

At-Speed Interconnect Test and Diagnosis of External Memories on a System

Heon C. Kim, Hong-Shin Jun, Xinli Gu, Sung S. Chung
Cisco Systems, Inc.
170 W. Tasman Drive
San Jose, CA 95134

ABSTRACT

This paper presents a Built-In Self Test (BIST) implementation for external memories like DDR (Double Data Rate), double DDR, QDR (Quad Data Rate) SRAM, DDR FCRAM (Fast Cycle RAM), and RLDRAM (Reduced Latency DRAM). We utilize the memory controller in the functional block to design the BIST so that the BIST design can be simplified and executed at the functional speed. However, there are many different types of the memory controllers depending on the types of external memories, functional interface protocols, and implementation methodologies. In order to support the various memory controllers, we defined the latency of the memory controllers and classified them into three different categories: fixed latency, handshake, and both fixed latency, and handshake memory controllers. With these three models, we developed a general BIST architecture to support different types of memory controllers. During the Boundary-scan driven BIST operation in the board and the system-level test and diagnosis, system clock, system hard reset, soft reset, and other programmable features were considered carefully to make the BIST operate properly. This paper also presents a unique way of utilizing special BIST functions during the board and system level test, and also during the system mission operation.

1 Introduction

With the increase in network system's performance, more and more large and high performance external memories are used. This increases the number of memories used in the system as well as their interconnections on a board or system. As an example, Table 1 shows that up to 93% percent of the functional IOs are used to connect the external memories and the memory controller ASICs. The Table shows the importance of testing external memories and their interconnections.

	Total Functional IOs	Interconnections with Ext. Mem	%	Types of External Memories
Chip 1	539	68	13%	SRAM
Chip 2	1382	1290	93%	D-DDR, QDR, FCRAM
Chip 3	1398	1047	75%	D-DDR, QDR, FCRAM
Chip 4	1448	1288	89%	DDR, QDR, FCRAM
Chip 5	1470	1154	79%	DDR, QDR, FCRAM
Chip 6	752	75	10%	DDR
Chip 7	611	102	17%	QDR, RLDRAM
Chip 8	804	301	37%	RLDRAM

Table 1: Percentages of Interconnections with External Memories

With the increasing use of memories, there are more possibilities of having board interconnect defects such as cold solder joints, non-homogeneous stripe line width, or thickness. Also, functional defects can be introduced in the memory during the board and system assembly process. Therefore, it is very important to test the external memory interconnections on a board or system at the system speed [5].

IEEE STD1149.1 Boundary-scan or In-Circuit Tester (ICT) is commonly used to perform the interconnect test, but at a very slow speed. Comprehensive interconnect tests between the memory devices and the memory controllers are also limited by these methods due to the limited availability of test points or Boundary-scan in the memory devices. Most often, the memory devices are not fully populated during the board assembly process. Instead, they are populated later during the system test depending upon the configuration and the given system requirement. This hampers comprehensive interconnect test. There are many examples utilizing these techniques to perform memory test [2][3]. However, the quality of the memory test may not be satisfactory in terms of the at-speed functional coverage because the test algorithms used during the ICT or Boundary-scan test are not comprehensive [1].

Embedded memory BIST style BIST techniques were also used within the memory controller ASICs to test external memories [4]. With this approach, it is difficult to achieve the at-speed functional timing requirement between the ASICs and the external memories. This is because the test speed generated by the BIST controller may not be of the same speed at which the actual memory controller accesses the memory, even if the same clock source is used. Furthermore, the memory controller design becomes very complicated if it has to support various types of external memories, such as DDR, QDR, Double DDR SRAM, FCRAM, and RLDRAM. Quite often, a typical single memory controller can support multiple memory types and it can be used in different design environments having different memory requirements.

Therefore, to perform at-speed memory test including interconnect test, we need a different mechanism for the test. Our approach is to define and architect a memory controller functional block within the ASIC, which has physical connections to the external memory. The BIST logic utilizes this memory controller to perform various BIST functions by using its own native memory access functions and speed [6] [7]. The controller provides timing and access to the external memory. The controller is shared by both the system memory access functions and BIST logics. This benefits the memory access in two ways: Firstly, it simplifies the memory access in terms of timing and control; Secondly, it also reduces the redundant circuits commonly practiced by the traditional BIST implementations where all the memory access and timing functions are duplicated for the purpose of BIST. The memory BIST can test both the external memories and the interconnections. Also, these tests are performed at-speed because both the memory controller logic and the memory BIST use the system clock. The timing between the ASICs and the external memories are handled by the memory controllers.

There is no standard way of implementing memory controllers. The memory controllers are designed by functional design teams. Their implementation can vary depending on the type of the memory, the functional requirements, and the implementation methodology even for a given memory type. To support various types of memory controllers, we measured the latencies for commands and data, and classified them into three different types: a fixed latency, non-fixed latency (or handshaking type), and mixed type. Different types of latency handling logics are implemented depending on the latencies used in the design.

The external memory BIST can be used during the board and system manufacturing test as well as during the mission operation. To support board and system level test, special provisions must be given to resolve the board-level

clock and reset generation, soft reset sequence, diagnostic interface, and system access features. In order to use the BIST functions in the mission mode, the BIST has special interfaces to the mission logic and special functions that are required by the design engineers.

In this paper, we define a memory BIST controller to test the external memory. We also present different BIST design examples for different types of memory controllers. Finally, different BIST utilization methods are presented for board and system level test and for the mission operation.

2 Memory Controllers

High performance memories in a system may have clocks at different speeds, different data rates, different burst lengths, multiple operational modes, and serpentine routings on a board. They may also have different memory organizations on a board; single memory, depth-expansion for deeper memory, and width-expansion for wider memory.

Memory technologies continue to evolve, and the BIST should accommodate all the different types of memory controllers. However, it will be extremely difficult to design and maintain different versions of BIST. Therefore, it is necessary to simplify and classify the various BIST functions with reasonable constraints. In order to achieve these goals, we defined a new memory controller model for the BIST design.

Figure 1 shows an example of one possible external memory BIST design. The memory BIST is placed outside of the functional logic and a multiplexer is used to select the memory access paths.

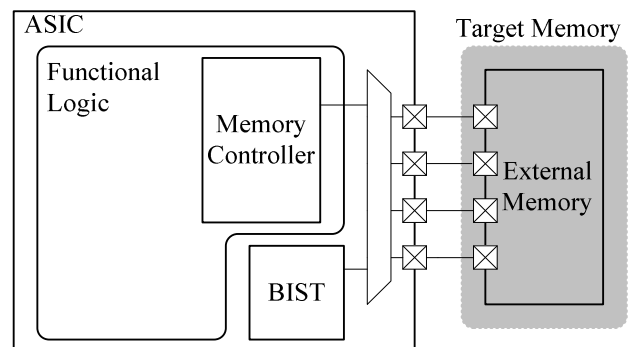


Figure 1: An Example of BIST for External Memory

In this case, the BIST has to consider all the complex functions and the timing of the external memory. But, this results in duplicate logic between the BIST and the memory controller. In general, an external memory has both operational clock and echo clock, and delays between them. BIST has to consider the different clocks and delays. If the external memory has DDR or QDR

operation, the BIST should use both the clock edges to generate write data and capture read data. If the external memory has burst operation, the BIST should write and read data continuously according to the burst length. Inserting multiplexers on the timing critical functional paths near pads will lead to timing penalties.

Figure 2 shows our approach to design external memory BIST. The memory controller used can be a functional block or a group of functional blocks to interface with the external memory.

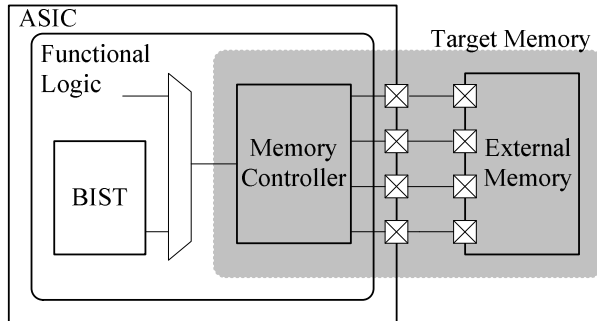


Figure 2: Our Approach of BIST for External Memory

The memory controller performs timing and data conversion for the external memory access operations. By using the memory controller, we can achieve simplified accesses to external memory, which uses the system clock, system reset, and control signals synchronized with the system clock. The memory controller can also transform DDR or QDR operation to a single data rate operation and burst operation to a non-burst operation with wider data width. As a result, an external memory test operation performed through the functional interface can be regarded as an embedded memory interface during the BIST operation.

3 BIST Designs

There is no standard on how to implement a memory controller. Different memory types may require different timing, address mechanism, refresh cycles, initialization sequences, burst operation, data latency, and so on. Depending on the functional requirements, the memory controller may have different interfaces, timing, and operations. Depending on the implementation methodology, the memory controller may have different initialization sequence, programmability, pipeline depth, and so on.

BIST implementation may be different for different types of memory controllers. To make the BIST design simpler, we should develop a general interface model of the BIST and classify the various memory controller functional blocks to obtain reasonable flexibility and scalability.

Figure 3 shows a block diagram of our BIST implementation. The BIST core is a general purpose BIST engine. The Protocol Handler block communicates with the memory controller and it may have different functions depending on the memory controller used. All memory controller protocols will be considered in the Protocol Handler so that the BIST core can have a consistent architecture regardless of the different memory controllers used in a design. Using this architecture enables us to support different variations of external memory BIST designs for different memory controllers with minimum design overhead.

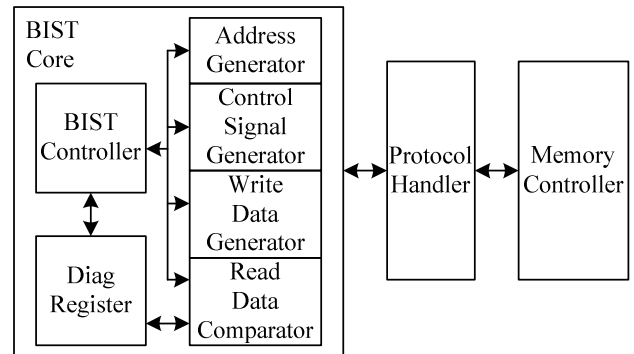


Figure 3: Block Diagram of BIST for External Memory

We define latency as the number of clock cycles between commands or between data, and it is implemented within the memory controller. We measure command latency, write data latency, and read data latency for a given memory controller. Based on the latencies, we classified the memory controller into three different types: memory controller with fixed latency, memory controller with non-fixed latency (or with handshake), and memory controller with both fixed latency and handshake. Each one of them is discussed in the following sections.

3.1 Memory Controller with fixed latency

In many cases, the memory controllers have a fixed latency as in SRAM controllers where all the operations to and from memory have the transactions in fixed clock cycles. In this case, the BIST for external memory design is similar to the embedded memory BIST design with the retiming logic.

Figure 4 shows an example of a timing diagram with fixed latencies where the command latency is 0, the write latency is 1, and the read latency is 6. The BIST implementation for this type of memory controller needs extra retiming logic in the Protocol Handler as shown in Figure 5.

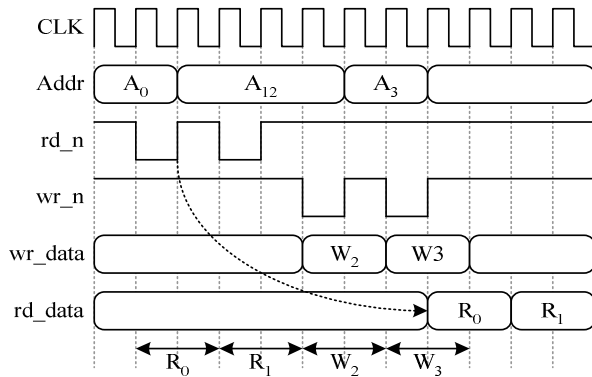


Figure 4: Timing Example of QDR SRAM Controller with Fixed Latency Protocol

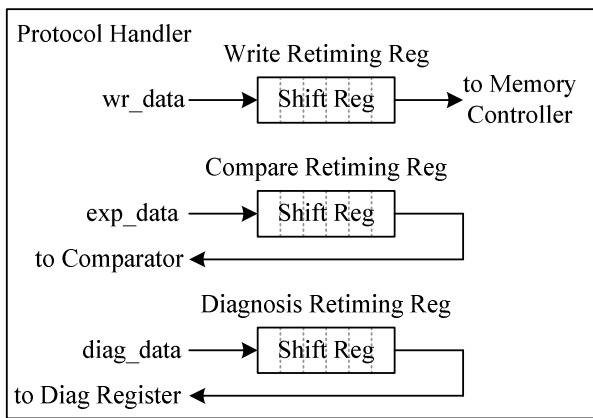


Figure 5: Protocol Handler for Memory Controller with Fixed Latency

A retiming register is not required if the commands have a fixed latency. The length of the retiming shift registers depend on the latency of the write and read data. In the timing example in Figure 5, the length of Write Retiming Register is 1 and all the other retiming registers are 6 bits long. One clock cycle after the write command was captured the memory controller captures the write data from BIST (called as **wr_data**). Six clock cycles after the read command was captured, the expected data from the BIST (called as **exp_data**) will be compared with the read data from the memory controller, and the diagnostic information (called as **diag_data**) will be captured at the same time in accordance with the read result.

3.2 Memory Controller with handshake

In many cases, memory controllers like the DRAM controller works interactively with the functional logic through a series of handshake method. Based on the given implementation of a memory controller and its functional interface, there are minor differences in the number of handshakes used, different mode of operations, different

handshake signals, and timing. However, if a controller uses handshake method to interface, it can be classified into this category.

As an example shown in Figure 6, the latency for the write and the read operation is not fixed and it can vary according to the readiness of memory controller and the read data. There are three most commonly used handshake signals from the memory controllers; **ack**, **d_ready**, and **d_val**. When the memory controller is in the available state, it accepts a command and toggles the **ack** signal. The command is now queued into the memory controller for execution. The **d_ready** signal indicates the timing to drive write data and the **d_val** signal indicates the timing for the valid read data. Depending on status of the **d_ready** and **d_val** signal, the BIST can drive write data or compare read data accordingly.

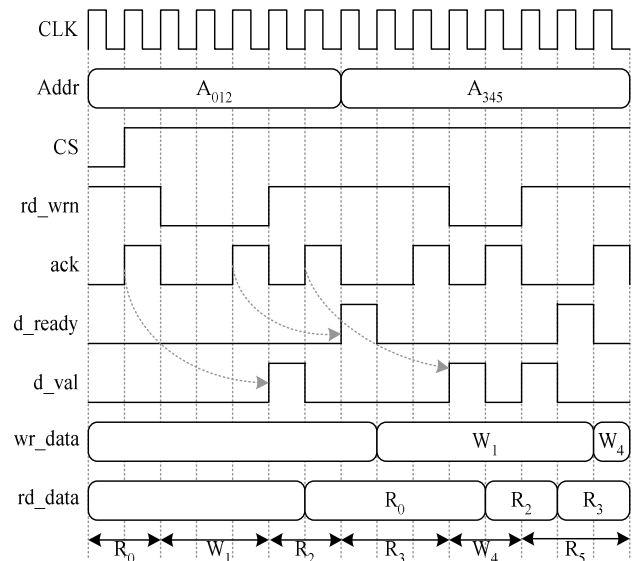


Figure 6: Timing Example of RLDRAM Controller with Handshake Protocol

For the purposes of implementing proper BIST for this type of memory controller, time-out and sync circuits are used to keep track of undesirable failures caused by the memory handshake circuit. An example implementation of FIFOs circuit in a Protocol Handler is shown in Figure 7.

The Time-Out Counter detects handshake signal failures and it plays the most important part to make handshake work properly. If there is no time-out counter and the handshake signal never toggles due to an undesirable failure condition, the BIST can be in an infinite wait loop. The Time-Out Counter prevents the BIST from waiting in the infinite loop by forcing the BIST controller out of this state.

The Write Sync FIFO is used to queue the write data, **wr_data**. The Compare Sync FIFO is used to queue the

expected data, **exp_data**, and it will be used to compare with the read data from memory controller. The Diag Sync FIFO is used to queue the diagnostic information, **diag_data**.

The FIFO control depends on the memory controller signals. As shown in Figure 6 example timing, write enable signal and command acknowledge signal (**~rd_wrn & ack**) are used to push the data into the Write Sync FIFO. The data ready signal, **d_ready**, is used to pop the FIFO. Both the Read enable signal and the command acknowledge signal (**rd_wrn & ack**) are used to push data into both the Compare Sync FIFO and Diag Sync FIFO, and the data valid signal **d_val** is used to pop both the FIFOs.

Each FIFO can have check logic to detect FIFO overflow or underflow conditions.

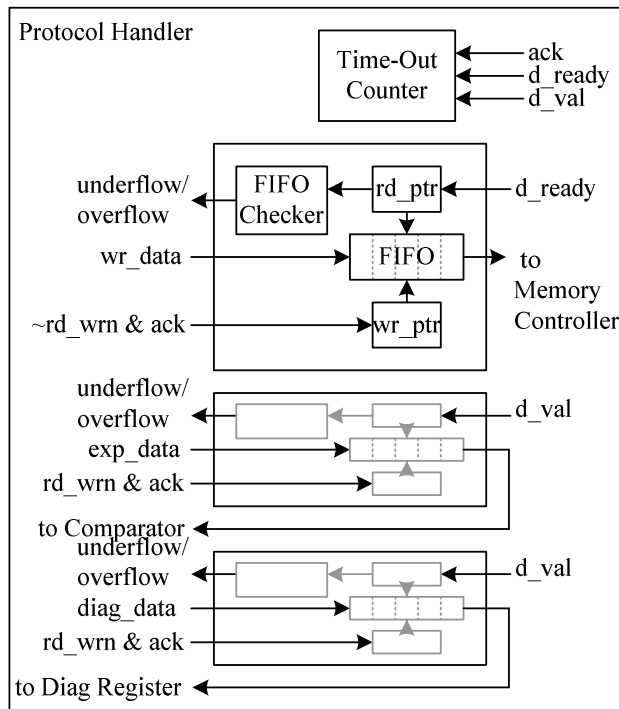


Figure 7: Protocol Handler for a Memory Controller with Handshake

3.3 Memory Controller with both fixed latency and handshake

Some memory controllers have both fixed latency for some signals and handshake mode of operation for the other signals. For example, the command and write data will have a fixed latency, but the read data operates in the handshake method. The BIST implementations for this type of memory controllers have retiming logic and contain FIFOs in the Protocol Handler.

4 Board and System-Level Usage of the BIST

To support various board and system level usages for the external memory BIST, the BIST has two different interfaces; Boundary-scan interface and functional interface. The Boundary-scan interface is mainly used by board and system test engineers during the manufacturing test. The functional interface is designed with requests from the ASIC designer or system designer and it will be used during the mission mode or as a part of diagnosis and debug function.

4.1 Board and System Level Test

During the board or system manufacturing test, the test engineer can control the BIST functions using Boundary-scan access. Once the BIST completes the given operation, the BIST execution results can also be accessed through Boundary-scan.

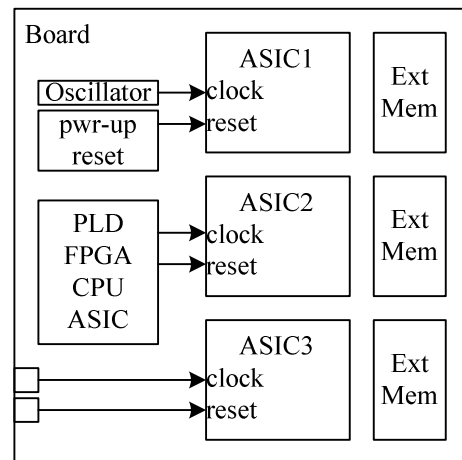


Figure 8: Generation of System Clocks and Resets

During the board level Boundary-scan test, the system clock and the appropriate system reset signal should be exercised properly for an ASIC, so that the external memory BIST in the device can have a proper known state. It is important because the memory controller has to receive system clocks and proper initialization signals through reset in exactly the same way as the system operates. If Boundary-scan blocks or over looks the key initialization signals and sequences for the memory controller, the BIST operation through Boundary-scan may not work properly. Figure 8 shows different configurations for the system clock and reset signal generation circuit.

If the system clock or reset is generated from the board as shown in "ASIC1" in Figure 8, no special consideration is required for the Boundary-scan test. After the normal power-up sequence, clock and reset will have proper operations to initialize the appropriate logic for the BIST.

If the system clock is generated from another device such as PLD, FPGA, CPU, or from a different ASIC as shown in “ASIC2” in Figure 8, the appropriate BIST functions should be available after the power-up sequence. Otherwise, the DFT engineer should work with the ASIC, board, or system engineer to have a free-running system clock and proper reset signal for the BIST and the memory controller.

If the system reset is generated from the other device as shown in “ASIC2” in Figure 8, the test engineer can control reset signal through Boundary-scan using the EXTEST instruction for the device.

If system clock or reset is driven from outside the board during test as shown in “ASIC3” in Figure 8, the test engineer should generate the signals from the Boundary-scan tester.

To ensure proper operation of the Boundary-scan driven BIST, the DFT engineer needs to pay close attention to the soft reset (warm reset) and the software programmable features of the device.

During the Boundary-scan driven test, it may not be possible to control the soft reset signal that is controlled by CPU or diagnostic software. The system reset signal which may be controlled by the soft reset and the hard reset might be asserted even during Boundary-scan. During BIST operation, if the soft reset is asserted it should be de-asserted somehow because the external memory BIST uses the system clock and system reset to perform at-speed test.

Some ASICs may have programmable features for the PLL and the memory controller, so that their software can change the PLL setting or the memory configuration. The programmability should be controllable during the boundary-scan driven BIST test in order to achieve at-speed operation. This can be achieved by adding control registers to drive the signals from Boundary-scan. An example of Boundary-scan control sequence to support programmable PLL feature is given below:

- ASIC Hard reset / Boundary-scan reset
- Load Boundary-scan instruction for the BIST
- Load PLL control data through TDI
- Turn-off PLL reset through TDI
- Turn-off soft reset through TDI
- Set BIST control registers and enable the BIST through TDI
- Wait for the BIST to complete operation
- Capture BIST result and shift-out to TDO

If the external memory and the memory controller require special initialization sequence then, it should be done automatically when the device is going through the power-up sequence. Otherwise, the external memory and

the memory controller should have extra controllability for the reset prior to Boundary-scan driven BIST operation.

4.2 Functional Test and Usage

Many systems require either an external memory test or memory initialization during system power-up sequences or before the system enters into the full functional mode of operation. This may take a long time if the systems use diagnosis software or CPU access to test or initialize external memories. Our BIST is a dedicated hardware to access the external memory so that it can perform external memory test or initialization extremely fast and effectively even without CPU intervention. It reduces the time to power-up the system substantially, reduces the warm start time, and improves the overall system up time substantially.

The BIST can be used in the system power-up sequence as well as for memory initialization while the system is running. The BIST can also be used for at-speed test and diagnosis. To support these system functions with the use of BIST, the BIST has additional modes and CPU interface signals that are required by the ASIC, board, or system designers:

- Initialization mode
- Initialization pattern register
- Programmable address range
- Diagnosis and Trace Register

When the BIST is enabled during initialization mode from the CPU controls, the BIST will write the pattern to the initialization pattern register for the specified address range of the external memory, and then the BIST will perform the operation accordingly.

When a BIST is enabled during the test mode from the CPU controls, the BIST will perform its programmed or built-in test operations and generate test results. The results can be accessed by CPU for diagnosis purposes.

5 Conclusion

This paper presented the implementation of high performance external memory BIST. It proposed to use the functional memory controllers to access the external memories for the BIST operation. The BIST can test external memories, memory controller, and their interconnections at functional speed. This paper also presented different types of memory controllers and their corresponding BIST implementation methods.

Several BIST designs for both programmable BIST and BIST with fixed algorithm were implemented. All of them consisted of the BIST controller and the protocol handler and supported various types of memories and controllers.

The BIST can be used for both the board and the system test, and also during the functional operation to improve the system up-time. This paper also presented key BIST interface solutions to interface BIST with various external memory test and initialization conditions including system level test during mission mode of operation.

References

- [1] IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE std. 1149.1-1990, IEEE Computer Society.
- [2] Benoit Nadeau-Dostie, Jean-Francois Cote, Harry Hulvershorn, and Stephen Pateras, "An Embedded Technique For At-Speed Interconnect Testing," *Proc. International Test Conference*, 1999, pp. 431-438.
- [3] Jongchul Shin, Hyunjin Kim, and Sungho Kang, "At-Speed Boundary-Scan Interconnect Testing," *Proc. Design Automation and Test in Europe (DATE)*, 1999, pp. 473-477.
- [4] Chen-Huan Chiang, "BIST TPG for SRAM Cluster Interconnect Testing at Board Level," *Proc. Asian Test Symposium*, 2000, pp. 58-65.
- [5] Benoit Nadeau-Dostie, "Design for at-speed Test, Diagnosis and Measurement," Kluwer Academic Publishers, Boston, U.S.A.
- [6] Xinli Gu, Weili Wang, Kevin Li, Heon Kim, and Sung Chung, "Re-Using DFT Logic for Functional and Silicon Debugging Test," *Proc. of International Test Conference* 2002, pp. 648-656.
- [7] Oliver Caty, Ismet Bayraktaroglu, Amitava Majumdar, Richard Lee, John Bell, and Lisa Curhan, "Instruction Based BIST for Board/System Level Test of External Memories and Interconnects," *Proc. of International Test Conference* 2003, pp. 140-149.