the cluster in a greedy fashion by selecting the edge with the largest weight attached to the current cluster until the size of the cluster reaches *m* which ensures a certain minimum aliasing probability for the MISR. This process is repeated to create the *n* initial clusters.

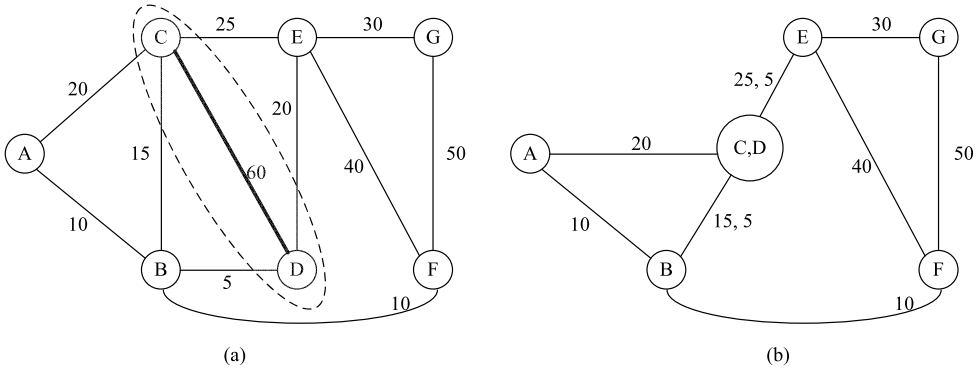

Figure 3. First Cluster Generation

2.2. Merge & Update Procedure

Once the initial clusters have been constructed, a merge & update procedure is iteratively performed to merge the remaining nodes together using the largest weight on an edge at each step. At the conclusion of the procedure, all the nodes will have been added to the initial clusters to form the *n* MISRs.

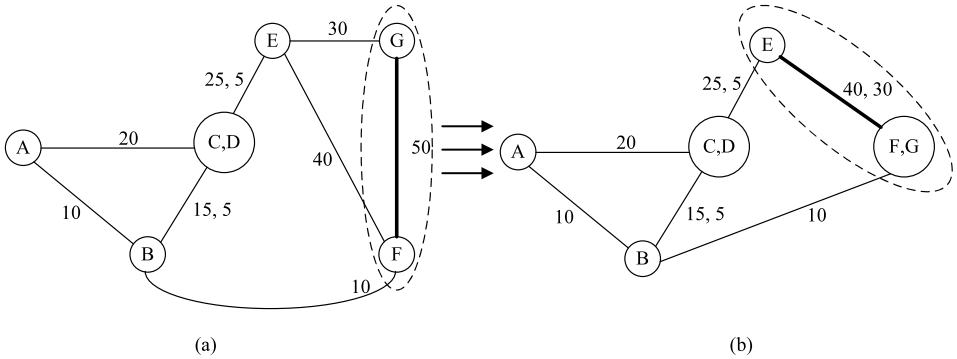

Figure 4. Merge & Update Process

In the merge & update process, since *G* and *F* have the largest weight edge, the second cluster is generated in Fig. 4(a). The iterative process merges node *B* into the second cluster.

In this small example, assume that the number of MISRs is 2, $n = 2$, and the minimum size of a MISR is 3, $m = 3$. The initial clustering procedure generates two clusters each with three nodes, A,C,D and E,F,G, as shown in Fig. 5(b). In Fig. 5(c), the final node *B* is merged with node *A,C,D* since it has more overlap with that cluster, and the two MISRs are finally generated.

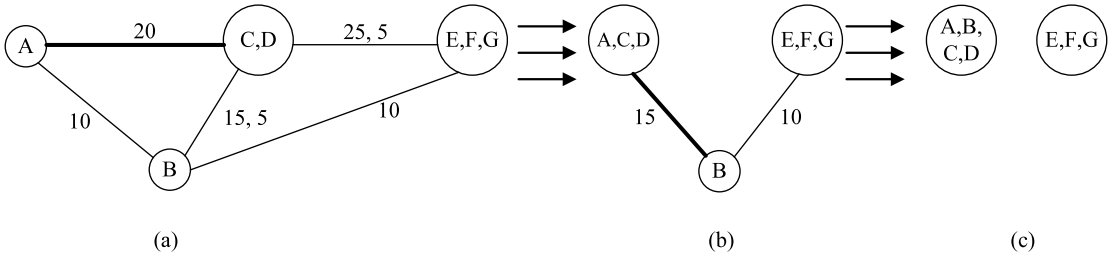

Figure 5. Merge & Update Example ($n = 2$ and $m = 3$)

3. Three Step Debug Process

Scan-based debug methods give high observability of internal signals by shifting out the internal state values. Conventional scan-based debug needs to stop the system to get the information from the CUD. Checking the internal states by scan dumps is a very time consuming task. If a large number of scan dumps is required, this may be an unattractive way of validating a chip. In Sec. 2, a procedure for constructing multiple MISRs is proposed. The MISRs keep compacting the internal state such that erroneous circuit responses will easily corrupt the MISR signatures. Therefore, if the erroneous input comes into the MISRs, although the internal states cannot be read out, the MISR signatures still provide a way to identify the error. In this section, we propose a technique to utilize the internal state information without scanning out the data via scan chains when they have non-destructive capability. A three step debug process is used to zero in on the first erroneous clock cycle. In the first step, single parity information is generated at every clock cycle for periodic monitoring. In the second step, more parity information is stored in a trace buffer to zoom in closer to the failing clock cycle. In the third step, MISR signatures are stored and checked so that the first erroneous cycle can be identified.

3.1. Step One : Checking Intermediate Parity

After configuring the MISR as described in Sec. 2, the MISRs are used to generate the signatures by compacting the outputs of the logic cones driving it. Linear compaction hardware is used to generate a single parity bit for each MISR signature. This can be done by XORing some subset of the flip-flops in the MISR. XORing the parities of the MISRs generates a single parity bit which represents the entire system state. Fig. 6. shows the logic cones with the MISRs and linear compaction circuits (which are simply XOR networks).

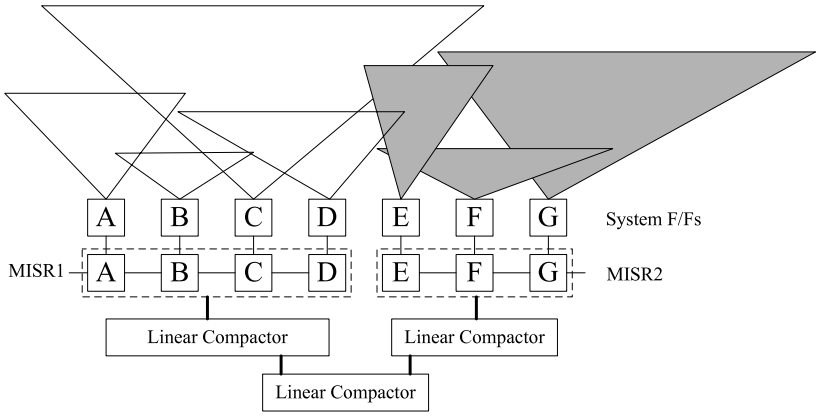

Figure 6. Configured MISRs, DFD and Logic Cones

The composite parity generated by combing the MISR parities is monitored periodically to identify erroneous behavior. It is compared with an externally provided golden parity from off-chip (either checked with an ATE or checked with data stored in some external memory). The periodic parity checking allows at-speed debug with far less volume than the conventional scan-based debug. Conventional scan-based debug requires to stopping the system to perform a scan dump and thus periodic monitoring is not feasible. The monitoring period is determined by the relationship between the system and the automatic test equipment (ATE) operating speed (or the speed of the external memory). Since the system typically runs internally at a higher frequency than the ATE does, the ATE can ideally check a golden parity bit at a period equal to the ratio of the internal chip clock rate to the ATE clock rate, i.e., *frequency of chip / frequency of ATE*. For example, if a chip operates at 3GHz and the ATE runs at 100MHz, the ideal monitoring period is 30 cycles.

Periodic real-time parity monitoring checks the highly compacted circuit response which is represented as a single parity bit and can detect the first erroneous clock cycle within some limited latency which depends on the frequency of the monitoring and the degree of aliasing. The debug session can be stopped when the first mismatching parity is found instead of waiting until the system fails in a functionally observable manner. Since only a single highly compacted bit is being monitored, aliasing is an issue. An even number of error bits will cause parity to alias and hence periodic monitoring would fail to detect a faulty state. However, because the MISRs continue to compact a corrupted signature after the first error, the cumulative aliasing probability exponentially decreases. The probability of 10 consecutive aliases when monitoring the parity is $1/2^{10}$ which is 1 in a thousand.

In addition to monitoring the parity, the proposed approach also involves storing the parity information in a trace buffer at every clock cycle. When the trace buffer gets full, the older data is overwritten so that when the period monitoring halts the debug session, the trace buffer contains the running history of the parity information for the most recent clock cycles. Even though there may be aliasing, the parity history is stored and can be used to find earlier erroneous cycles than the periodic monitoring did. If a 512 Byte trace buffer is used with a 500 cycle monitoring period (one check every 500 clock cycles), then the trace buffer has the history of last 8 monitoring periods. This can be used to more closely zero in on the first erroneous cycle and significantly reduces the debug search space.

3.2. Step Two : Storing Parity of MISRs

Periodic parity monitoring reduces the the range of cycles over which more careful debug is needed. Moreover, since the trace buffer contents provide cycle by cycle information, the first erroneous clock cycle can be approximated with even better clock resolution. This helps to find the neighborhood of the first erroneous clock cycle, however, it may not be able to zero in on the exact clock cycle since the parities stored in the trace buffer also have aliasing issues for even bit errors.

In step two, the parities of the MISRs are used to provide more specific information for identifying the first erroneous clock cycle as well as spatial information on where the error originates. By looking at the parity of each MISR signature and identifying which ones contain errors, information about which logic cones the errors are getting generated in can be deduced.

The periodic monitoring isolates the error to a small range and allows the debug process to be halted at that point. Periodic monitoring is not required in step two. The parities of the MISRs are stored in the trace buffer from a trigger point based on the information obtain in step one about the rough location of the first error. The triggering can be implemented similar to the internal breakpoint mechanism used in [Cabrine 97], [Anis 07a, 07b]. When the debug session stops, the trace buffer will contain each MISR's parity information for the last set of clock cycles. The number of cycles worth of data will be equal to *size of trace buffer/ number of MISRs*.

3.3. Step Three : Storing MISR signatures

After steps one and two, the search space for the first erroneous clock cycle is significantly reduced and some spatial information is available. However, the first erroneous clock cycle is not precisely known due to aliasing uncertainty. It is necessary to shift out MISR signatures and compare them with golden signatures to find the first corrupted signature. Scan dumps are only required in step three and only a small number of MISR signature scan outs are required. The volume of debug data can be reduced in comparison to full scan dumps. For example, if the first MISR signature is corrupted while the second MISR signature is clean in Fig. 6, the second MISR signature does not need to be investigated at the next scan out. A programmable counter can be used and only the MISR signatures of interest need to be compared to reduce the debug effort. In comparison, binary search based debug requires performing a full scan dump at every iteration.

When the first mismatching signature is found, it provides spatial diagnostic information as well because the MISRs are configured using structural information as described in Sec. 2. Multiple MISRs divide the logic cones into multiple regions and aids the debug process. Logic cones that are driving fault-free MISRs can be pruned out to reduce the space of possible root causes.

4. Error Bypassing

One difficult aspect of silicon debug is how to bypass errors in order to continue searching for additional bugs after the first bug is found. In a finite state machine (FSM), the state transition depends on the previous state and the circuit inputs. If a faulty response is found, the downstream state transitions are not guaranteed to follow the expected transitions. This makes downstream debug inefficient. A benefit of using scan chains is the accessibility to internal flip-flops. If the first bug in a design is detected and diagnosed, the downstream debug process needs to be able to continue to find additional bugs in a system. Using the proposed approach to precisely identify the first erroneous clock cycle, it is possible to

determine what the correct state values should be from either simulation or emulation. By utilizing this information, golden values can be shifted in through the scan chains to return the system to the correct state and facilitate downstream debug. Because the system can be rerun with bypassing of the erroneous state, periodic monitoring can now be used to catch other bugs.

5. Experimental Results

The debug method proposed here tries to detect errors early on and thereby reduce the search space and number of debug sessions. Experiments were performed on the larger ISCAS-89 benchmark circuits [Brglez 89] and OpenRisc processor [OR1200]. Random faults were injected in the benchmark circuits and random input patterns were applied. MISRs were configured using the algorithm in Sec. 2 and the parity information was periodically monitored. The results obtained are shown in Table 1. The first and second columns show the circuit name and the number of scan elements. The third column indicates the first clock cycle in which an error becomes functionally observable at a primary output. The next three columns show the first clock cycle that the error is identified using the methods described in Secs. 3.1, 3.2, and 3.3, respectively. The last column shows the number of scan outs required to precisely identify the first erroneous clock cycle. The simulation results show that periodic monitoring detects the erroneous behavior quite close to the first erroneous clock cycle and that using the parities of the MISRs help give more precise information. Hence, very few scan outs are required.

Table 1. Erroneous Response Detection Clock Cycles

Circuit	Scan Size	Error at P.O.	Single Parity Monitor (Monitoring Period)	Parities of MISRs	First Erroneous Clock Cycle	Num. of Scan Out
s9234	211	493	100 (50, 100) 200 (200)	45	43	3
s15850	534	868	100 (50, 100) 200 (200)	7	7	1
s13207	638	828	600 (50, 100, 200)	597	597	1
s38584	1340	2362	1400 (50, 100) 1800 (200)	1338	1338	1
s38417	1636	1680	1250 (50) 1300 (100) 1800 (200)	1221	1221	1
s35932	1728	210	100 (50, 100) 200 (200)	6	5	2
OR1200	1989	1226	900 (50, 100) 1000 (200)	834	834	1
		5712	4050 (50) 4200 (100, 200)	3975	3975	1

Table 2 shows a comparison in terms of the number of debug sessions required using the proposed method and conventional binary search for the same designs and faults as in Table 1. The conventional binary search based debug is initiated when the errors are functionally observable at the primary outputs of the circuit, and it uses scan dumps to check the state of a system at the end of each debug session. However, with the proposed method, the three step

process requires only 3 debug sessions and only a few number of scan dumps. MISR signatures can be stored in a trace buffer without additional debug sessions.

Table 2. Number of Debug Sessions

Circuit / Methodology	s9234	s15850	s13207	s38584	s38417	s35932	OR1200	O1200
Binary Search Debug	16	18	20	22	12	17	18	26
Proposed Debug	3	3	3	3	3	3	3	3

6. Conclusion

In this paper, a new debug technique using a three step process is proposed to zero in the first erroneous clock cycle using a small number of scan dumps. By using conventional scan chains with non-destructive scan out capability, multiple MISRs can be configured to provide high observability with periodic monitoring. Note that the proposed scheme can also be selectively applied to only part of a design, e.g., for newly implemented and unverified design blocks or parts of a design where bugs are more likely to originate from. The scan chains can also used to restore the system state with golden values to facilitate downstream debug.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant No. CCR-0426608.

References

[Abramovici 06] Abramovici, M., P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A Reconfigurable Design-for-Debug Infrastructure for SoCs," *Proc. of Design Automation Conference,* pp. 7-12, 2006.

[Anis 07a] Anis, E., and N. Nicolici, "On Using Lossless Compression of Debug Data in Embedded Logic Analysis," *Proc. of Int. Test Conference,* Paper 18.3, 2007.

[Anis 07b] Anis, E., and N. Nicolici, "Low Cost Debug Architecture using Lossy Compression for Silicon Debug," *Proc. of Design, Automation, and Test in Europe*, pp. 1-6, 2007.

[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.

[Carbine 97] Carbine, A. and Feltham, D., "Pentium®Pro Processor Design for Test And Debug", *Proc. of Int. Test Conference*, pp. 294-303, 1997.

[Gu 06] Gu, X., Wang, W., Li, K., Kim, H. and Chung, S., "Re-Using DFT Logic for Functional and Silicon Debugging Test", *Proc. of Int. Test Conference*, pp. 648-656, 2006.

[Hopkins 06] Hopkins, A., and K. McDonald-Maier, "Debug Support for Complex Systems on-Chip: A Review," *IEEE Proc. on Computers and Digital Techniques*, Vol 153, No. 4, pp. 197-207, Jul. 2006.

[ITRS 05] "The International Technology Roadmap for Semiconductors," Semiconductor Industry Association, 2005.

[Kuppuswamy 04] Kuppuswamy, R., DesRosier, P., Fletham, D., Sheikh, R. and Thadikaran, P., "Full Hold-Scan Systems in Microprocessors : Cost/Benefit Analysis", *Intel Technology Journal*, Volume 08, Issue 01, 2004.

[OR1200] OPENCORES, *http://www.opencores.org*

[Vermeulen 02] Vermeulen, B., Waayers, T. and Goel, S.K., "Core-Based Scan Architecture for Silicon Debug", *Proc. of Int. Test Conference*, pp. 638-647, 2002.

[Yang 08] Yang, J.-S., and Touba, N. A., "Expanding Trace Buffer Observation Window for In-System Silicon Debug through Selective Capture"*, Proc. of IEEE VLSI Test Symposium*, pp. 345-351, 2008.

[Yen 06] Yen, C-C., Lin, T., Lin, H., Yang, K., Liu, T. and Hsu, Y.-C., "Diagnosing Silicon Failures Based on Functional Test Patterns", *International Workshop on Microprocessor Test and Verification*, pp. 94-98. 2006.