

# JEDEC STANDARD

---

## Fully Buffered DIMM Design for Test, Design for Validation (DFx)

---

### JESD82-28A.01

(Editorial revision of JESD82-28A, July 2008)

SPECIAL DISCLAIMER: JEDEC has received information that certain patents or patent applications may be relevant to this standard, and, as of the publication date of this standard, no statements regarding an assurance or refusal to license such patents or patent applications have been provided.

<http://www.jedec.org/download/search/FBDIMM/Patents.xls>

JEDEC does not make any determination as to the validity or relevancy of such patents or patent applications. Prospective users of the standard should act accordingly.

March 2023

---

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION



## NOTICE

JEDEC standards and publications contain material that has been prepared, reviewed, and approved through the JEDEC Board of Directors level and subsequently reviewed and approved by the JEDEC legal counsel.

JEDEC standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for use by those other than JEDEC members, whether the standard is to be used either domestically or internationally.

JEDEC standards and publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC standards or publications.

The information included in JEDEC standards and publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC standard or publication may be further processed and ultimately become an ANSI standard.

No claims to be in conformance with this standard may be made unless all requirements stated in the standard are met.

Inquiries, comments, and suggestions relative to the content of this JEDEC standard or publication should be addressed to JEDEC at the address below, or refer to [www.jedec.org](http://www.jedec.org) under Standards and Documents for alternative contact information.

Published by  
©JEDEC Solid State Technology Association 2023  
3103 North 10th Street  
Suite 240 South  
Arlington, VA 22201-2107

JEDEC retains the copyright on this material. By downloading this file the individual agrees not to charge for or resell the resulting material.

**PRICE: Contact JEDEC**

Printed in the U.S.A.  
All rights reserved

DO NOT VIOLATE  
THE  
LAW!

This document is copyrighted by JEDEC and may not be  
reproduced without permission.

Organizations may obtain permission to reproduce a limited number of copies  
through entering into a license agreement. For information, contact:

JEDEC Solid State Technology Association  
3103 North 10th Street  
Suite 240 South  
Arlington, VA 22201-2107  
<https://www.jedec.org/contact>

# Special Disclaimer

**JEDEC has received information that certain patents or patent applications may be relevant to this standard, and, as of the publication date of this standard, no statements regarding an assurance or refusal to license such patents or patent applications have been provided.**

**<http://www.jedec.org/download/search/FBDIMM/Patents.xls>**

**JEDEC does not make any determination as to the validity or relevancy of such patents or patent applications. Prospective users of the standard should act accordingly.**

# Fully Buffered DIMM Design for Test, Design for Validation (DFx)

## Contents

	Page
<b>1 Scope.....</b>	<b>1</b>
<b>2 Related Documents.....</b>	<b>1</b>
<b>3 Terms, Definitions, and Abbreviations .....</b>	<b>1</b>
<b>4 Test and Validation Overview .....</b>	<b>4</b>
4.1 FBDIMM Overview .....	4
4.2 FBDIMM DFx Objectives.....	5
4.3 DIMM Test Process.....	5
4.4 Memory Manufacturing Flow.....	6
4.5 Reducing Test Cost.....	6
4.6 Test Strategy .....	7
4.7 AMB Test.....	8
4.7.1 AMB Functionality and Defects .....	8
4.7.2 DDR Functionality and Timing .....	8
4.7.3 FBDIMM Functionality and Timing.....	9
4.7.4 Connection of FBDIMM Pins to the Tester .....	9
4.7.5 FBDIMM and DDR DC Tests .....	10
4.7.6 Testing DRAM Access Features .....	10
4.8 DIMM Test.....	10
4.8.1 Assembly Defects .....	10
4.8.2 Leakage Testing.....	10
4.8.3 DRAM Array Test.....	11
4.8.4 Programming the SPD .....	11
4.8.5 FBDIMM IO Test .....	11
4.8.6 DDR Interface Testing.....	11
4.9 System Test .....	11
4.10 Validation.....	12
<b>5 Requirement Summary.....</b>	<b>12</b>
5.1 DFx Requirements .....	12
5.2 Required Features .....	12
5.3 DFx Features and Usage .....	13
5.4 AMB Overview .....	14
5.4.1 Memory BIST .....	14
5.4.2 Interconnect BIST .....	15
5.5 DDR Interface Tester Compatibility .....	15
5.5.1 Logic Analyzer Interface.....	16
5.5.2 DC Testing .....	17
5.6 Host Controller Requirements.....	18
<b>6 AMB Component DFx .....</b>	<b>18</b>
6.1 Normal Mode Debug/Validation Hooks.....	18
6.2 Error Injection and In-Band Events .....	18
6.2.1 Decode of Southbound In-Band Debug Events.....	18
6.2.2 Pattern Match on a Command .....	18
6.2.3 Programmed Event Delay .....	18

---

**Contents (cont'd)**


---

6.2.4	Error Injection .....	18
6.2.5	Sourcing Northbound In-Band Event .....	19
6.3	Logic Analyzer Interface Mode .....	19
6.3.1	Link Protocol Validation/Debug .....	20
6.3.2	LAI Debug Examples .....	20
6.3.3	System Level Debug .....	21
6.3.4	LAI Mode Architecture .....	21
6.4	LAI Requirements .....	31
6.4.1	Capture all Southbound and Northbound Traffic .....	31
6.4.2	De-multiplex Captured FBDIMM Channel Traffic .....	31
6.4.3	Drive Link Traffic with Framing Signal to Logic Analyzer .....	31
6.4.4	Detection of Debug Events .....	31
6.4.5	Event Response Mechanisms .....	33
<b>7</b>	<b>Common Dfx Features .....</b>	<b>34</b>
7.1	FBDIMM IO Test .....	34
7.2	Interconnect BIST .....	34
7.2.1	FBDIMM IBIST Architecture Specification .....	34
7.2.2	Reference Architecture .....	43
7.2.3	Pattern Generation .....	47
7.2.4	Transmitter Block .....	53
7.2.5	Receiver Block .....	54
7.2.6	Error Detection .....	54
7.2.7	Validation and Test Usage Models .....	54
<b>8</b>	<b>DIMM Test and Manufacturing .....</b>	<b>58</b>
8.1	Transparent Mode .....	58
8.1.1	Transparent Mode Architecture .....	60
8.1.2	Clock Frequency and Core Timing .....	60
8.1.3	Edge Placement Accuracy .....	61
8.1.4	Transparent Mode Timing .....	61
8.1.5	Error Reporting .....	65
8.1.6	Transparent Mode IO Specifications .....	68
8.1.7	IO Implementation Guidelines .....	68
8.2	Memory BIST .....	70
8.2.1	System Level Test .....	70
8.2.2	DIMM Manufacturing .....	71
8.2.3	DDR Interface Testing .....	71
8.2.4	MemBIST Overview .....	72
8.2.5	Algorithmic Testing .....	77
8.2.6	DRAM Operations not Supported .....	79
8.2.7	Quad Rank Support .....	79
8.2.8	MemBIST Flow Control FSM .....	79
8.2.9	MemBIST CSFSM .....	82
8.2.10	MemBIST Feature Summary .....	83
8.2.11	MemBIST Registers .....	84
8.2.12	MemBIST Timing Control .....	88

---

**Contents (cont'd)**


---

<b>9</b>	<b>System Test .....</b>	<b>93</b>
9.1	Overview .....	93
9.2	Voltage Margining .....	93
9.3	Timing Margining.....	93
9.4	Voltage, Timing Margin Support Indication .....	94
9.5	Margin Test Usage.....	94
9.6	Register Definitions .....	94
<b>Annex A</b>	<b>(Informative) Differences between Revisions .....</b>	<b>95</b>
A.1	Differences between JESD82-28A.01 and JESD82-28A .....	95
A.2	Differences between JESD82-28A and JESD82-28.....	95

**Figures**

Figure 4-1	Memory Channel Architecture .....	4
Figure 4-2	Memory Assembly and Test Process .....	6
Figure 4-3	200 MHz Tester Driving 800 MHz DQ .....	8
Figure 5-1	AMB Architecture and DFT Blocks .....	14
Figure 5-2	AMB IO Loopback Methods .....	15
Figure 5-3	Logic Analyzer Interface Card .....	17
Figure 6-1	AMB LAI Mode Signals .....	20
Figure 6-2	AMB LAI Mode Architecture .....	22
Figure 6-3	Match and Mask Logic .....	25
Figure 6-4	LAI Qualification Signal Block Diagram .....	26
Figure 6-5	Event Signaling Timing Characteristics .....	29
Figure 6-6	Event Bus Topology .....	29
Figure 6-7	Probing the AMB-LAI Signals .....	30
Figure 7-1	Example of a Common Platform Topology for FBDIMM .....	35
Figure 7-2	IBIST Loopback Operation on a Selected AMB.....	35
Figure 7-3	IBIST Controller-to- Controller Mode Operation on a Selected AMB .....	36
Figure 7-4	FBDIMM IBIST Pattern Definition .....	43
Figure 7-5	AMB FBDIMM IBIST Architecture .....	44
Figure 7-6	AMB IBIST Instantiation Diagram .....	45
Figure 7-7	Pattern Generator Block Diagram.....	48
Figure 7-8	Auto-Inversion Sweep Example.....	49
Figure 7-9	Pattern Set 2 with Auto-Inversion Example .....	51
Figure 7-10	Standard IBIST Loopback Testing.....	55
Figure 7-11	Controller-to- Controller Board Test Diagram.....	56
Figure 7-12	AMB External Wire Trace for Loopback Testing .....	57
Figure 8-1	DRAM Architecture .....	58
Figure 8-2	Transparent Mode Simplified Block Diagram .....	60
Figure 8-3	Transparent Mode Timing.....	62
Figure 8-4	Transparent Mode Write Timing .....	63
Figure 8-5	Transparent Mode Read Timing .....	64
Figure 8-6	BL=8 Read Timing .....	65
Figure 8-7	Mapping of Burst Position Bits to Error Capture .....	66
Figure 8-8	Memory Address Definition, BL=4 .....	73

**Contents (cont'd)**

Figure 8-9	Memory Address Definition, BL=8 .....	73
Figure 8-10	Failure Address Format .....	76
Figure 8-11	MemBIST Flow Control State Machine .....	81
Figure 8-12	MemBIST Command State Machine .....	82
Figure 8-13	DRAM Write Timings (WL=2, AL=0).....	88
Figure 8-14	DRAM Read Timings (CL=3, AL=0) .....	89
Figure 8-15	READ - WRITE - READ Timings (CL=4, AL=0, WL=3) .....	89
Figure 8-16	Relation Between DRAM Timing Control Parameters and MemBIST Functionality .....	90
Figure 8-17	Schematic Representation of the MemBIST Read/Write Functionality.....	91
Figure 8-18	Address Initialization and Sequencing During MemBIST Operation .....	92

**Tables**

Table 2-1	Related Documents.....	1
Table 3-1	Terms and Definitions .....	2
Table 4-1	Test Overview .....	7
Table 5-1	Required DFT Features .....	13
Table 6-1	AMB SMBus Addressing for Logic Analyzer Mode.....	22
Table 6-2	LAI Mode Signal and Pin Count.....	24
Table 6-3	Local Event Pool .....	27
Table 6-4	LAI Local Events .....	27
Table 6-5	LAI Event Selection .....	28
Table 6-6	Different Event Signal Types .....	28
Table 6-7	LAI Signal Definitions3 .....	30
Table 6-8	Setting bit 39 in Match/Mask Registers to Qualify Cmd/Data Match .....	32
Table 7-1	Summary of IBIST Features .....	37
Table 7-2	AMB IBIST Modes .....	39
Table 7-3	Start and End Delimiter Description .....	42
Table 7-4	Patterns Generation Examples .....	52
Table 7-5	Transmitter Block Output Select .....	53
Table 8-1	Transparent Mode Pin List.....	62
Table 8-2	Selection of 8 Bit Data Paths When ENDOUT is Set .....	67
Table 8-3	Transparent Mode FBDIMM Interface Signaling Specifications .....	68
Table 8-4	Transparent Mode Mapping .....	69
Table 8-5	Transparent Mode Mapping of XORA2 and XORA6 .....	70
Table 8-6	Address Inversion .....	74
Table 8-7	Refresh Programming.....	77
Table 8-8	MemBIST Features .....	83
Table 8-9	Memory Technology Settings .....	84
Table 8-10	DRAM Timing Values .....	84
Table 8-11	Memory Data Register .....	86



## Fully Buffered DIMM Design for Test, Design for Validation (DFx)

From JEDEC Board ballot, JCB-06-68 and JCB-08-30, formulated under the cognizance of the JC-40.4 Subcommittee for Digital Logic Liaison and the JC-40.5 Subcommittee on Logic Verification and Validation.

---

### 1 Scope

---

This FBDIMM DFx standard covers Design for Test, Design for Manufacturing, and Design for Validation (“DFx”) requirements and implementation guidelines for Fully Buffered DIMM technology.

---

### 2 Related Documents

---

**Table 2-1 — Related Documents**

Document	Revision	Description
FBDIMM Architecture and Protocol Specification	1.0	FBDIMM architecture and interface protocol
FBDIMM Connector Specification	1.0	Connector physical parameters: pinout, footprint, mechanical drawing, and requirements
FBDIMM Signaling Specification	1.0	PTP link parameters: signaling, I/O, and AC and DC parameters
FBDIMM AMB Specification	1.0	AMB Characteristics: pinout, package type, mechanical outline, footprint, AC/DC specs, power/thermal requirements, buffer TPT, special feature requirements (e.g., thermal sensor), & basics DFT
FBDIMM DIMM Specification	1.0	DIMM module parameters: multiple raw card designs, block diagrams, net topologies, routing details, timing budget, pin out, mech. Outline, stack up, and SPD requirements
SMBus	2.0	System Management Bus Specification <a href="http://smbus.org/specs/smbus20.pdf">http://smbus.org/specs/smbus20.pdf</a>
DDR2 JEDEC Spec		JEDEC DDR 2 SDRAM Data Sheet JC-42.3

---

### 3 Terms, Definitions, and Abbreviations

---

This standard uses the following terms, definitions, and abbreviations:

**NOTE** The terms chipset and memory controller are used interchangeably throughout the rest of this document. The term motherboard is used as a generic term to describe the PCB onto which the memory controller is mounted. Actual implementations could have distributed memory controllers mounted on separate memory boards.

### 3 Terms, Definitions, and Abbreviations (cont'd)

**Table 3-1 — Terms, Definitions, and Abbreviations**

<b>TERM</b>	<b>Definition</b>
AMB	Advanced Memory Buffer
ATE	Automatic Test Equipment
Chip disable	An ECC encoding specifically tailored for memory such that the data from any defective memory device can be reconstructed from some aggregate of surviving memory devices. Corrects data from failed device.
Bit Lane	A differential pair of signals in one direction.
D+ and D-	The D+ and D- terms used in this document are used to indicate the two conductors or signals of a differential signaling pair.
DDR	Double Data Rate (SDRAM)
DDR Channel	A DDR channel consists of a data channel with 72 bits of data and an addr/ control channel
DFx	Design-for-(test, validation, debug etc.). The collection of design features for silicon test, validation and other uses.
DIMM	Dual In-Line Memory Module. A packaging arrangement of memory devices on a socketable substrate.
DIMM Slot	Receptacle (socket) for a DIMM. Also, the relative physical location of a specific DIMM on a DDR Channel.
DIMM Stack	Dual-ranked x4 DRAM DIMM physical topology: refers to two physical rows of DRAM “stacked” one above another
DRAM Page (Row)	The DRAM cells selected by the Row Address
DPM	Defects per million
DRAM	Dynamic Random Access Memory
ECC	Error Correction Code.
EMI	Electro-magnetic interference
FBDIMM	Fully Buffered DIMM
Frame	Group of bits containing commands or data
HMI	Host-memory interface. The connection between a host controller and memory.
Host	Memory controller agent on an FBDIMM channel
ISI	Inter Symbol Interference
JEDEC	JEDEC Solid State Technology Association (once known as the Joint Electron Device Engineering Council)

### 3 Terms, Definitions, and Abbreviations (cont'd)

**Table 3-1 — Terms, Definitions, and Abbreviations (cont'd)**

JESD79	JEDEC Standard 79, DDR SDRAM Specification
Link	A dual-simplex communications path between two components. The collection of two Ports and their interconnecting bit lanes.
NB	Northbound
Northbound	The direction of signals running from the farthest DIMM toward the host.
PCB	Printed circuit board
PLL	Phase Locked Loop
Port	In physical terms, a group of transmitters and receivers physically located on the same chip that define one end of a Link.
PVT	Process, Voltage and Temperature
Rank	A DIMM is organized as one or two physical sets of memory, called ranks. Note that single rank or dual rank is different from single-sided or double-sided, e.g., a single rank DIMM build from x4 DRAM devices is actually double-sided. It is also common practice to distribute the 9 devices of an x8 DIMM between both sides of the DIMM to enhance the thermal performance of the module.
Resample	A resampler repeater is a serial data in and serial data out node that attenuates jitter by re-generating the serial data using a clock recovered from the incoming data stream derived from a common reference clock. It also resets the voltage budget of the re-transmitted data.
Resync	A resync repeater is a serial data in and serial data out node that re-synchronizes data to a local clock after it has been sampled with a recovered clock derived from a common reference clock. The local clock is also generated from the same reference clock by a PLL multiplier. A drift compensation buffer is inserted between the two clock domains which absorbs the maximum link delay change over worst case voltage and temperature changes. Both the jitter and voltage budgets for the re-transmitted data are reset.
SB	Southbound
SDRAM	Synchronous Dynamic Random Access Memory
SI	Signal Integrity
Serial Presence Detect	An EEPROM device on the DIMM, used to store DIMM and DRAM parameters
SMBus	System Management Bus. Controlled by a system management controller to read and write configuration registers. Limited to 100 KHz.
Southbound	The direction of signals running from the host controller toward the DIMMs.
SSTL_18	Series Stub Terminated Logic for 1.8 V
Unit Interval	Average time interval between voltage transitions of a signal

### 3 Terms, Definitions, and Abbreviations (cont'd)

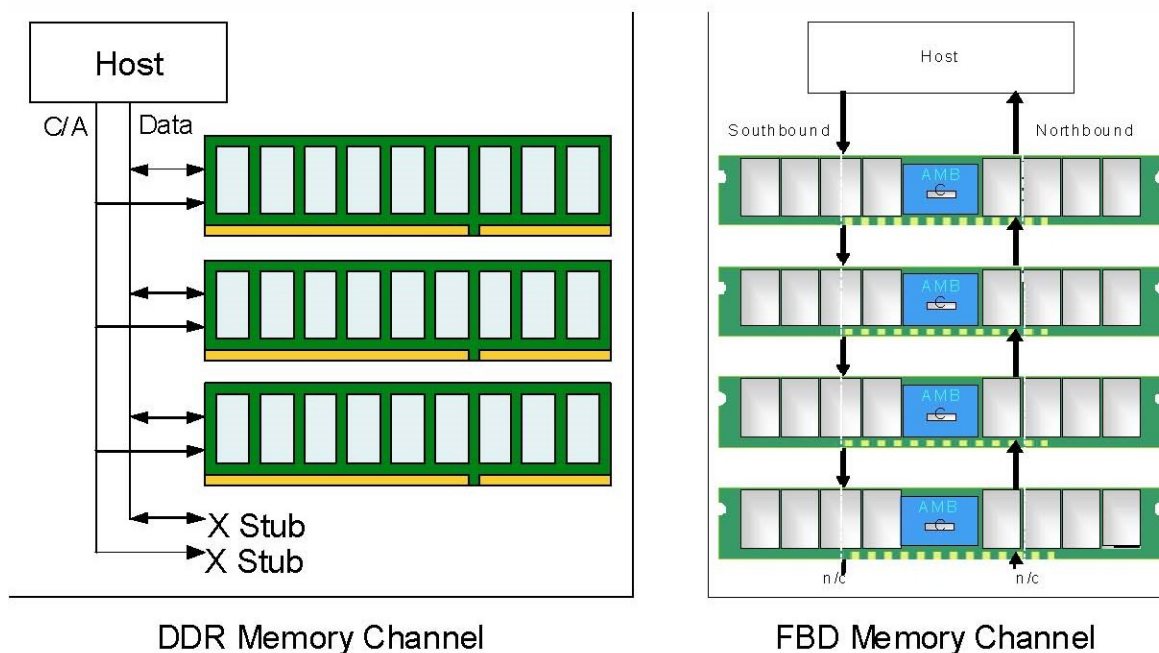
**Table 3-1 — Terms, Definitions, and Abbreviations (cont'd)**

Vss	Ground (0V)
Vddq	I/O buffer voltage for DDR-II buffers. Nominally 1.8 V

## 4 Test and Validation Overview

### 4.1 FBDIMM Overview

The Fully Buffered DIMM (FBDIMM) interface is a high speed (2.4 to 4.8 GT/s) narrow (67 pin) interface. The DFT strategy for FBDIMM has several objectives: first, because the memories on an FBDIMM can only be accessed via the AMB, DFT features in the AMB are required to allow economical testing of the memories. Second, DFT functions in the AMB can automate some test functions that can reduce the cost and improve the quality of DIMM testing. Third, DFT functions can facilitate system level debug operations such as memory channel integrity verification that will be useful to OEMs and end users of systems employing FBDIMM technology.



**Figure 4-1 — Memory Channel Architecture**

Traditional memory solutions are based on stub bus architecture with up to 4 DIMMs per memory channel. Normally the maximum speed of the bus is determined by the number of DIMMs on the bus. The FBD architecture connects DIMMs together in a daisy chain fashion using a high speed point to signaling scheme.

## **4.1 FBDIMM Overview (cont'd)**

Each DIMM has a buffer (Advanced Memory Buffer or 'AMB') that decodes/encodes data for local DRAMs and re-drives high speed data to the next DIMM in the chain. The signaling speed on an FBD channel therefore does not change as more DIMMs are added, although latency would increase. Note that there is no direct access to the DRAMs after DIMM assembly. Furthermore the native high speed interface is beyond the capability of conventional production testers. This makes DFT support on the AMB essential for the cost efficient production testing of FBDIMMs.

## **4.2 FBDIMM DfX Objectives**

The desired outcomes of the DFT/DfX strategy are:

- FBDIMM must be testable using conventional equipment and methods
- DFT will allow enhanced testing over traditional methods at component, DIMM and system level
- DFT will be compatible with tools and methods developed for other interfaces Evaluation of these objectives led to the several key test modes:
- Memory BIST to be used during buffer, DIMM test and system test
- Transparent mode, which allows access to the DRAMs behind the buffer
- IBIST – used for interconnect testing at DIMM and system level
- LAI mode – allows the AMB to act as an intelligent front end for a logic analyzer

## **4.3 DIMM Test Process**

DRAMs are tested at the manufacturer using standard test processes: sort, package test, burn in, and class test. There are variations in the use of parallel test and test time. DRAMs tend to have very long test times but due to the moderate pin count, many DRAMs (typically 64 or 128) are tested in parallel.

For DIMM test many suppliers use a 2 pass test. The first pass uses ATE or low cost leakage and continuity tester to verify connectivity, shorts and DRAM leakage. This step is also used to program the SPD device with appropriate parameters. The second step uses ATE or a motherboard-based tester. Test time on ATE varies depending on manufacturer or customer requirements. On the motherboard-based tester test times are typically one to three minutes, consisting of a simple boot test and memory stress tests. In some cases, suppliers use gravity-fed handlers to automate insertion of the DIMM. Motherboard system manufacturing uses a similar test strategy with memory test times of approximately 60 seconds.

For suppliers that rely on ATE, it is possible to test the DIMM in one insertion. One advantage of ATE is the DIMM can be tested with controlled timings, voltages and algorithmic patterns. If the DRAM topology is known the tester can apply topologically correct patterns, re-creating the conditions present during DRAM manufacturing.

The reason suppliers use stress tests is that DRAMs are sensitive to the surrounding electrical environment. This environment is different than that presented at DRAM component test, resulting in some level of fallout due to differences in loading, noise, or power delivery. Existing memory buffers (used on registered DIMMs) provide partial isolation, reducing electrical loading.

### 4.3 DIMM Test Process (cont'd)

Registered DIMMs have an added cycle of latency but the DRAM is otherwise accessible from the edge fingers of the DIMM. With FBDIMM the DRAM is completely isolated behind a high speed interface and a memory buffer. This reduces the load on the DRAMs but has the disadvantage of loss of direct access to the DRAMs from the DIMM connector. This isolation presents new challenges for failure analysis or application of specific DRAM tests.

Although it is possible to model much of the environment, it will take time and product volume to determine the impact on DIMM manufacturability and quality. Furthermore, it will take time to develop new manufacturing techniques while maintaining product quality. This places us in a position of needing to support traditional test methods while paving the way for improvements. For this reason there may be extra DFT features to facilitate learning and test process improvement.

FBDIMM supports ATE, motherboard-based testing or testing with dedicated fixtures. In a system setting the test process need not change significantly from the traditional process. System behavior will propagate to the DRAMs in much the same way it does with conventional DIMMs. For higher throughput and reduced test cost AMB DFT features may be used to deliver known patterns to the DRAMs. Several DIMMs can be tested in parallel using this method.

### 4.4 Memory Manufacturing Flow

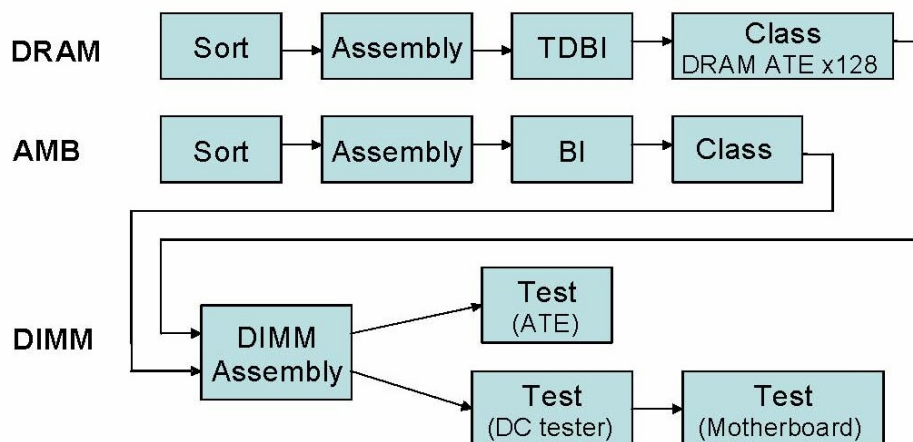


Figure 4-2 — Memory Assembly and Test Procedure

### 4.5 Reducing Test Cost

Most of the DFT features are aimed at improving testability and test coverage. However there is another benefit to be gained. Consider use of Memory BIST and IBIST at system power on. Rather than executing a traditional power on self test, an FBDIMM enabled system can run IBIST to verify the integrity of the installed DIMMs. This would include the DIMMs, connectors and other hardware along the path. The system can also use memory BIST to test and zero all DIMMs at the same time rather than one by one as in today's systems. This will save considerable startup time on systems with dozens of DIMMs.

The same principle can be applied to DIMM test. Memory manufacturers tend to use motherboards or other low cost testers to test DIMMs after assembly. With FBDIMM it is easy to construct a system capable of testing four, 8, 16, or more DIMMs in parallel.

## 4.6 Test Strategy

It is desirable to define a set of features that allow testing of the buffer, DIMM and system with little or no new capital equipment. In many cases FBDIMM features can be tested on older equipment through the use of DFT. The supplier of the AMB or DIMM will determine the equipment to be used but there are some useful guidelines.

A typical test strategy is illustrated in Table 4-1. Rows in the table represent key tests at various stages of integration (AMB, DIMM, system). The table columns show which DFx features would be used at each test step. During AMB test the functions of the buffer and the DFx features should be tested. Several of the DFx features can help with testing interfaces that exceed tester capability.

**Table 4-1 — Test Overview**

AMB test	ATE	Transp Mode	Mem BIST	IBIST	LAI Mode	Vendor Specific
(1st insertion, 100-200 MHz tester, FBDIMM looped back, DDR connected to tester)						
High speed IO test				✓		
Test AMB core function and defects	✓					
DDR functional test at speed	✓					
Verify thermal sensor	✓					
Test DFx features	✓	✓	✓			
(2nd insertion or relay on load board, all pins connected to tester)						
High speed IO leakage	✓					
Additional AMB functional or structural test	✓					✓
Test transparent mode	✓					
<b>DIMM Test</b>						
(1st insertion, all pins connected to tester)						
Assembly defects (opens, shorts)	✓					
High speed IO leakage	✓					
DRAM array test (ATE only)	✓	✓				
Program SPD	✓					
(2nd insertion or relay on load board, IO looped back)						
High speed IO functional test	✓					
DDR functional test at speed	✓					
<b>System Test</b>						
DIMM initialization, DRAM function	✓					
FBDIMM link test				✓		
<b>System Validation</b>						
Analog Inspection	✓	✓	✓			
Link observability			✓			
Protocol verification					✓	

## **4.7 AMB Test**

Most of the AMB features can be tested with the DDR pins connected to a tester and the FBDIMM pins configured for loop back testing.

There are two ways to loop FBDIMM outputs to inputs. The simplest method is to loop the ten southbound transmitters back to the receivers and the northbound Tx to Rx. The disadvantage of this method is that it may not allow testing of the re-time path in the IO.

To cover the re-time path, connect 10 SB lanes to 10 NB lanes, test the interface and then connect and test the remaining NB lanes. This results in a figure-8 type of loop back. The four remaining NB lanes may be connected using internal loop back if the design supports this. Otherwise external RF relays may be used to switch in the remaining lanes.

### **4.7.1 AMB Functionality and Defects**

Many of the DFX features can help with AMB self-test. AMB suppliers may implement specific functional or structural tests. The exact nature of these tests will vary depending on the internal buffer implementation and manufacturing test strategy. For the purpose of this document the primary focus is testing of the external interfaces and externally-visible functionality.

Testing at this step would include all of the core logic of the buffer, digital sections in the IO area, thermal sensor and other circuits. In some cases FBDIMM pins may need to be connected to the tester.

### **4.7.2 DDR Functionality and Timing**

DDR pins may be tested with low speed equipment by using the MemBIST engine to read or write DDR traffic at full speed. If patterns are constructed to read multiple bits from the tester, MemBIST may be used to over-sample tester data. For example, if MemBIST is programmed to expect "11110000" at 400 MTS, apply a "10" sequence from a tester at 200 MTS. This is usually possible even on 100 MHz testers with pin multiplexing. For DDR writes, MemBIST will drive data at full speed. This data can be sampled by a tester using multi-pass testing. Using a combination of these modes functionality and timing of MemBIST engine and the DDR interface may be verified.

In the example below, the tester is driving a "01" at 200 MHz while the AMB is programmed to expect four zeros followed by four ones at 800 MHz. This example uses a combination of user-defined data and circular shifting for the expected data pattern. Tester data needs to be delayed by varying amounts depending on the DQ line. For example DQ0 will drive 0 starting at 0ns while DQ1, DQ2, and DQ3 will drive the same data delayed by 1.25 ns, 2.5 ns, or 3.75 ns respectively. DQS timing would be offset in the middle of the DRAM cycle at 625 ps. One difficulty is DQS will need to toggle at 400 MHz. This may require a combination of pin multiplexing and surround by compliment formatting. This should be straightforward on a 200 MHz tester but will require creative timing on a 100 MHz tester. The problem of creating 400 MHz DQS for FBDIMM should follow established solutions for DDR2-800 DRAMs or unbuffered DIMMs.



#### 4.7.2 DDR Functionality and Timing (cont'd)

Tester Cycle	DQ Cycle	DQ pin							
		7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	1	0
0	1	0	0	1	1	1	1	0	0
0	2	0	1	1	1	1	0	0	0
0	3	1	1	1	1	0	0	0	0
1	4	1	1	1	0	0	0	0	1
1	5	1	1	0	0	0	0	1	1
1	6	1	0	0	0	0	1	1	1
1	7	0	0	0	0	1	1	1	1
2	8	0	0	0	1	1	1	1	0

**Figure 4-3 — 200 MHz Tester Driving 800 MHz DQ**

#### 4.7.3 FBDIMM Functionality and Timing

The FBDIMM interface can be tested with IBIST. IBIST provides a programmable pattern generator and checker for each FBDIMM path (one for north and one for south). This capability allows generation of streams of traffic at the normal data rate of the channel. The tester will need to provide a reference clock of 100 MHz to 200 MHz. IBIST registers can be programmed from the SMBus, which will be connected to the tester.

The AMB supplier may choose to implement additional capability to stress the FBDIMM interface. These methods will vary somewhat depending on the specific IO buffer design. Note that all AMBs will support reduced drive strength on FBDIMM links. This reduced drive (60% drive strength) will create smaller than normal swing allowing verification of voltage margin. Design capability signal quality can be verified during AMB characterization using high speed lab equipment.

#### 4.7.4 Connection of FBDIMM Pins to the Tester

Several features of the AMB require connection to the tester. For these situations there are three possible approaches:

- Two insertion testing: In one insertion the FBDIMM pins are looped back as described above. For the second insertion the pins would be connected to the tester. This is a straightforward test method but undesirable as it involves two test steps, two passes through the tester and handler, additional material tracking and so forth.
- RF relays: RF relays may be used to allow connection of pins to the tester for part of the test program and loop back for the other. There are manufacturers that specialize in relays of this type.
- Two socket testing: This is similar to two insertion testing but there is a looped back socket and a direct connect socket on the same load board. Automatic handling equipment can be used to move the AMB from one socket to the other, completing the AMB test in one step. Depending on tester and handler capability one might test two buffers, switching the positions of the devices mid-test.

#### **4.7.5 FBDIMM and DDR DC Tests**

Assuming the FBDIMM and DDR pins are connected to the tester, one may perform DC testing in the usual manner with tester pin electronics. The test limits to be used are described in the AMB component specification.

The DFX IO chapter describes more sophisticated IO test methods using self test circuitry. These methods allow IO testing without connection to the tester.

#### **4.7.6 Testing DRAM Access Features**

Transparent mode assumes connection to the tester for proper operation. This will require a second test insertion or relays as described above. Since transparent mode is only used at DDR2-400 timings the tester will supply a 100 MHz reference clock. In this mode each leg of the FBDIMM differential inputs will supply data to the AMB. This data will be presented to the DDR interface but delayed by one clock. Transparent mode pin mapping and timing are described later in this standard.

If the AMB supplier has implemented structural test methods (e.g. logic scan chains) these may be used in place of some or all of the functional tests described above.

### **4.8 DIMM Test**

A key objective of DIMM testing is to verify proper assembly and connectivity of the components on the DIMM.

#### **4.8.1 Assembly Defects**

Assembly-related opens or shorts on the DDR interface may be detected with a memory BIST pattern. Unlike older DIMM technology the FB DIMM uses ball-grid packages. This complicates opens and shorts testing since there may be several neighboring pins or traces corresponding to each ball location. One of the capabilities provided in the MemBIST engine is the ability to rotate data through the 72-bit DQ bus. This can be used during opens and shorts testing by walking a one through a field of zeros and vice versa. MemBIST can also increment addresses in the X, Y or XY direction. Combining these capabilities, fairly compact patterns may be constructed that test all possible address and data lines. The command lines will toggle as a by-product of the test, allowing us to verify these as well.

#### **4.8.2 Leakage Testing**

The FBDIMM pins may be tested at DIMM level as described above for AMB testing. If the pins are connected to the tester as with transparent mode the pins may be tested directly. If the pins are not connected to the tester they can be tested using the IO techniques described later in this standard.

### **4.8.3 DRAM Array Test**

In many cases the exact topology or sensitivities of a DRAM are not known. In these situations it is probably best to use random data or other MemBIST or system-generated patterns to verify the DRAM on DIMM.

If the DRAM topology is known, or there are other DRAM specific tests, transparent mode can be used to gain direct access to the DRAMs behind the AMB. Transparent mode is designed to operate at 200 MHz, which is the maximum speed within the DRAM array. By varying timing of address and data arrival it is possible to test the array and its associated timings such as T<sub>cl</sub> or T<sub>rcd</sub>.

### **4.8.4 Programming the SPD**

The SPD device may be programmed at any time during the DIMM test flow using the same methods as any other DIMM.

### **4.8.5 FBDIMM IO Test**

As with the AMB component the DIMM FBDIMM pins may be looped back and tested with the interconnect BIST engine. The IBIST engine allows definition of the type of pattern(s) desired and the length of time to run the pattern(s). The selection of patterns may vary somewhat from the standalone AMB due to DIMM routing. In most cases, targeted patterns should be created that have lone pulses, neighboring traces in opposite states and so forth to detect signaling anomalies.

### **4.8.6 DDR Interface Testing**

The DRAM on DIMM may be tested at full speed using the MemBIST engine. As described later in this document, the engine is can read, write or run a few industry-standard algorithms with various data patterns and addressing schemes. The ability provided should be adequate for testing the interfaces and internal multiplexers, decoders and related logic inside the DRAM.

## **4.9 System Test**

At system level the AMB provides the ability to test and initialize memory as well as testing the FBDIMM links from host to DIMM and DIMM to DIMM.

Memory initialization may be accomplished with the MemBIST engine using one of the built-in or a user defined pattern. The MemBIST engine can initialize ECC bits to the correct value for shorter ECC codes. If the system uses a more complex ECC code it may be necessary to initialize some or all memory in the usual manner through the host controller.

AMB and DRAM functionality may be verified at system boot time with the MemBIST engine. It is noteworthy this process can be executed much faster than with conventional methods. Multiple DIMMs may be tested at the same time, reducing test time considerably. Since the MemBIST engine can generate data at the full DRAM rate, the test will consume considerable power, taxing both DRAM/DIMM cooling and the system power supplies. The number of DIMMs that can be safely tested and the duration of test will vary depending on system design.

The FBDIMM links can be tested in much the same way as AMB and DIMM. However, testing will be from one link agent to another (between two AMBs for example). The selection of patterns should be tailored to the topology of the memory channel in the system.

#### **4.10 Validation**

Link observability is supported in some AMB implementations with logic analyzer mode. In this mode the AMB will de-serialize FBDIMM traffic (both north and southbound) and present the data on the DDR pins. This data stream will be one sixth of the link rate, or 800MTS for a full speed 4800 MTS link. The LAI interface is designed to be usable on a logic analyzer or similar equipment. It is expected that equipment suppliers will provide LA probes and related instrumentation to be used with this mode.

The electrical behavior of the FBDIMM interface can be tested by programming appropriate test sequences into the IBIST engine, letting the engine run in continuous mode and, using probes and an oscilloscope, verifying that the interface meets the voltage and timing specifications. This testing may be performed at any level of integration from standalone AMB to complete systems.

---

### **5 Requirement Summary**

---

#### **5.1 DFx Requirements**

FBDIMM DFT, DFV and related features are intended to facilitate testing, validation and coverage of defects. The key requirements are:

- DRAM access: The AMB must allow testing of DRAM behind the buffer by direct access, BIST or a combination of the two.
- Determinism: Operation must be deterministic in all modes.
- Tester connectivity: FBDIMM interfaces must allow connection to low speed (100-200 MHz) testers.
- Control and Observation: All DFx features must be controllable either in-band (as applicable) or SMBus. The AMB will provide features to allow observation of internal registers, link behavior and DRAM response. There must be debug hooks in the AMB and host interface to support validation/debug of FBDIMM components and the systems employing them.

#### **5.2 Required Features**

DFT features that relate to normal functionality or test modes for manufacturing must be included in all AMBs and must function as described in this specification. DFT features that support debug or evaluation are strongly recommended but are not required. It is permissible to implement additional, AMB specific innovations. AMB-specific features will not be considered part of the FBDIMM DFx specification. The specific breakdown of required features is listed in Table 5-1.

## 5.2 Required Features (cont'd)

**Table 5-1 — Required DFT Features**

Feature	Description	Required?	Anticipated Use		
			AMB	DIMM	System
Transparent Mode	Allows access to DRAM, can be used for DRAM core test; requires ATE or equivalent stimulus/response	Required	Yes	Yes	No
Memory BIST	Test DDR interface and DRAM functionality	Required	Yes	Yes	Yes
Interconnect BIST	Tests high speed link integrity; used during initialization	Required	Yes	Yes	Yes
DDR Leakage	Indirect DC leakage test of DDR interface	Required	No	Yes	Yes
Error Injection	Allows injection of errors to test controller/BIOS response	Required	Yes	No	Yes
Logic Analyzer Mode	De-serializes high speed link for analyzer or capture; needed for LAI or similar capture tools	Recommended	Yes	No	Yes
Voltage Margin	Transmitter drive strength settings to stress FBDIMM link	Required	No	No	Yes
Timing Margin	Receiver timing control allow intentional offset of data eye	Recommended	No	No	Yes

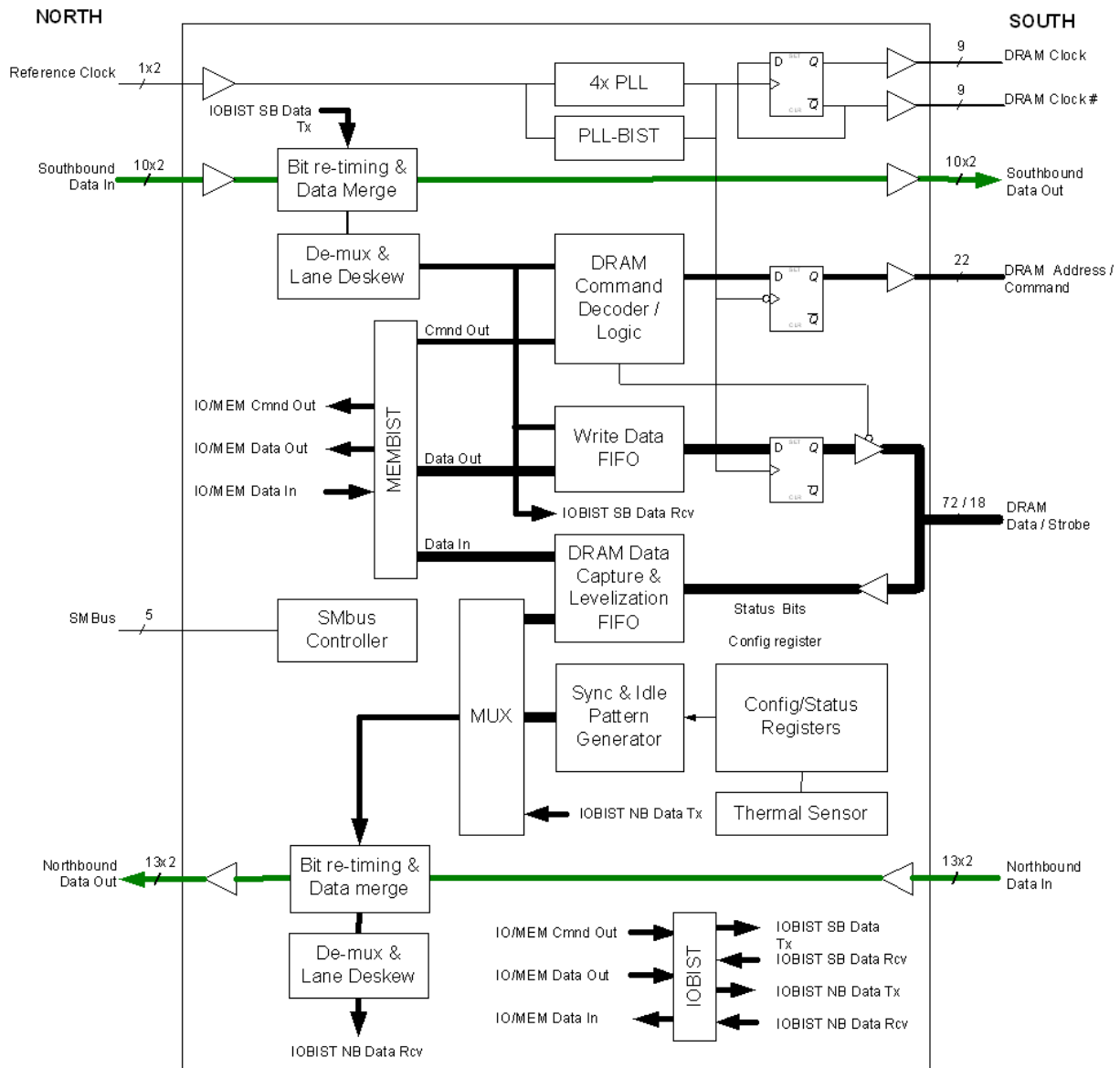
## 5.3 DFx Features and Usage

The FBDIMM DFx features are summarized in Table 5-2. As indicated by type, the DFx features support a variety of requirements: ATE test, burn in, IO interfaces and margin testing. The major DFx features include:

- MemBIST for memory initialization at system power on and testing the DRAM behind the buffer
- Interconnect BIST, which allows testing of the high speed interconnects at system level
- LAI mode, which de-serializes link traffic and presents in a form usable by a logic analyzer for link and protocol debug
- Transparent mode which allows a tester to connect to the high speed pins and send data to/from the DRAMs behind the buffer

## 5.4 AMB Overview

The major DFT blocks are the MemBIST engine and IBIST block. All DFT modes and registers are accessible from the SMBus controller.



**Figure 5-1 — AMB Architecture and DFT Blocks**

### 5.4.1 Memory BIST

The DIMM has two primary failure mechanisms: assembly related defects and unexpected electrical behavior. Assuming DRAMs have been tested prior to assembly, there is less need to look for cell failures, although the need will arise from time to time to support re-screening of DRAMs for issues discovered after DIMM assembly. In most cases simple patterns can verify proper connectivity between the buffer and DRAM. If there are connectivity or gross DRAM component failure the BIST engine will be used to identify the specific failing net or DRAM to be replaced.

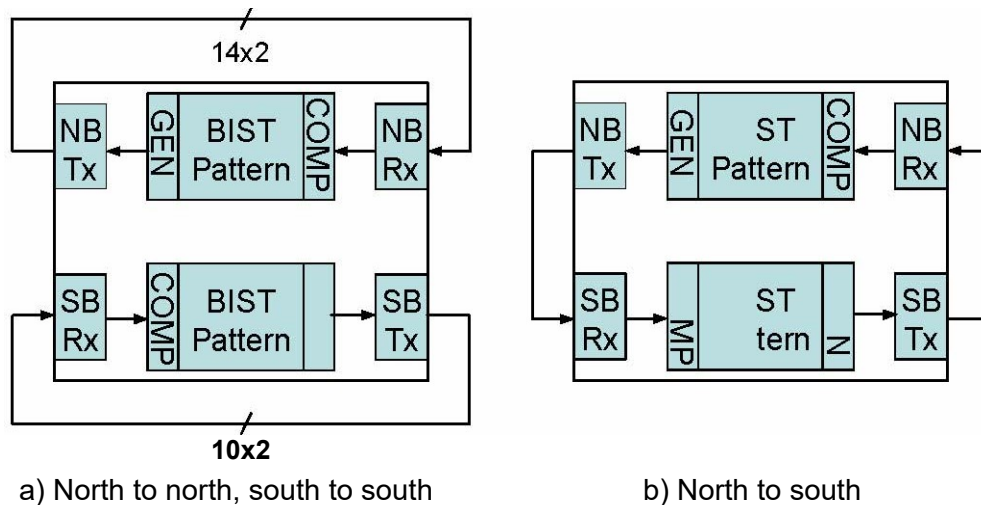
### 5.4.1 Memory BIST (cont'd)

To stress electrical behavior or to facilitate failure analysis there is a need to generate patterns to isolate electrical failures or interactions. Some of this analysis will be completed in systems and some will be conducted on testers, creating a need to accommodate both methods.

### 5.4.2 Interconnect BIST

As illustrated in Figure 5-2 there are two pattern generators and comparators: one for southbound data and one for northbound data. The pattern generator is used to create fixed patterns during link initialization or programmable patterns for IO buffer testing.

For buffer component test, the north and southbound drivers and receivers may be connected as shown. This will allow testing of the IOs with various test patterns or other conditions. There are two methods for loopback. The simplest is to connect the southbound outputs to southbound inputs and north to north. The advantage of this approach is the lane widths are balanced. The disadvantage is it may be possible to test only the resync path through the AMB. With the second approach we gain the ability to test the re-sample path, with some added complexity due to the imbalanced lane widths. In this mode the tester will have to drive 10 pins in one pass and the remaining 4 in a second pass. This may be accomplished with two insertions or with RF relays on the tester load board.



**Figure 5-2 — AMB IO Loopback Methods**

## 5.5 DDR Interface Tester Compatibility

The AMB interfaces with the DRAM memory through the DDR interface which runs at a data rate of 800 MT/s. With testers capable of only supporting a max data transfer rate of 100 to 200 MT/s, there exists the issue of having to exercise the core behind DDR I/O at speed functionally while running the I/O's themselves slower.

One solution is to apply the same data for more than one DDR clock, reducing the input data rate to 200 MT/s. For example, one might apply "01" at 200 MHz, which would be interpreted as "00001111" at DDR2-800 speeds. In this mode MemBIST can generate full speed address and data. The DDR outputs can be sampled with 2-pass testing in this mode to verify AMB operation. Transparent mode should be used to verify proper operation of the DRAM core.

## **5.5 DDR Interface Tester Compatibility (cont'd)**

Supplier-specific DFT may be added to the DDR interface to facilitate testing on specific equipment. There are several possible approaches depending on supplier preference and equipment capability. It is not expected suppliers will use full speed testers in production as this would contribute significantly to the overall cost of the AMB.

### **5.5.1 Logic Analyzer Interface**

In addition to the full complement of DFT features, the buffer will support validation and debug by having a mode to act as a transparent repeater which captures North- and South-bound link traffic, passing it de-multiplexed and framed to a logic analyzer. The high speed ports (link interfaces) run at 2.4 to 4.8 GT/s. At these frequencies it is virtually impossible to physically probe the signals. In addition, input to the logic analyzer capture module is limited to operation below about 800 MHz. To resolve this issue the buffer operating in a LA interface mode decodes link activity and sends the decoded, de-multiplexed data out through the DRAM pins for direct input to LA capture modules. In this mode the north and south-bound links will pass data to the next buffer in the normal manner. Triggering events on the links will be recognized using protocol-aware logic and match registers in order to supplement native triggering capabilities of the logic analyzer. Simple filtering of events may be performed in the buffer with simple signals passed to the logic analyzer to qualify trace storage.

LAI mode can be incorporated in the AMB design or in a dedicated chip. This function is recommended but not required as a test mode in the AMB design.

An LAI card is being developed to provide link traffic trace capability to support debug and validation needs for systems featuring FBDIMM links. The FBDIMM LAI probing solution is intended to support silicon debug and evaluation of memory channel behavior.

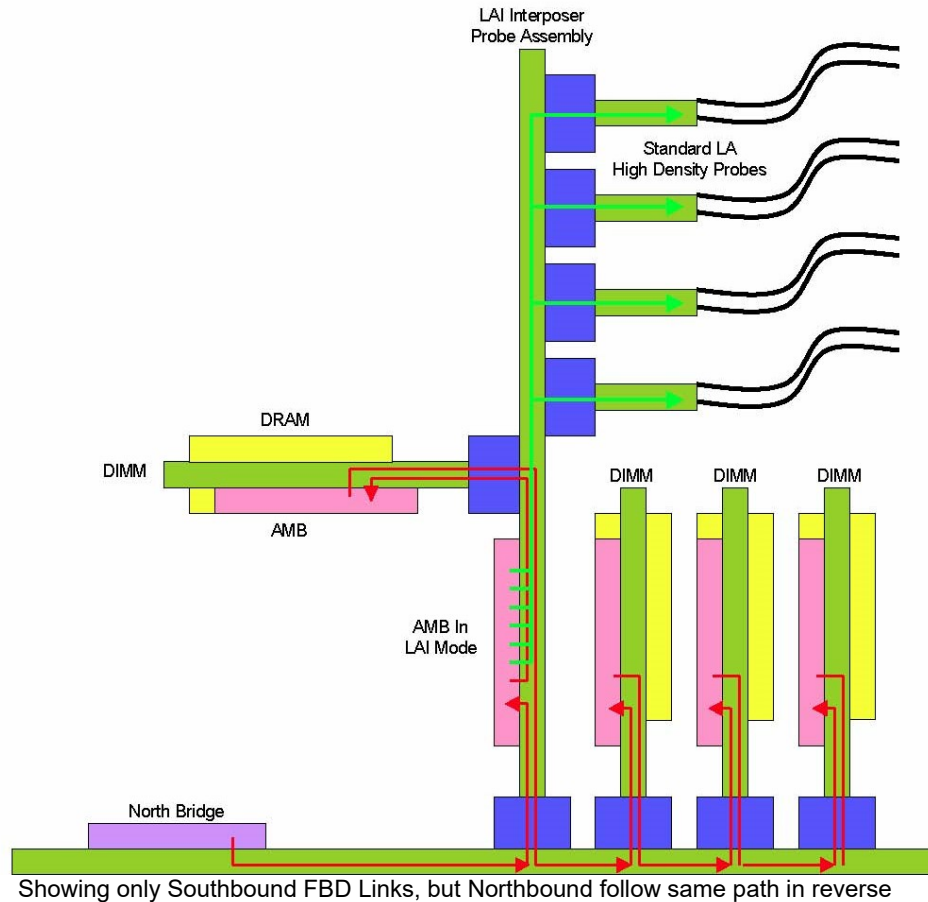
In the LAI mode, the AMB performs the following functions:

- Retain normal repeater pass-through southbound and northbound links.
- Perform enough link protocol unwinding to provide extraction of frame boundaries, idle filtering opportunities, and pattern match triggering.
- Enable independent de-multiplexed Northbound and Southbound traffic stream in LA compatible signal levels and timing format using AMB DDR IO pins.
- Provide system independent SMB access to configure the AMB in LAI mode.

As illustrated in Figure 5-3 the AMB in LAI mode passes data to the next AMB in the chain as normal as well as providing decoded data and event triggers to a logic analyzer via the DRAM pins. The pass through data may be sent to the displaced DIMM, allowing full channel memory capacity while in LAI mode.



### 5.5.1 Logic Analyzer Interface (cont'd)



**Figure 5-3 — Logic Analyzer Interface Card**

The LAI interposer features a single buffer chip which operates in a Logic Analyzer Interface (LAI) mode, a standard slot for installation of the first DIMM on the chain, and connectors (or direct attach cables) for output to a logic analyzer. The connector for the interposed DIMM is assumed to be the normal vertical connector, but alternate connector styles, such as angled or straddle mount, might be more appropriate to minimize possible physical interference problems. Users would install the LAI interposer on one or more FBDIMM links within a target system to trace link activity. Additional cables to provide LAI power, SMB programming and cross-triggering between multiple LAI are not shown, but would be required.

### 5.5.2 DC Testing

At component level all pins may be tested in the conventional manner. The BIST engine can be programmed to drive outputs to the correct state for drive strength testing.

At DIMM level, most assembly-related defects can be detected with a simple functional test. The MemBIST engine may be programmed to walk a one or a zero across the 72 data bits. Toggling one pin at a time will detect all possible open or shorted conditions. The same is true for address lines - the walking data pattern (or other BIST pattern) may be repeated over the address space.

## **5.6 Host Controller Requirements**

The host controller will have access to the buffer either in-band through the FBDIMM memory channel or out of band through SMBus. Since FBDIMM has a controller-target architecture the host will be responsible for initializing the channel and directing operations within the memory buffer. For the most part this is no different than normal operation. However the host must understand how to work with buffers that are in LAI or other special modes of operation.

---

## **6 AMB Component DfX**

---

### **6.1 Normal Mode Debug/Validation Hooks**

Each AMB used to buffer DRAM shall require a minimum set of debug features to support debug/validation of both HW and SW at the channel, component, and system levels. These features are defined in the following clauses.

### **6.2 Error Injection and In-Band Events**

Debug and validation require selectable stimuli to drive timing of response mechanisms.

#### **6.2.1 Decode of Southbound In-Band Debug Events**

Each AMB shall support detection and decode of In-band Debug Commands and conversion of each of the eight carried events into local event pulses. These local events then can be used as stimuli for various response mechanisms defined below.

#### **6.2.2 Pattern Match on a Command**

The AMB will support pattern matching on command position A, B or C using one mask and one value register. This local event may be used as stimulus for various response mechanisms defined below. If LAI mode is supported as described below the pattern matching mechanism in LAI should be used instead.

#### **6.2.3 Programmed Event Delay**

Because it is difficult to time sensed events (in-band or command match) with the desired response time, the AMB must provide a programmable event delay. This programmable delay must accommodate a minimum of 63 frame times.

#### **6.2.4 Error Injection**

The AMB supports injection of specific errors in response to selected in-band events or stimulated by mask/match of commands arriving at the AMB, or in one case (stuck lanes) directly via control registers on the AMB.

There are several types of errors that buffers must be able to inject in order to enable validation and debug hardware and software mechanisms intended to deal with each error type in operating systems.

#### **6.2.4.1 Errors Injected In Northbound Read and Read Data Frames**

In response to a selected local event, the AMB will inject an error in frame data during or after calculating frame CRC's to be transmitted Northbound.

##### **6.2.4.1.1 Force Alert**

In response to a selected local event, the AMB will force an alert Northbound. Note that forcing an error status in a following status block will likely be required.

##### **6.2.4.1.2 Selectively Force "Stuck" on Lanes**

This capability is required to test ability of components and system to accomplish lane fail-over. Note that this is the only error injection feature which is not event driven, but rather shall be controlled directly by a register in the AMB.

#### **6.2.5 Sourcing Northbound In-Band Event**

The Northbound event shall be asserted in the next transmitted status block following assertion of a selected local event.

### **6.3 Logic Analyzer Interface Mode**

Implementation of Logic Analyzer mode in the AMB is recommended, but it is not a required AMB feature. If implemented, LAI mode must follow the description in this specification. Note that LAI mode requires more sophisticated pattern matching capability than that described above for error injection. If LAI mode is supported the LAI pattern matching mechanism should be used for error injection.

The AMB in LAI mode converts the DRAM pins from the normal use to drive logic analyzer probes instead. The logic analyzer requires more pins than can be supplied from the data bus alone, so some of the CA pins are re-used to drive additional LA signals.

The primary uses of the LA features will be debugging FBDIMM or system issues that will have visibility through the FBDIMM channels or observing the AMB for correct behavior in a system environment. This observability may be used to verify hardware, software or BIOS functionality.

### 6.3 Logic Analyzer Interface Mode (cont'd)

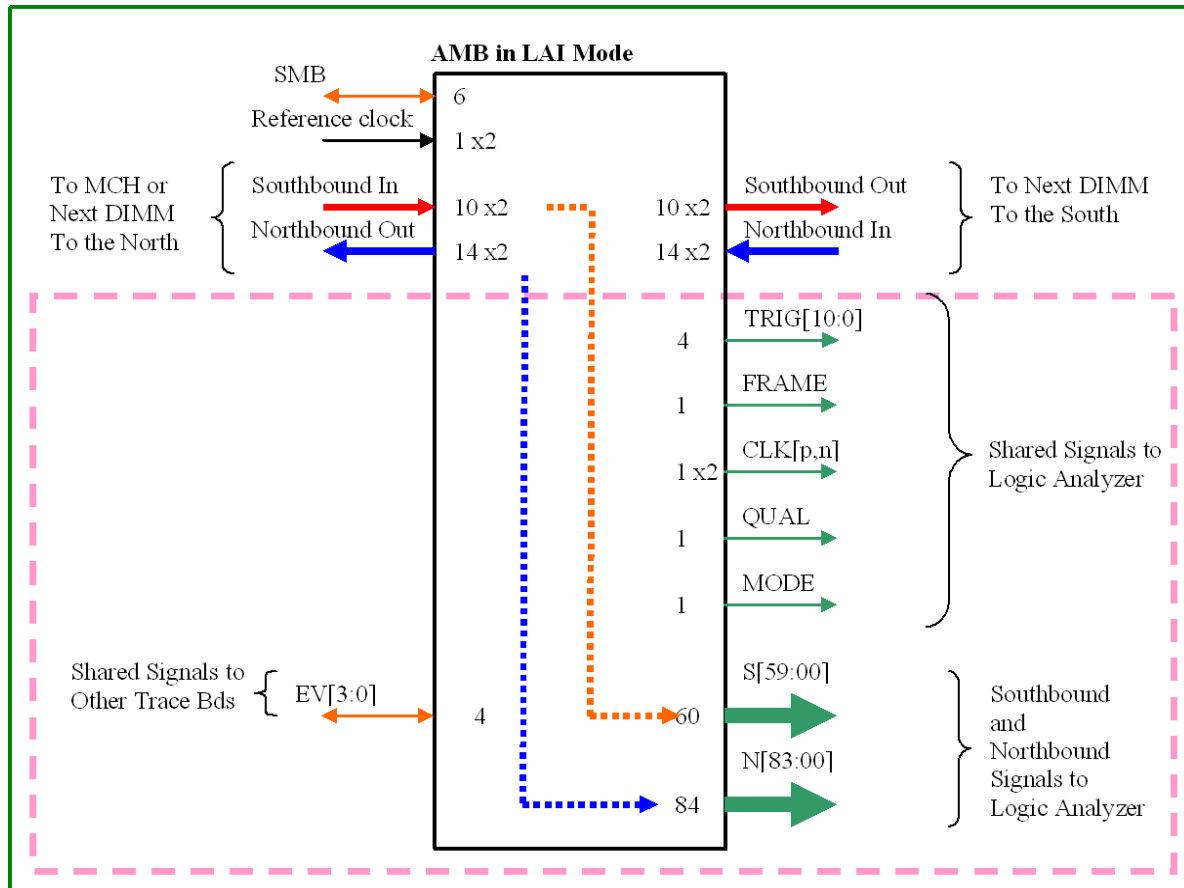


Figure 6-1 — AMB LAI Mode Signals

#### 6.3.1 Link Protocol Validation/Debug

In this scenario, traces will be captured of single FBDIMM links performing simple to complex operations. The traces will be used to debug low level link protocol/operations (link init, retraining, power transitions, resets, error recovery, and other normal link transactions). For this purpose the host side agent would usually initiate traffic as result of diagnostic or focus test software execution, rather than operating system/application activity, since this would allow specific behaviors to be produced repeatedly and without significant interference due to other link activity. Analysis of the traces could be primarily manual, since it is expected to be fairly simple, although automated checkers could also be employed to check rigorously for protocol flaws.

#### 6.3.2 LAI Debug Examples

In this case traces would be used to debug/validate BIOS (and other SW) configuration of buffer chip register write and read sequences as part of system initialization.

Memory controller logic interaction with AMB may be captured to enable debug of DRAM control issues. FBDIMM traffic would reveal the exact sequences and timing of interactions, aiding diagnosis of the problem. The FBDIMM trace could also be used to validate and debug test software used to stress the memory components.

### 6.3.3 System Level Debug

Traces of FBDIMM traffic under normal operating system and application execution might reveal important clues about misbehavior of logic elsewhere in the system. In this case the interaction with FBDIMM and DRAM may work flawlessly, but the content and/or sequencing of the traffic on the link can be used to provide indirect observability of system behavior. The trace data, with time stamps, could also be combined with other data for a complete record of the system state leading to an event or failure.

#### System Debug Case - FBDIMM on Processor

When a FBDIMM channel is hosted directly by a processor chip, additional debug data will be needed as other trace data won't provide adequate clues on what agent initiated transactions to memory via the FBDIMM channel. A debug mode supporting additional debug bits per command can provide agent ID and/or other debug information (definition for exposing these added debug bits synchronously in FBDIMM frames is provided in later clauses).

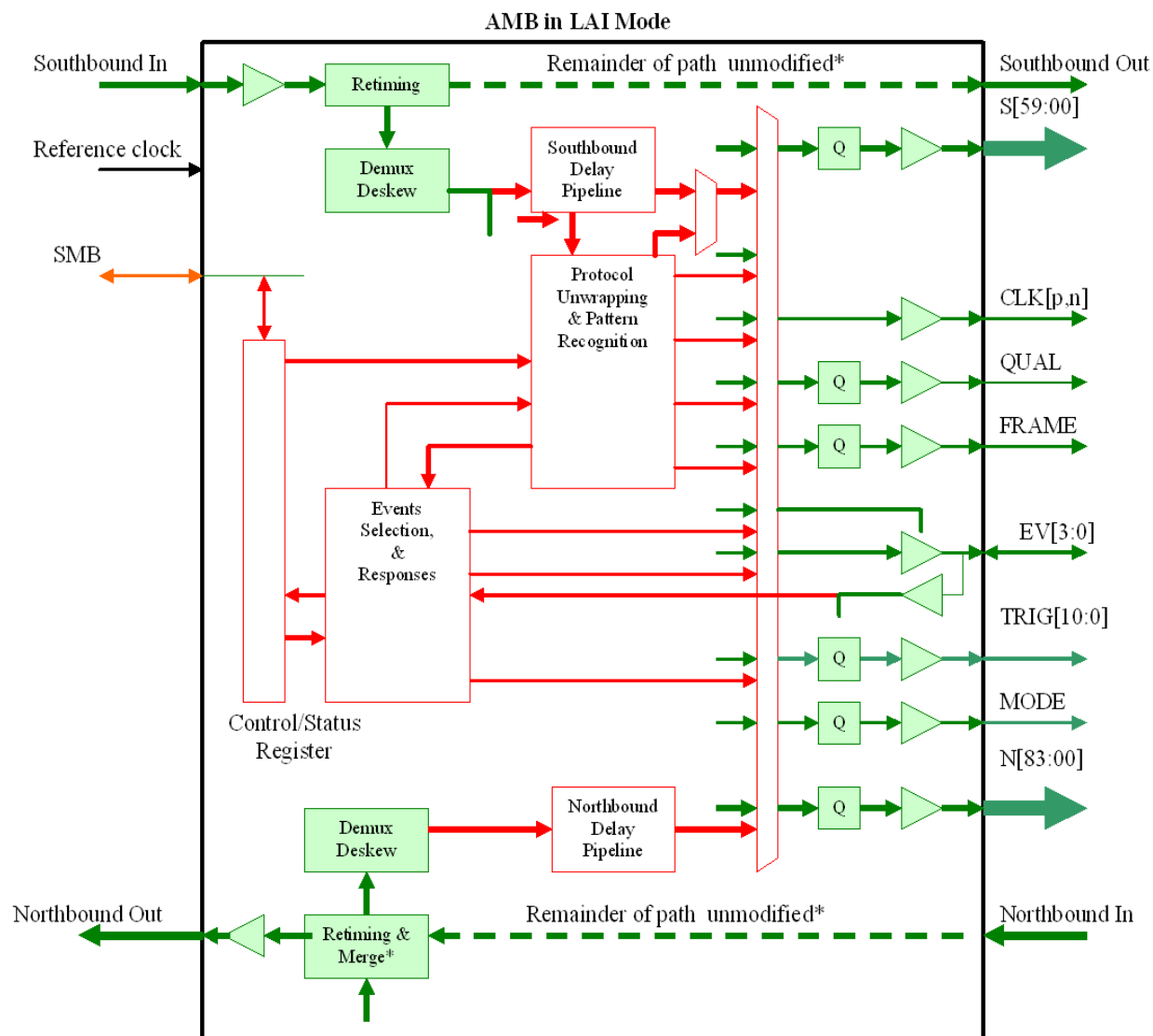
### 6.3.4 LAI Mode Architecture

The diagram below illustrates the AMB as a functional block when used as an LAI. The normal Southbound and Northbound links, the reference clock, and the SMBus are used “as is” with the DRAM interface reused for added LAI functionality. To be effective in collecting useful FBDIMM traces, the information provided to a logic analyzer must include not only a demuxed copy of the direct information transferred on the links, but also several types of derived information that a logic analyzer is not equipped to derive itself. These include specifically:

- Cross-triggering information with finer timing granularity than LA can achieve
- Filtering (qualified storage) opportunities recognition
- Traffic framing

A normal mode AMB will have to be configured to operate in the LAI mode. The AMB BFUNC and SA[2:0] pins define the buffer operating mode (normal, repeater, and LAI). The last four combinations of those bits represent the LAI mode (see Table 6-1). The two most significant pins (along with a fuse) have to be correctly set to enable the AMB to function as a Logic Analyzer Interface buffer.

### 6.3.4 LAI Mode Architecture (cont'd)



**Figure 6-2 — AMB LAI Mode Architecture**

**Table 6-1 — AMB SMBus Addressing for Logic Analyzer Mode**

AMB BFUNC, SA[2:0]	AMB Function (other modes not shown)
1100	Logic Analyzer Interface 0
1101	Logic Analyzer Interface 1
1110	Logic Analyzer Interface 2
1111	Logic Analyzer Interface 3

### **6.3.4 LAI Mode Architecture (cont'd)**

As the most two significant bits are pulled high to define the logic analyzer mode, only four AMB or similar devices can be addressed on a single SMBus, i.e., with the addresses 1100, 1101, 1110, and 1111. Also, to avoid any conflict and errors in configuration programming via the SMBus, the LAI configuration registers should not normally be dually accessible by the FBDIMM link and LAI.

#### **6.3.4.1 Configuration of LAI Device**

The Logic Analyzer Interface buffers will only be addressable using SMBus. The AMB device on the LAI card will not normally be connected to the System SMBus channel, except immediately following channel hard Reset and possibly at other (infrequent) times to set critical link interface parameters necessary to allow the LAI to operate on the link. These shall necessarily occur prior to starting any link training. Normal isolation of the AMB-LAI from the system SMBus is necessary to avoid conflict between the system controller and AMB-LAI configuration controller when the LA is setting LAI mode parameters. An external SMBus controller will be used to configure the AMB-LAI. In the laboratory, more than one AMB-LAI device can be on the same SMBus, in which case there is an additional constraint that the SMBus addresses for each AMB-LAI must be configured with different values to be able to program each separately.

#### **6.3.4.2 Pin Assignment**

In LAI mode the DRAM interface pins will be remapped to logic analyzer mode signals to send traffic and data to the logic analyzer. All the LAI signals (except the event bus pins) will transmit signals to the Logic analyzer. The event pins will both transmit on and receive data from the event bus. The AMB-LAI signals must be correctly mapped to the appropriate DRAM interface IO, as not all the AMB-LAI signals have the same electrical characteristics.

#### **6.3.4.3 DDR Pin Reuse**

The DDR interface, i.e., i/o buffers and pins will be re-used to transmit or receive the LAI signals. Based on the available DDRII pin out, there are just enough DDR pins to map against the required LAI signals. Therefore, no additional pins will be added to accommodate the AMB-LAI mode. Most LAI signals are required to operate at equivalent DDR speed. The LAI signals will be single-ended, except the clocking, which will be the differential DRAM clock.

All AMB-LAI signals except the Event signal i/o will transmit data to the logic analyzer. The event bus does not connect to the logic analyzer but has a daisy chain topology connecting all AMB-LAIs. This will allow communication of event triggers between AMB and similar components.

#### **6.3.4.4 FBDIMM Signals**

The FBDIMM signals will be operating unmodified from the default normal mode. The signals will connect to the preceding AMB or host-interface on the primary side. In the secondary side it will connect to the downstream DIMM AMB. The FBDIMM channel will operate at link speed. The AMB-LAI along with the two AMB on the two downstream DIMMs must run at full strength (and correct pre-emphasis level) even if the other AMB-DIMMs are configured differently to operate in half strength mode to conserve power. Check design guide to determine which of the SB and NB in the two downstream AMB devices must be operated in full strength.

#### 6.3.4.4 FBDIMM Signals (cont'd)

The default drive strength in AMB-LAI will be full strength. The System SMBus may have to be provided direct access to the AMB-LAI to configure select registers which are necessary to make the channel acceptably functional. FBDIMM systems should not be allowed to reconfigure the drive strength register in the AMB-LAI using the in-band commands. To avoid potential conflict, the LAI SMBus management resource should not be used to modify these CSRs.

#### 6.3.4.5 LAI Signal List

The LAI signals at the DRAM interface are listed below. All of the AMB-LAI signals (except the Event signals) will transmit data to the logic analyzer. The event bus does not connect to the logic analyzer in its topology. It implements a daisy chain to connect AMB-LAIs. This allows transmission and receipt of event triggers between AMB and similar components. More description are given later.

**Table 6-2 — LAI Mode Signal and Pin Count**

AMB Signal in LAI Mode	Signal Name	Pins	Comment
Northbound Data	N[83:0]	84	FBDIMM signal
Southbound Data	S[59:0]	60	FBDIMM signal
Trigger signal	TRIG[10:0]	11	LAI mode signal
Clk – differential	CLK[1:0]	2	One differential clock
Qualifier	QUAL	1	LAI mode signal
Frame boundary identifier	FRAME	1	LAI mode signal
Event signals	EV[3:0]	4	AMB-AMB shared signal
FBDIMM Mode signal	MODE	1	LAI mode signal
Total		164	

#### 6.3.4.6 Pattern Matching

The LAI block can pattern match on three command values at any of three positions and combine the results into independent and combined local events. There are 13 total pattern matching events: a command value matches a command slot (9 events), a command value matches any slot (3 events), and all command values appear in the frame (1 event).

A pictorial presentation of the LAI match and mask logic is shown in Figure 6-3.



### 6.3.4.6 Pattern Matching (cont'd)

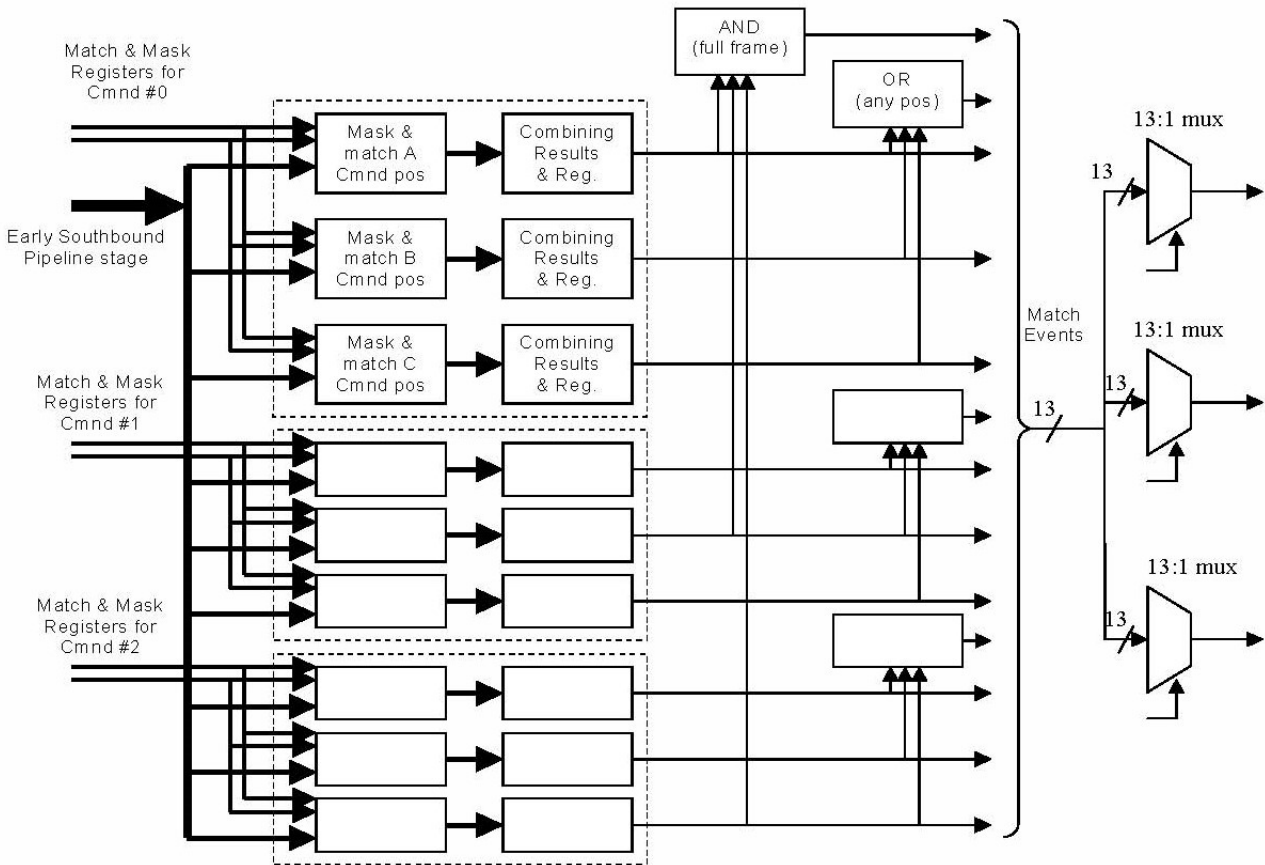


Figure 6-3 — Match and Mask Logic

### 6.3.4.7 Qualification

The AMB in LAI mode sends a qualification signal on a DDR pin along with each frame. The logic analyzer may use the qual signal to capture data when qual is asserted, and ignore data when qual is de-asserted.

Once a qualified frame is seen, the qual signal is asserted, and it remains asserted for an additional programmable number of cycles, using a timer ranging from 0 to 63. This timer is restarted if it is already running when a new qualified frame is seen.

The error injection timer will be used to push out the triggering of the QUAL\_STOP signal by N timer count of clock cycles (where N is user programmed for delay range of 0 to 63), thus increasing the length of the QUAL\_FLAG interval. One QUAL\_START event will enable the assertion of QUAL\_FLAG and another QUAL\_STOP event will disable assertion of QUAL\_FLAG.

A frame of all NOPs is not a qualified frame. A sync frame is not a qualified frame if the filter\_sync configuration register bit is set; otherwise it is considered qualified.

### 6.3.4.7 Qualification (cont'd)

If the `qual_mode` configuration register bit is not set, frames are only qualified if they occur between `qual_start` and `qual_stop` events. A frame that triggers `qual_start` may also cause the `qual` signal to assert, and a frame that triggers `qual_stop` will not be considered qualified. The `qual_start` and `qual_stop` events are programmable and are selected from the 32 local events.

A block diagram of the qualification signal control logic is shown in Figure 6-4.

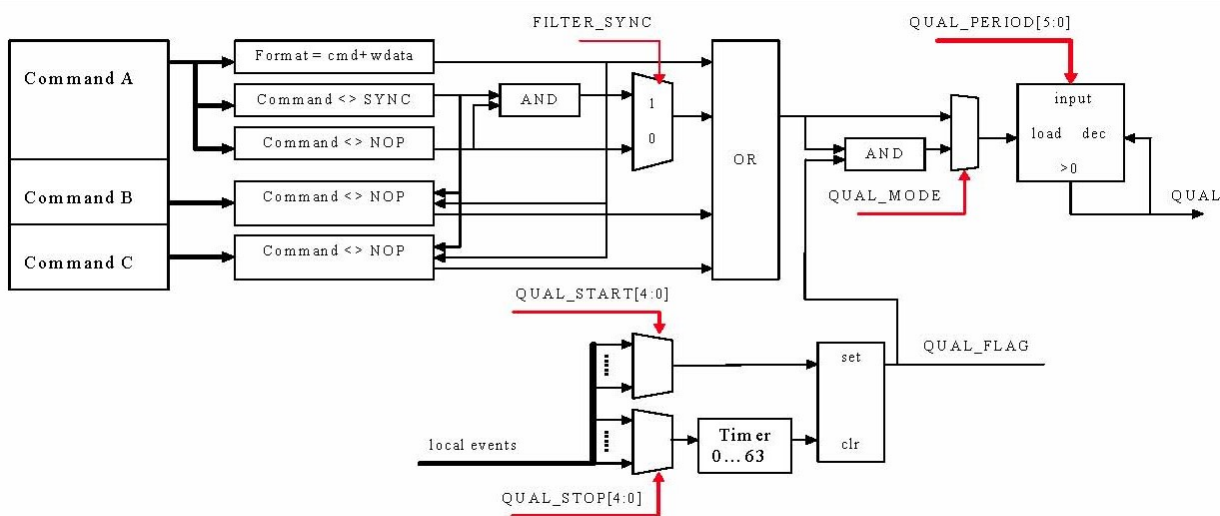


Figure 6-4 — Qualification Signal Block Diagram

### 6.3.4.8 Power Delivery

When systems add LAI card(s) during validation and debug experiments, additional power is needed to support AMB-LAI requirement. The components on the LAI board include the AMB and other active devices. Power supply is needed for both the FBDIMM and DDR interface. The options for the power supply are a) Draw power from system voltage regulators or b) design voltage regulators on the LAI board. However, most system power solution do not have enough margin to accommodate the additional LAI requirement. It is therefore recommended to have an external power supply. It is recommended to have voltage regulators on the LAI board. It may be required to support more than one voltage regulator as the AMB requires at least two different voltages to support different IO and core circuitry.

### 6.3.4.9 Clocking Input

Configuration of the LAI mode registers in the AMB requires that the system clock is enabled before LAI programming can begin. The clock for the AMB-LAI must be referenced from the system clock synthesizer or oscillator, essentially to eliminate any frequency drifts. As no additional clocks are available for the AMB-LAI, a clock buffer is recommended. The buffer is placed on the AMB-LAI card.

In the LAI configuration, the clock from the DIMM connector is input to a clock buffer. Clocks from this buffer drives the AMB on LAI card and the displaced DIMM. The clock buffer on the LAI card must be selected to match the properties of the system clock buffer.

### 6.3.4.10 Local Events in AMB

Trigger and mask/match features are being designed in the AMB\_LAI to generate local events pool. Along with these, global events propagated through in-band debug and external trigger events (EV) will add events to the pool. Possible events are listed below along with an illustration of the select logic.

**Table 6-3 — Local Event Pool**

Name	Potential Cases	Comment
Mask and Match	3	Combination and Muxed Pattern recognitions
In-band debug	8	Features TBD with host interface
Event bus inputs	4	EV[3:0]
Others	17	8 initialization states, 1 qual_flag, 7 error indicators, Null event
Total	32	Total to be limited to 32

The mask/match features in the AMB\_LAI, and certain internal state and error conditions, will be used to generate local events. Global events propagated through the in-band debug and external event (EV) bus will also generate local events. All of the local events can be selected by muxes as trigger sources for LAI event signals and event bus signals. While not used in LAI mode, these events are also used in error injection and for sourcing events in the NB status frames. For exact details see Chapter 12, Configuration Registers.

Table 6-4 shows the 32 local events. Each of these local events is logged in a configuration register and is sent to 15 32:1 muxes. The select lines for these muxes are programmed in configuration registers. The final table shows the destination (intended use) of the 15 selected events.

**Table 6-4 — LAI Local Events**

Name	Events	Sel Addr	Description
Mask and Match	3	11:9	MMEVENT2:MMEVENT0 Slot a, Slot b, Slot c, and Frame command matches - 3 events preselected from 13 possible matches
Inband debug	8	23:16	Received on sb link in-band EV[7:0]
EVBus events	4	15:12	Received on select DDR pins Event Bus EV[3:0]
Initialization States	8	1:8	Disable[1], calibrate[2], training[3], testing[4], polling[5], config[6], I0[7], I0s or recalibrate[8]
Errors	6	30:25	SB/NB failover[25], when unmasked: <ul style="list-style-type: none"> <li>• SB crc error[26],</li> <li>• thermal overload[27],</li> <li>• clock training violation (&lt; 6 transitions in 512 UI) [28],</li> <li>• unimplemented register access[29],</li> <li>• other implementation specific errors[30]</li> </ul>
Qual	1	24	
Spare	1	31	
NOP	1	0	Null Event
<b>Total Events</b>	<b>32</b>		

#### 6.3.4.10 Local Events in AMB (cont'd)

**Table 6-5 — LAI Event Selection**

Name	Events	Description
Output event/triggers	11	Sent to LA on DDR pins
EVBus events	4	Sent on DDR pins
Inject Event NB	1	Assert nb event bit
Error Injection Trigger	1	Inject errors
Qual Events	2	Start and Stop events for qualification signal
<b>Total Events</b>	19	

#### 6.3.4.11 In-Band Debug Events on FBDIMM Southbound Links

There is a need to pass debug events between FBDIMM host agent device and the DIMMs chained on the link in order to support timely interaction between the internal debug. For the Southbound direction (from Host Interface to DIMMs) the primary function is to augment triggering mechanisms in AMB. It also supports event-stimulated operations in DIMM buffers, such as injection of selected errors so that error detection logic can be validated/tested. This feature does not add pins to agent devices or require any additional routing on any board.

Internally, modes for device specific debug event transmission and interpretation are implemented using local control registers configured by out-of-band SMB. This feature is expected to be implemented on devices other than AMB to provide broader coverage and sharing of debug events between different devices in a system. FBDIMM Host and DIMM buffers shall broadcast debug event messages with a standardized extension for the FBDIMM protocol using an added channel command, called a Debug Event Packet (DEP).

#### 6.3.4.12 Event Signaling

The AMB LAI mode enables four event signals (EV[3:0]) to be shared between AMB devices in a system. The signals are shard through a uniquely defined interconnect that connects all the devices to the 4-bit wide daisy chain bus. The 4 lanes are independent and carry separate events or triggers. There are three types of events, as shown in Table 6-6.

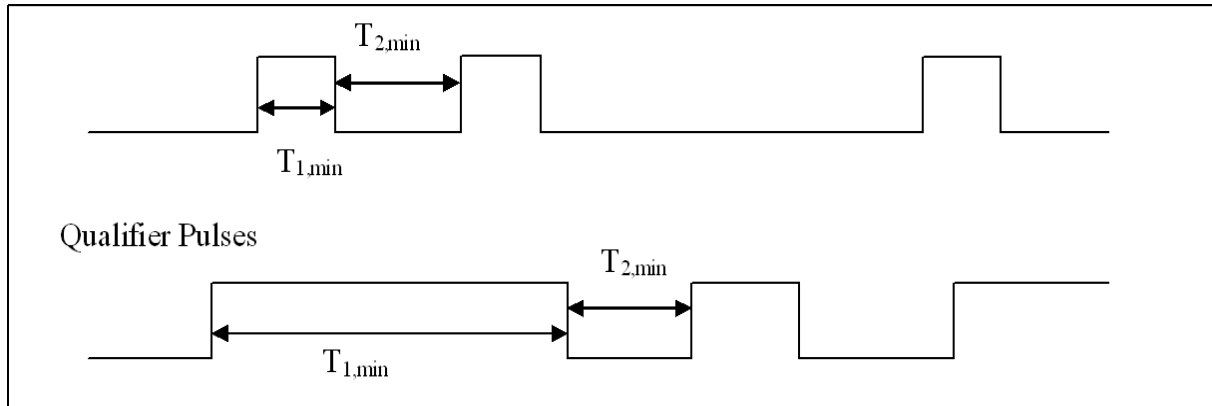
**Table 6-6 — Different Event Signal Types**

Type of Event Signal	Signal name	Sources on the Bus	Type of Trigger
1	Event Drive Pulse	1	Edge
2	Global Trigger Pulse	Any	Edge
3	Global Qualifier Level	1	Level

The event bus receiver must lock a transition (edge) and not change states after the first transition is captured. The event drive must return to its normal state (de-assert) within  $T1, \text{min} = 6$  clock cycles. The next event can be asserted  $T2, \text{min} = 16$  clock cycles after the de-assertion of the event pulse. For a qualifier event, the pulse width is defined by  $T1, \text{min}$ . The next qualifier event may be asserted at least 16 clock cycles after de-assertion, to accommodate potential ringing in the event bus. Figure 6-5 shows the event signal timing.

### 6.3.4.12 Event Signaling (cont'd)

The data has been derived from simulation of an event bus simulating wired-or configuration using 3 state buffer models.

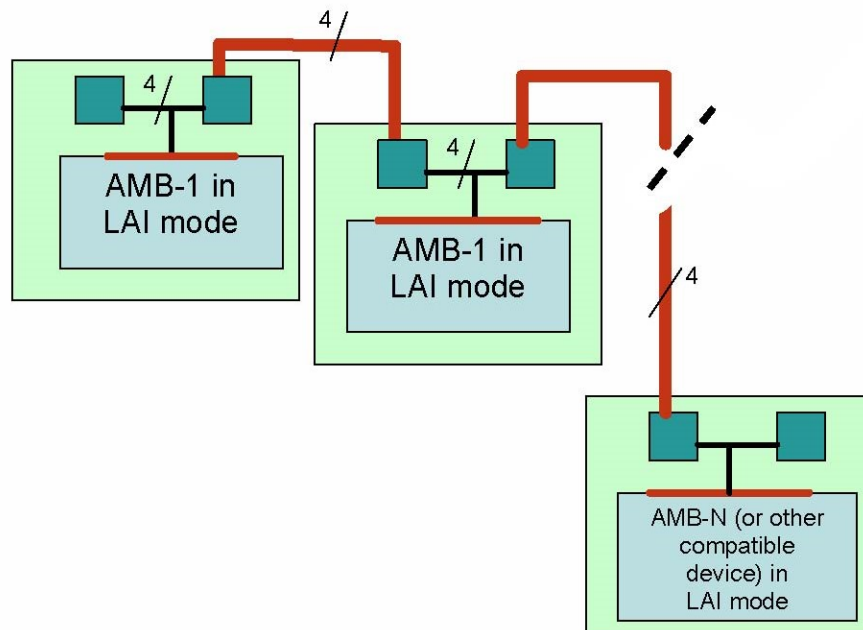


$T_{1,min}$  = 6 clock cycles (this parameter is defined by the signal quality at the rising edge)

$T_{2,min}$  = 16 clock cycles (the signal quality at the falling edge has ringing that takes time to settle)

**Figure 6-5 — Event Signaling Timing Characteristics**

The signals will be driven or captured by four DDR IO buffers. The topology below demonstrates how the event bus connects the AMB and similar devices. The signal integrity on the event bus and limitations of the EV[3:0] IO defines the length of the cable and the number of AMB devices that can be daisy chained together.



**Figure 6-6 — Event Bus Topology**

### 6.3.4.13 Logic Analyzer Interface Topology

The DRAM interface of the AMB components will be used to connect to the logic analyzer. The AMB component must ensure that LAI mode signals driven by the mapped DDR I/O buffers operate at full DDR data rate as necessary. The topology of the Logic Analyzer Interface must also be correctly modeled and designed to accurately read data driven by the DDR IO. Only one differential clock signal will be sent to the logic analyzer. These signals are sent as an asynchronous group of data to the logic analyzer. The topology must incorporate appropriate termination schemes to ensure the signal integrity of the LAI signals. Figure 6-7 is an abstract topology of the LAI.

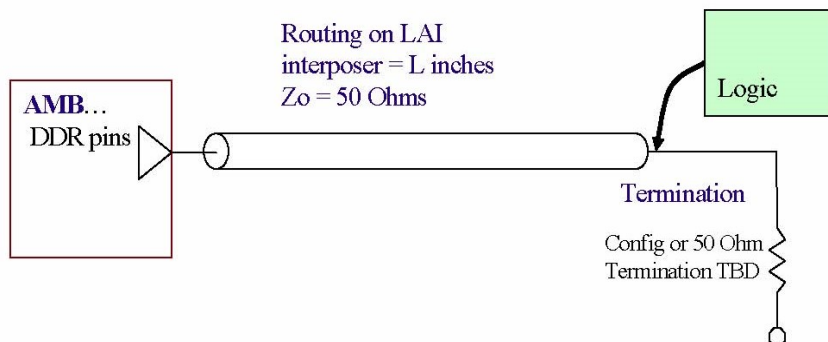


Figure 6-7 — Probing the AMB-LAI Signals

### 6.3.4.14 Definition of LAI Mode Signals

Table 6-7 — LAI Signal Definitions

Signal Group	Direction	Definition
CLK[p,n]	Out	Logic analyzer reference clock (differential). This provides a LA compatible timing reference clock for capture of all signals to the LA.
N[83:00]	Out	Northbound demuxed (6x14) traffic (MODE=0), link state (MODE=1). This is the actual 1:6 demuxed link traffic, or alternately the link state (when traffic is not valid).
QUAL	Out	Store qualifier: 1 = store, 0 = do not store
MODE	Out	Southbound link mode: 0 = demuxed traffic, 1 = encoded link state
FRAME	Out	With an external pulldown, FRAME = 0 when the DRAM interface is tri-stated. This is used by LA as a qualifier to mask the rest of the LAI and derived signals. FRAME = 1 indicates stable FBDIMM traffic.
S[59:00]	Out	Southbound demuxed (6x10) traffic (MODE=0), link state (MODE=1)
TRIG[10:0]	Out	Triggers to LA (shared between N & S). This provides eleven unique trigger signals to the LA from the AMB LAI as result of frame pattern matching, state matching, and/or cross-triggering events from other LAIs. This allows cooperating AMB LAI to overcome (a) limitations of LA in recognizing serial traffic patterns that exceed LA pattern matching capabilities and (b) relatively long latencies in cross-triggering between LA modules.
EV[3:0]	IO	Inter-AMB event bus for cross-triggering. This four bit event bus allows multiple AMB LAIs to be programmed to inter-communicate locally detected matching and/or filtering opportunities events (cross-triggering and cross-qualification). These signals must be designed to drive across cables and clocking domains with adequate timing to drive trigger sequences and trace qualification with better timing than can be accomplished by LA.

## **6.4 LAI Requirements**

### **6.4.1 Capture all Southbound and Northbound Traffic**

LAI mode buffer shall normally be installed between the North most DIMM and the host device where it can monitor all Southbound and Northbound traffic. It shall act as a pass-through repeater for link traffic while monitoring and transmitting both directions to a logic analyzer.

### **6.4.2 De-multiplex Captured FBDIMM Channel Traffic**

Captured Southbound and Northbound traffic shall be de-multiplexed to a reduced rate equal to 1/6<sup>th</sup> of the FBDIMM channel rate prior to transfer out to a logic analyzer.

### **6.4.3 Drive Link Traffic with Framing Signal to Logic Analyzer**

De-multiplexed Southbound and Northbound traffic shall be retimed to a common frame timing such that first half of both directions of traffic are aligned, although with variable number of frames offset between Southbound command frames and corresponding Northbound frames. Frames of demuxed traffic, a start of frame signal, and several additional derived trigger and qualification signals shall be driven to the logic analyzer using remapped DRAM pins of the AMB. A single clock shall provide timing reference to the logic analyzer for capture of the signals.

### **6.4.4 Detection of Debug Events**

Debug events used in the AMB LAI come from several sources, internal and external to the device, as defined in the following clauses.

#### **6.4.4.1 Pattern Match on Commands/Frame**

AMB LAI shall simultaneously check for pattern match on, using independent bit level mask and value registers, for any of three DRAM/channel command patterns in any of the three possible command positions of a frame. The result shall be individual local events for each pattern in each unique command position (3 x 3 = 9 total), each pattern in any of the positions (3), or the three command simultaneously in the three sequential positions respectively, corresponding to a single full frame (1). These events are then input into three selectable 14:1 multiplexers. The result shall be 3 unique events provided to the local event pool. Note that matching of each pattern can occur during different frames so that sequential or widely spaced commands can be detected. Only for the case of the full frame match are the pattern matches required to occur in the same frame time.

AMB LAI pattern mask and match events can be selectively qualified by the type of commands in the frame compared. Match and mask bits 39 in each of the slots A, B, and C are used in defining the type of commands or data being matched. The cases are as follows:

- 1) When mask[39] is 1 for particular match/mask set, the match will be to raw data and will not be decoded to determine data or command content of the slots.

#### 6.4.4.1 Pattern Match on Commands/Frame (cont'd)

- 2) With mask[39] = 0,
- a) programming bit match[39] to 1 will qualify mask/match events for that set only when the frame has command in slot A and data in slots B and C.
  - b) programming bit match[39] to 0 will qualify mask/match events only when the frame is a full frame command with FS0=FS1=0 (excepting full frame frames without true commands in slots B and C, i.e., SYNC and SOFT RESET). A slot C command match is also qualified by slot B write config command match.

**Table 6-8 — Setting Bit 39 in Match/mask Registers to Qualify Cmd/Data Match**

Mask[39]	1	0	0
Match[39]	X	1	0
Slot A	Any	Cmd	Cmd
Slot B	Any	Data	Cmd
Slot C	Any	Data	Cmd

#### 6.4.4.2 Decode Received Southbound In-Band-Events

In-band debug event commands shall be recognized by the AMB in LAI mode, and the eight transported event bits shall be translated into local events for further use. Disable the decode of in-band event commands in slots B and C when either of these slots are data instead of commands.

#### 6.4.4.3 Received EV Bus Signaled Events

The four EV bus inputs are digitally filtered (fast assertion, de-glitched de-assertion), and converted into an assertion edge pulse for distribution as part of the local event pool. The filtering is required to tolerate ringing that may occur on the wire-OR'd EV bus without causing a single sourced assertion appear to be multiple assertions at receiving devices.

#### 6.4.4.4 Detect Southbound Errors

Errors occurring in the transmission formatting of commands/frames anywhere north of the AMB LAI shall be detected in the device to produce local events.

##### 6.4.4.4.1 Command CRC Error

AMB LAI shall monitor southbound traffic to detect any CRC errors in commands or data occurring in the formatting or transmission of commands/frames North of the AMB LAI. Detection of any error shall result in generation of a combined local CRC error event.

##### 6.4.4.4.2 Command Format Faults

AMB LAI shall monitor southbound traffic to detect any invalid command format errors in commands. This functionality is not required in the minimum set of AMB LAI features, although it may be useful to implement to enhance debug/validation coverage.



#### **6.4.4.5 Initialization State Machine Entry into Selected States**

A local event shall be generated any time the AMB LAI detects its local initialization state machine enter into a program selectable state.

#### **6.4.5 Event Response Mechanisms**

Local events shall be individually selected as stimulus for a set of event driven response mechanisms. Some of the responses result in transmitting occurrence of the selected event to a logic analyzer or onto an event bus between LAI. The other usage is in setting and clearing a flag used to provide longer term filtering of trace captured by gating QUAL signal assertion.

##### **6.4.5.1 Send Selected Local Events to EV Bus**

Any four of the local events (or none if individual event signals are not driven) shall be selected and then conditioned (stretched to insure a minimum pulse width) to drive the EV bus. This bus shall be used to communicate local events to other LAI devices.

##### **6.4.5.2 Send Selected Local Events to Logic Analyzer**

Any four of the local events shall be selected and sent to the LA as TRIG signals synchronous to demuxed link traffic. These signals can be used as either LA triggers or as markers in traces corresponding to the occurrence of the selected events.

##### **6.4.5.3 Produce Capture QUAL Signal for Trace Filtering**

Decode NOP, SYNC, and cmd+data commands in southbound direction adequately to perform logical combinations and generate a trace storage qualification signal to allow selective filtering in a logic analyzer.

Filter modes:

- No filtering (by LA ignoring QUAL signal)
- Filter out only frames containing only NOP commands
- Filter out frames with NOP and SYNC commands
- Filter out frames with NOP and SYNC commands or if event driven QUAL\_FLAG set.

Any traffic arriving in a southbound frame besides those currently being filtered out will cause assertion of a QUAL signal sent to the logic analyzer. The QUAL signal shall be de-asserted after a programmable time-out if only filterable traffic is seen. Otherwise the QUAL signal remains asserted and the time-out is restarted. This feature causes the QUAL signal to be “stretched” starting from the point of detecting any active Southbound commands so that any resulting Northbound frames shall also be captured. The time-out shall be selectable from 0 to 63 frame times in order to adjust for total Southbound to Northbound frame round-trip delays. The variability is required due to variable number of AMBs that can be present in the chain South of the AMB LAI.

##### **6.4.5.4 BIOS Dependencies**

When the AMB-LAI is detected by BIOS the proper protocol must be followed BIOS. See the platform BIOS reference code for details.

---

## **7 Common DFx Features**

---

### **7.1 FBDIMM IO Test**

This clause provides requirements and guidelines for implementing the DFT to characterize and test the FBDIMM interface in production chips on available test equipment. While the equipment used for debug and characterization may operate at the FBDIMM data rates, Automatic Test Equipment (ATE) used in High Volume Manufacturing (HVM) is not expected to operate at FBDIMM data rates. Therefore, testing at a faster speed is possible only with DFT modes.

### **7.2 Interconnect BIST**

As component to component bus speeds increase and circuit boards get denser, testing high speed bus connections in an HVM system test environment is becoming increasingly difficult and in some cases impossible. Board level DFT features such as In-circuit test points have been eliminated on high performance buses (speeds greater than 200 MHz) due to board/component electrical issues.

The on-die Interconnect Built-In Self Test (IBIST) architecture is designed for system level bus testing in an HVM environment as well as during electrical verification and validation. The methodology replaces the need for invasive probing. The goals of the IBIST technology are to maintain/increase board test coverage levels, reduce platform test time, eliminate the need for invasive probing in electrical verification/validation, facilitate platform level test access and availability, and enable test reduction/elimination. Further, IBIST has significant benefits from a verification/ validation perspective as it provides a more deterministic way to test (i.e., improved/faster fault isolation, eliminates doubt as to where failures occur), reduces pattern development time, and enables true concurrent travel on buses.

#### **7.2.1 FBDIMM IBIST Architecture Specification**

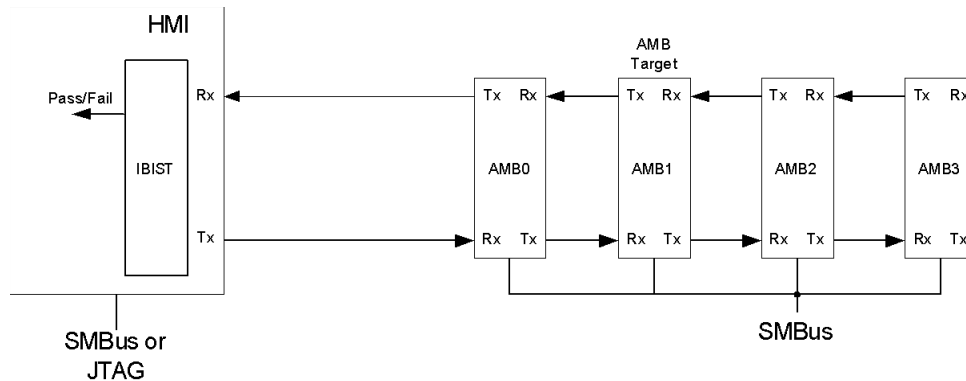
The basic premise of the FBDIMM IBIST architecture is to send stressful data patterns onto the FBDIMM link and check for errors. This operation must occur without requiring the higher level protocol to be involved. Bypassing the FBDIMM protocol allows extensive testing and debugging at the physical layer.

Because the FBDIMM protocol is asymmetric, IBIST operations differ depending on the usage model. Also, platform IBIST operations are slightly different than HVM component testing. Several configurations are available, each with a unique characteristic, depending on usage model.

In a platform usage model, the host memory interface is responsible for initiating the IBIST test sequence. This is due to the fact that IBIST is leveraging the training sequence for deterministic test entry and exit conditions. The TS1 state defines the entry point and exit point for the stress testing section of the training state machine. Byte 4 begins the test sequence for IBIST. At this point control is passed from the training state machine to the IBIST logic. IBIST requires a start delimiter to synchronize the checking logic. An exit from TS1 is defined with two sets of end delimiters (refer to the FBDIMM Architecture and Protocol spec). The start and end delimiters are a unique set of data values that are not likely to be of interest for stress testing but contain a near equal number of transitions to be training pattern friendly.

### 7.2.1 FBDIMM IBIST Architecture Specification (cont'd)

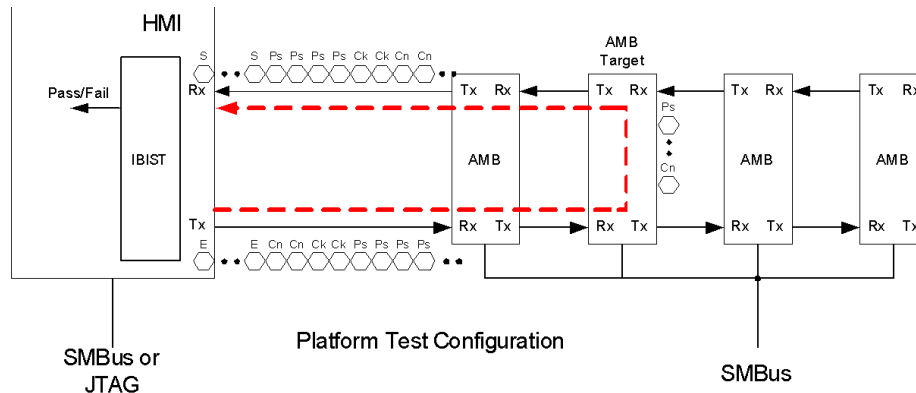
Shown in Figure 7-1 is an example of a common platform topology with a host memory interface (HMI) and the daisy chained Advance Memory Buffer components. Connections to the DDR memory are not shown in the figure but assumed to be there as part of platform. A target AMB is selected for testing by setting the AMB DIMM slot number in the training control register. The following example assumes the loopback model where the AMB reflects or echoes back data symbols to the HMI exactly as received. The concept of loopback applies to reflection of data symbols and not on the physical construction of the loop. As an example, for HVM component testing it is possible to physical connect the Tx and Rx pairs to each other to form a loopback path for data testing. The wire trace is not the loopback but rather the data be sent out is reflected back to the driving component.



**Figure 7-1 — Example of a Common Platform Topology for FBDIMM**

Executing IBIST on this platform topology is relatively straight forward because the training state machine coordinates the entry and exit conditions. Once directed to begin the IBIST engine is self-contained to deliver and check symbol stress patterns.

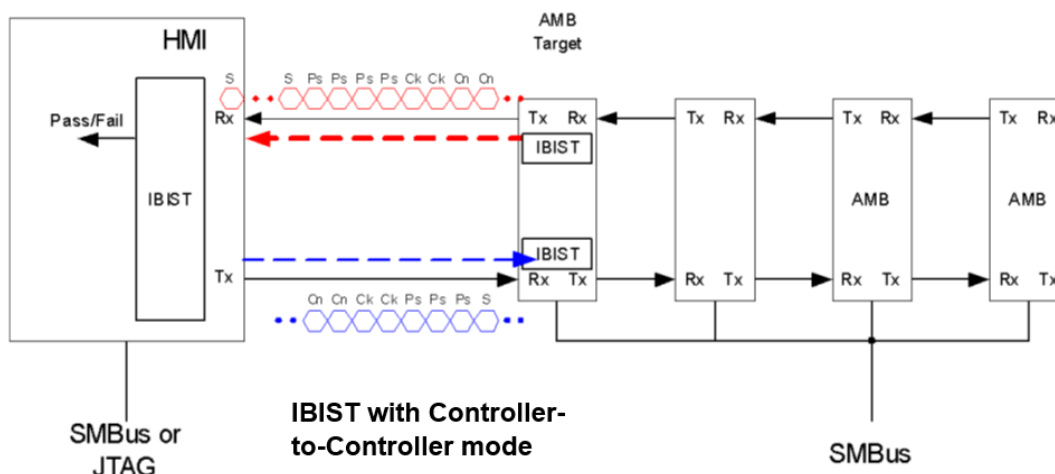
After the training proceeds to TS1 state a transfer of control occurs from the training state machine to the IBIST state machine starting with byte 4. Figure 7-2 is a simple example of the IBIST test patterns looping through the selected AMB back to the HMI. For this platform usage model the HMI generates and checks the symbols. Each AMB under test only re-synchronizes the data stream back to the HMI. The IBIST in the AMB is not used in this model. Results from the IBIST operation are obtained from the registers within the HMI.



**Figure 7-2 — IBIST Loopback Operation on a Selected AMB**

### 7.2.1 FBDIMM IBIST Architecture Specification (cont'd)

Another platform testing usage model enables the controller-to-controller (C2C) mode. This model uses the IBIST engines in both HMI and AMB components. Figure 7-3 shows the two IBIST symbol streams originating from both components. This is an important feature for link characterization or debugging since the northbound lanes are a different width than the southbound. The previous loopback model replicates lanes [4:0] across the northbound link. If a specific pattern is required for debug, the C2C mode can apply un-replicated unique symbols across the link where the loopback mode cannot. The final results are extracted from both components using SMBus (the HMI component may also have JTAG as an option).



**Figure 7-3 — IBIST Controller-to- Controller Mode Operation on a Selected AMB**

IBIST is used for component level testing as well. An AMB can be made to act like a host by setting the IBISTINITEN bit in the FIBINIT register along with the MASTRMD bit in the FIBPORTCTL register. This will cause the AMB to send TS0 and TS1 sequences as needed and pass control to the IBIST TX engine at the appropriate time in TS1 to transmit IBIST patterns.

Since all FBDIMM IBIST operations are based on programming IBIST control registers, a consistent user interface must include identical register definitions of the feature presented here. This insures different IBIST implementations have the same user interface allowing a common software tool to access any FBDIMM component. Table 7-1 lists all basic IBIST features.

## 7.2.1 FBDIMM IBIST Architecture Specification (cont'd)

**Table 7-1 — Summary of IBIST Features**

	Feature	Function and Usage Model
IBIST Operating Modes	Fixed Mode	Fixed mode executes a fixed set of patterns. The pattern generator operates with the default value set in the registers. The training state machine indicates when IBIST can run (TS1).
	Open Mode	Open mode allows user to define the IBIST patterns and its run format.
	Continuous Mode	When the continuous loop bit is set the sequence of patterns from all the sub-types (pattern buffer, clock pattern, or constant value pattern) is executed indefinitely. The user can exit from continuous mode by clearing the bit.
	Compliance Measurement Mode	When CMMSTR bit is set, the IBIST TX engine will take control of the physical layer and start transmitting IBIST pattern. The link layer will be disconnected from the physical layer. The user can exit from this mode by clearing this bit.
IBIST Controls / Status	Enter IBIST mode	All implementations must provide a means to enter IBIST mode. Upon entering IBIST mode, FBDIMM physical layer is disconnected from the link layer. The training state machine takes over and sets up the conditions that allow for control to be turned back over to the IBIST controller.
	Exit IBIST mode	The training state machine defines the entry and exit conditions. When IBIST is complete, control is turned back over to the training state machine where the end delimiter is sent indicating that TS1 is complete.
	Loadable pattern memory	User-defined patterns can be loaded into the pattern register any time IBIST is not executing.
	Starting and stopping IBIST	<p>There are two methods of starting IBIST. The first is every time the training sequence is executed. The default patterns defined in the pattern generator will launch an IBIST pattern sequence during the TS1 state of the training sequence. A user defined start is caused by setting the start bit in the IBIST Control register (FIBPORTCTL.IBSTR). This will force the training state machine back into idle and begin a new training sequence. After byte 4 of TS1 is sent, the training state machine passes control to IBIST to begin sending patterns.</p> <p>Stopping IBIST is automatic if the loop counters are used. If loop continuous is enabled the patterns will be continuously sent on the link. The start bit should be cleared to stop pattern transmission. Note that IBIST will not stop immediately. IBIST will complete transmission of the current sub-pattern loop count and the receiver will complete comparison of the current sub-pattern loop count. Each sub-pattern has its own loop counter. One loop count equals two FBDIMM frames of 24 bits.</p> <p>Another option is to clear the loop continuous bit. Clearing the bit does not stop the patterns but rather enables control of the counters to expire as normal where control is handed back to the training state machine and a delimiter is sent to complete the TS1.</p>

## 7.2.1 FBDIMM IBIST Architecture Specification (cont'd)

**Table 7-1 — Summary of IBIST Features (cont'd)**

	Feature	Function and Usage Model
	IBIST Completion Status	All IBIST implementations must set “IBIST Completion Status” bit in the IBPCTL when IBIST run completes. NOTE An exception is when IBIST is configured to run in “Continuous” mode.
	Error Reporting	Error information per channel is captured for debug and diagnostic purposes. The error logs report the lane number(s) with errors.
	Error Counting	Error counting on a per lane basis improves the granularity by which a validation engineer can diagnosis problems with the FBDIMM interface.
IBIST Operation Support	Loopback	This is the basic operational mode of conducting an IBIST test. Because of the nature of FBDIMM the Host Memory Interface is not required to implement loopback but the AMB must.
	Controller-to-Controller	Due to the asymmetric FBDIMM interface, controller-to- controller allows independent testing of each uni-directional link. This implies one pattern generator for the Tx control block and one for the Rx. In AMB, the Southbound and Northbound need to have separate registers and independent pattern generators to enable this.
	Automatic Cross-talk Generation	This feature generates cross-talk traffic by inverting the data pattern for a particular lane. This causes the opposite data values on adjacent lanes.
	Pattern Inversion Sweep	By moving the cross-talk pattern across the link's transmitters forms a dynamic inversion to sweep across the bus for improved cross interference. The purpose is to introduce resonance variation across the width of the port.
	Transmitter Mask	Control is provided to individual transmitters to select whether they will participate in the IBIST run. If a transmitter is not selected, then it remains in electrical idle.
	Receiver Mask	Control is provided to individual receivers to select whether they will participate in the IBIST run. The bits mask off the error status for the lane selected.
	Secondary Pattern (Optional)	A secondary pattern set is allowed to direct more stressful patterns surrounding a particular lane.

### 7.2.1.1 IBIST Operating Modes

Selection of IBIST operating modes is done by programming bits the FIBPORTCTL register. A bit to select between Fixed or Open mode does not exist. By default, the IBIST block is configured to run a fixed number of patterns from the pattern generator. This is what is called Fixed mode. Once the user customizes the pattern generator with varying number of pattern types and loop counts this is considered Open mode. A continuous operation mode is enabled by the loop continuous bit (bit 3 in the FIBPORTCTL). If it is cleared then the IBIST pattern set is limited to the quantities defined in the loop counts of the various pattern types.

### 7.2.1.2 Loopback

Loopback is the default mode of operation as defined in the FBDIMM architecture and protocol specification. When the IBIST start bit is set, the controller sends test patterns within the TS1 packet and the target reflects back the packets to the controller as defined in the specification. Checking is done in Host Memory Interface.

### 7.2.1.3 Controller-to- Controller

A second testing paradigm is available that allows an IBIST in each component (HMI controller and AMB target) to transmit and check the IBIST patterns in a unidirectional link (i.e., HMI Southbound Tx to AMB Southbound Rx). This capability takes advantage of the fact that an IBIST test sequence is framed with start and end delimiters and the existence of IBIST engine in each component.

To initiate the operation, the IBISTR bit is set in target AMB's SBFIBPORTCTL register and in Host's NBFIBPORTCTL register. The initial is at ion engine is enabled in the target AMB by setting the IBISTINITEN bit in the target AMB's NBFIBINIT register. Then the MASTRMD and IBISTR bits are set in the Host SBFIBPORTCTL register and target AMB's NBFIBPORTCTL register. It is assumed that host will always send init patterns as described in the Architecture and protocol specification document before transmitting IBIST patterns.

Once the testing phase of TS1 (byte 4) is entered, each component launches the IBIST test sequence. These will happen at different points in time. The IBIST receiver on the Host NB side will check the patterns transmitted by the target AMB NB IBIST TX engine and the target AMB's SB IBIST RX engine will check the data transmitted by the Host SB IBIST TX engine.

### 7.2.1.4 IBIST Entry and Exit

Descriptions for entering and exiting IBIST for the different testing paradigms are presented here. As a general rule, IBISTR always enables the receive engine and the IBISTR along with MSTRMD enables the TX engine. IBITINITEN bit in AMB specifies if the init sequences are sent before the actual IBIST patterns.

For Host, setting FIBPORTCTL.IBISTR bit, should take the link into disable state and then transition to Training and testing states. For AMB, the behavior is slightly different.

**Table 7-2 — AMB IBIST Modes**

IBISTR	MSTRMD	INITEN	AMB
1	1	0	IBIST Transmit engine will start transmitting IBIST patterns starting with start delimiter without going through training sequence. Receive engine will look for start delimiters (but will never get it since the IO is not trained).
1	1	1	Same as above except that the transmission starts with TS0 sequence, followed by TS1 and IBIST patterns. The receivers will check received data.
1	0	X	No transmission happens from IBIST. IBIST receive engine is active and checks incoming IBIST data.

#### 7.2.1.4.1 Loopback

The following description indicates how the IBIST logic interacts with the channel training algorithm. If the components are part of a platform test then IBIST expects the AMB slot number of the component under test has been selected by firmware or external software through JTAG/SMBus.

- The link is forced into a disabled state with the assertion of the IBIST start bit in the HMI.
- The AMB target number bit field will indicate which AMB is the target of the IBIST loopback. During TS0 training sequence, the controller will set the target AMB id to this value.
- The last AMB will reflect all training packets sent by the Host as defined in the architecture specification document.
- Once the training state machine has progressed to TS1 on the host side, control is passed over to the IBIST TX engine for transmitting IBIST patterns.

NOTE For the AMB component in loopback mode the selection of which lanes are replicated are programmable in the FIBPORTCTL register in the Host. Either [4:0] or [9:5] will be replicated across the northbound lanes ([13:10],[9:5][4:0]. Since [13:10] is only 4 lanes then only 4 least significant lanes are replicated.

#### 7.2.1.4.2 Controller-to-Controller

This is a special mode where the IBIST in both components (HMI and target AMB), behave as if it is a controller sending IBIST data sequences. A C2C mode should be setup via SMBus. IBIST C2C mode cannot be initiated by hardware only. The default value is C2C mode disabled.

Due the asymmetric links and the controller-to- controller mode this implies independent unidirectional testing of the links.

Usage model highlights and core logic expectations:

- 1) For an HMI component, the IBIST is always a controller.
- 2) C2C usage with last AMB:
  - a) IBISTR bit is set in the SBFIBPORTCTL register in the last AMB. The IBISTINITEN bit is set in NBFIBINIT register.
  - b) The IBISTR and MASTRMD bits are set in the NBFIBPORTCTL register. The Host init engine is also started. At this point the NB IBIST engine will start transmitting init patterns on the NB interface followed by IBIST patterns at the appropriate time in TS1. The Host also does the same independently on the SB side. The receive engines on target AMB SB side and Host NB side check the data. The NB side can wait till the SB side has started receiving TS0 sequences before starting transmission as a controller on the NB side or it can be independent of the SB side entirely.
  - c) After TS1 byte 4 the SB IBIST on the target AMB checks the incoming IBIST patterns. The NB IBIST on the Host side checks the incoming pattern at the appropriate time in TS1.



#### 7.2.1.4.2 Controller-to-Controller (cont'd)

- 3) C2C usage with all AMBs except for the target AMB (Last): a. If the IBISTR is set in the Southbound IBIST control register and this AMB is not the target AMB then the SB IBIST will check the patterns (assuming the pattern generator is configured the same as all others) after byte 4 of TS1
- 4) C2C usage with only AMBs no HMI. (Useful in AMB to AMB validation without an HMI present):
  - a) The AMB that needs to emulate a host will have its IBISTINITEN, MASTRMD and IBISTR bits set on the SB IBIST registers.
  - b) The target AMBs and the other intermediate AMBs will be configured the same way as described in the sequences above.

Basic IBIST operation:

- An AMB target number must be selected in the AMB Target bit field of the Host Memory Interface before IBIST operations begin.
- Using a configuration write operation, set the IBIST start bit in the AMB target component that is expected to participate in link testing. Configuration may use SMBus or in-band config accesses.
- A write to the HMI (IBIST controller) start bit begins IBIST operations. The link is forced into disabled state and proceeds with the training sequence.
- The training state machine sends TS0 packets with the AMBID set.
- Training progresses according to the channel spec until the TS1 packet. Starting with byte 4 (of TS1) the test patterns that are defined in the packet are generated by the IBIST transmit engine. The number of IBIST test frames is not defined but the pattern should not violate the minimum transition density.
- When the target receives the TS1 packet, it checks the test pattern portion of the packet against the expected pattern generated by the IBIST receive pattern generator. The target loops back the data to the host as specified in the protocol specification document.
- The controller checks the test pattern returned against the expected pattern generated by its IBIST receive pattern generator.

When the pattern set is complete the pattern generator (or IBIST main state machine) will pass control back to the training state machine where it will send two sets of two end delimiters.

#### 7.2.1.5 Delimiter Description

It is necessary to frame the IBIST payload to give the control block time to synchronize the checking logic with the incoming data streams. The start and end delimiters are selected to be as non-IBIST as possible. Since the patterns most likely to be used during testing are of the ISI, high frequency, and resonance beat patterns variety, a delimiter comprised of regular data values would appear as non-invasive as possible. A set of four frames composed of 2 sets of 2 frames will determine the start and end delimiters. The start delimiter is composed of two frames of binary values. The first frame is 3, 4, and 5 hex and the second is 6, 7, and 8 hex. This sequence is repeated once more for the second set of start frames.

### 7.2.1.5 Delimiter Description (cont'd)

The values selected are such that the number of 1s and 0s are nearly equal to maintain edge density before the stressful IBIST patterns start. Also, because the numbers are sequenced this reduces the checking logic complexity (i.e., a simple counter could be the reference). The end delimiter is reverse order as the start delimiter. The delimiter values are shown in Table 7-3. The first bit sent for the start delimiter is bit 0 of the digit 3 (right-most bit in table). Similarly, the first bit sent for the end delimiter is bit 0 of the digit 8 (right-most bit in table). The end delimiter is already defined in the FBDIMM Channel spec. When the IBIST engine has completed its payload, the control block instructs the training state machine to send the two sets of end delimiters.

**Table 7-3 — Start and End Delimiter Description**

Symbol	Frame 4			Frame 3			Frame 2			Frame 1		
Start	1000	0111	0110	0101	0100	0011	1000	0111	0110	0101	0100	0011
	8	7	6	5	4	3	8	7	6	5	4	3
End	0011	0100	0101	0110	0111	1000	0011	0100	0101	0110	0111	1000
	3	4	5	6	7	8	3	4	5	6	7	8

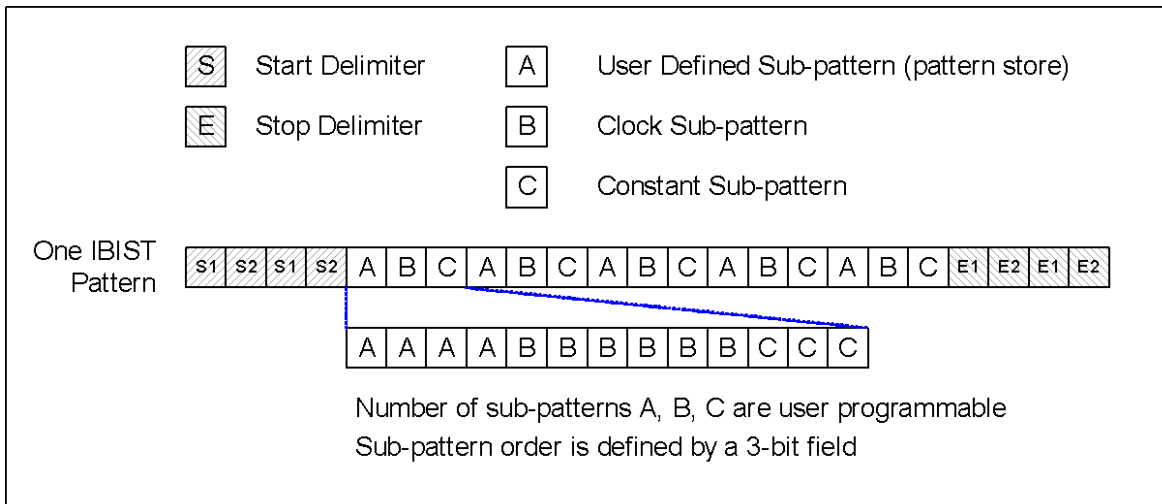
### 7.2.1.6 Access Mechanisms

All FBDIMM IBIST registers must be accessible by software through normal configuration reads and writes and out-of-band configuration mechanisms. It is possible to conduct IBIST testing with in-band configuration reads and writes only in the Host Memory Interface. A loopback usage model makes it possible to operate completely within the HMI component.

### 7.2.1.7 Pattern Definition

The pattern generator produces a mixture of patterns to provide the user with flexibility in developing inherently longer patterns than the 2 frames offered by the pattern buffer. Shown in Figure 7-4 is an example of rotating through 3 sub-pattern sets and looping on the whole set N number of times. The start and end delimiters frame the entire IBIST test. The pattern generator register contains bit fields to program the type and loop length of each of the sub-patterns. If a pattern is not required either an enable bit is cleared or the loop count is set to zero. If all the sub-patterns are disabled the IBIST still goes through the initialization sequence and sends the start and end delimiters however, there is no pattern payload.

### 7.2.1.7 Pattern Definition (cont'd)



**Figure 7-4 — FBDIMM IBIST Pattern Definition**

### 7.2.2 Reference Architecture

This clause provides a reference design that satisfies the FBDIMM IBIST architectural requirements. It is assumed that one IBIST engine is instantiated per FBDIMM port in a device. This provides greater flexibility and helps simplify the software tool support understand a port level of operation. A port has a slightly different definition depending on the component. For a HMI component one port (it is called a channel in FBDIMM spec) would be composed of 10 southbound Tx lanes and 14 (or 13) northbound Rx lanes. A port definition for an AMB will be the southbound Tx and Rx lanes and the northbound Tx and Rx lanes.

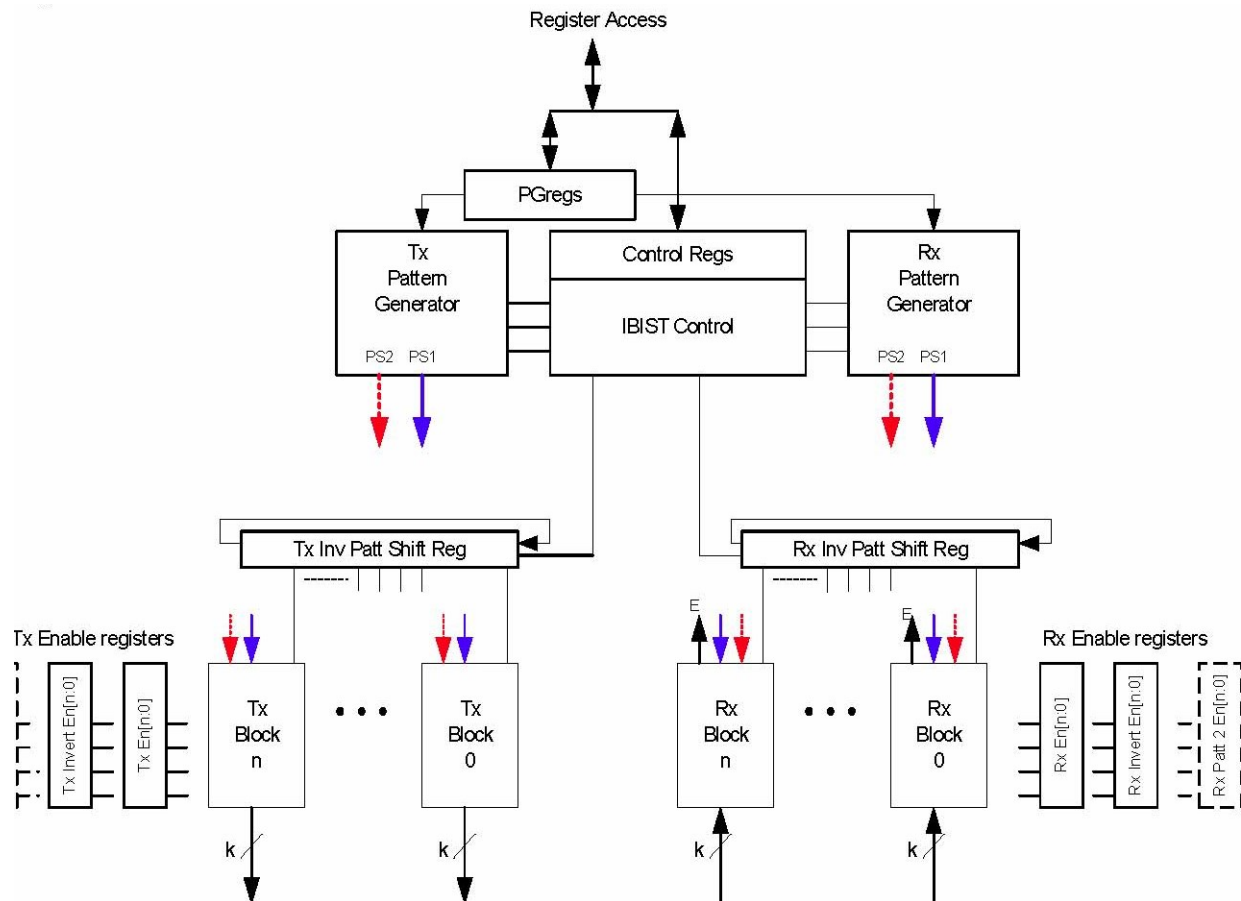
The reference design for an AMB has one pattern generator for each Tx and Rx port. This design consists of the following components:

- IBIST Control – provides the user interface and coordinates the IBIST operations.
- Pattern Generator – generates the IBIST pattern.
- Transmitter Block – muxes in the IBIST pattern into the data path of the regular FBDIMM traffic.
- Receiver Block – that manages receiver data and provides error detection.
- Inversion shift registers – to control a rotating pattern of inverting data patterns on selected lanes.

The control block contains all of the synchronization and control logic necessary to initiate an IBIST operation and coordinate the patterns to the transmitter and receiver checking logic. The Tx pattern generator develops patterns based on settings in the port control and pattern registers. The Tx block passes the data to the physical layer. The logic within the transmitter block is simple; the logics main function is to dynamically switch between Pattern Set 1 and Pattern Set 1 Bar. This is accomplished with a shift register configured to rotate the most significant bit (MSB) to the least significant bit (LSB). Once the patterns are looped back to the receiver a second pattern generator provides the reference by which the incoming data is checked against. A single bit deviation from this reference results in an error.

## 7.2.2 Reference Architecture (cont'd)

The purpose of the second pattern generator is twofold; first it helps to reduce routing congestion (this depends on several factors), second, it provides an independent test to run on the receive link for a controller-to-controller (C2C) IBIST operation. With this pattern generator and the C2C mode it is possible to run a different set of pattern tests than what is occurring on the transmitter for unique topological signal integrity problems.



Extended Feature Set

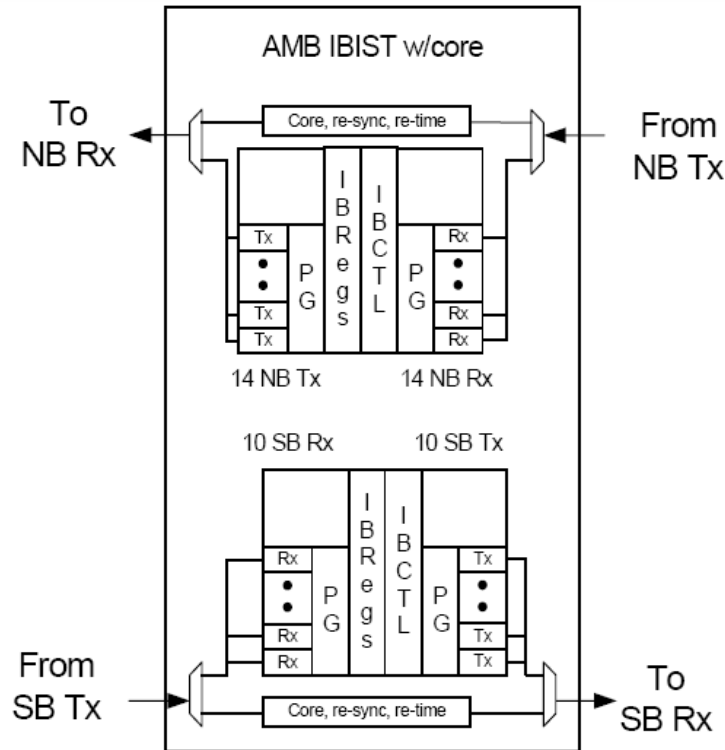
NOTE 1 Bit transfer width (k) to/from physical layer.

NOTE 2 One IBIST is instantiated per port in the AMB. Example, IBISTSB is for Southbound (SBTx and SBRx). Another IBISTNB is instantiated for the Northbound lanes NBTx and NBRx.

**Figure 7-5 — AMB FBDIMM IBIST Architecture**

The IBIST engine's Tx and Rx block operate at a k-bit granularity, where k is the ratio of link clock frequency to IBIST engine's clock frequency (typically same as link layer clock domain). For the first AMB component this ratio is 12. So the bus widths in these diagrams refer to 12-bit wide data to transfer information to and from the Tx and Rx blocks. Thus, the Tx block of IBIST engine drives k-bits of pattern every IBIST clock cycle, and Rx block consumes k-bits every IBIST clock cycle. Alternatively, if the Physical Layer and IBIST engine are driven from the same clock, the Tx (Rx) block would generate (consume) 1 bit for every phase of the clock, requiring only 1-bit data routes. Details on this buffering scheme are implementation dependent, and not shown in Figure 7-5.

## 7.2.2 Reference Architecture (cont'd)



**Figure 7-6 — AMB IBIST Instantiation Diagram**

### 7.2.2.1 IBIST Control

The IBIST control block provides a user interface and coordinates all IBIST operations. Values in the IBIST registers define how IBIST executes. Accesses to the IBIST registers are provided in band or through the SMBus interface to the IBIST control.

### 7.2.2.2 Entering IBIST Mode

The IBISTR bit is set, the IBIST receive engine is enabled. If MASTRMD is set along with IBISTR, the transmitters will get their inputs from the IBIST pattern generator. The receiver will direct their outputs to IBIST engine for error detection.

To enable Loopback mode of IBIST operation in a platform test, set the Start bits in the Host Memory Interface (HMI) for a system test.

To enable the Controller-to-Controller (C2C) mode of IBIST operation, it is necessary to first start the receive engine on both sides before beginning with IBIST transmission on the controller sides. Whenever the C2C model is used, a controller and target component must be identified. Usually C2C is used on a platform and the HMI is designated as the controller on the SB side and the target AMB as the controller on the NB side. For AMB only testing, one of the southern AMBs can be configured to be the controller on the SB side. This method is also used for component level testing. A data symbol loopback is established by connecting the Tx and Rx pairs together on a tester load board.

### **7.2.2.3 Exiting IBIST Mode**

After the IBIST payload is sent, the IBIST control block instructs the training state machine to send two sets of two end delimiter frames. When the last symbol is sent and IBIST instructed the training state machine to send the end delimiter, IBIST releases control of the link. The FBDIMM training state continues as directed in the channel spec during normal operation only in controller-target mode.

### **7.2.2.4 Starting IBIST**

Setting the IBISTR bit in the IBCTL register forces the training state machine to idle state. When the training sequence enters the test phase of TS1, IBIST executes the pattern generator according to the bit settings in the registers.

### **7.2.2.5 Stopping IBIST**

If the loop continuous bit is cleared then IBIST will stop automatically when the loop count expires. IBIST can be stopped manually by clearing the IBSTR bit. This is used when the LOOPCON bit is set for extremely long test sequences.

If the stop on error bit is set and an error occurs, the IBIST control state machine will stop operations and release control back to the training state machine. The normal TS1 exit condition is executed with two sets of end delimiters are sent on the link. An exception to this is the case where the transmit and receive engines are on different ports. An example of this would be the external loopback in an AMB where the output of the SB transmitters are looped to the NB receivers. In this case, the receiver will stop but the transmitter may continue to run till its loop counters expire.

When the IBISR receive engine receives the end delimiters or is stopped due to error or by clearing the IBISTR bit, it clears the IBISTR bit and sets the DONE flag in IBCTL register.

### **7.2.2.6 Transmitter and Receiver Masking**

During debug, it is desirable to mask off certain transmitters and / or receivers to isolate a problem. The FIBTXMSK and FIBRXMSK registers are provided for this purpose. Each bit position is equivalent to the lane number to which it is attached. Logic 1 in any bit position enables the lanes for IBIST operations. If the bit is a logic 0 in the FIBTXMSK register the transmitter will remain electrically idle (i.e. not sending any IBIST data) during an IBIST run. Similarly, when a bit in the FIBRXMSK is at logic 0 the receiver will disable any error detection logic for that lane.

### **7.2.2.7 Reporting Errors**

The FBDIMM IBIST stores the error logs of a report error from the checking logic in the RX blocks. When one of the RX blocks detects an error the IBIST control logic stores the status and the errant lane number in the global status register (SBFIBPORTCTL or NBFIBPORTCTL). The error status (ERRSTAT) indicates the type of error. An error lane number stored at ERRLNNUM identifies the offending lane that first reported the error. If multiple errors are identified at the same time, then the design picks one of the identifiers to be recorded in this field. Usually a priority encoding is used to force a deterministic decision but it is implementation specific.

### 7.2.2.7 Reporting Errors (cont'd)

The error count (ERRCNT) provides the total number of errors encountered during this IBIST run if the stop on error bit is cleared. If more than one lane reports an error in any frame, then each must be counted. This implies the error counter is an adder with a one-hot encoding to a binary converter. A frame is 12 bits wide and is minimum data size seen by the IBIST logic from each lane. Data is always compared a frame (12 bits) at a time and any mismatch in the frame is counted as one error by that lane. A specific implementation can use a higher granularity (less than 12 bits) for comparison and error counting.

It is very important to know which lanes report an error. Besides the first error, knowing which lanes have errors helps to identify IO pad or lane-to-lane interactions with stressful pattern sets. Adjacency issues cannot be known with a single error reported. Also, HVM testing will require an all fail condition because IO margin testing will be conducted in parallel. Therefore, it is a mandatory requirement that an error register collect all errors from the receivers. This register (FIBRXLNERR) contains bits from each receiver that remain sticky to identify which lanes are in error at the conclusion of an IBIST run.

### 7.2.2.8 Link Initialization Control

IBIST includes registers (SBFIBINIT and NBFIBINIT) to control timing of events during link initialization. Specifically, there are register fields to control the duration of disable state, TS0 and TS1. The SBIBISTMISC and NBIBISTMISC registers contain the AMB ID to be used during TS0 and the number of cycles to drive 1 during calibration (NB/SBIBISTCALPERIOD) Refer to the AMB architecture or register specification for further details on link initialization and register bit locations.

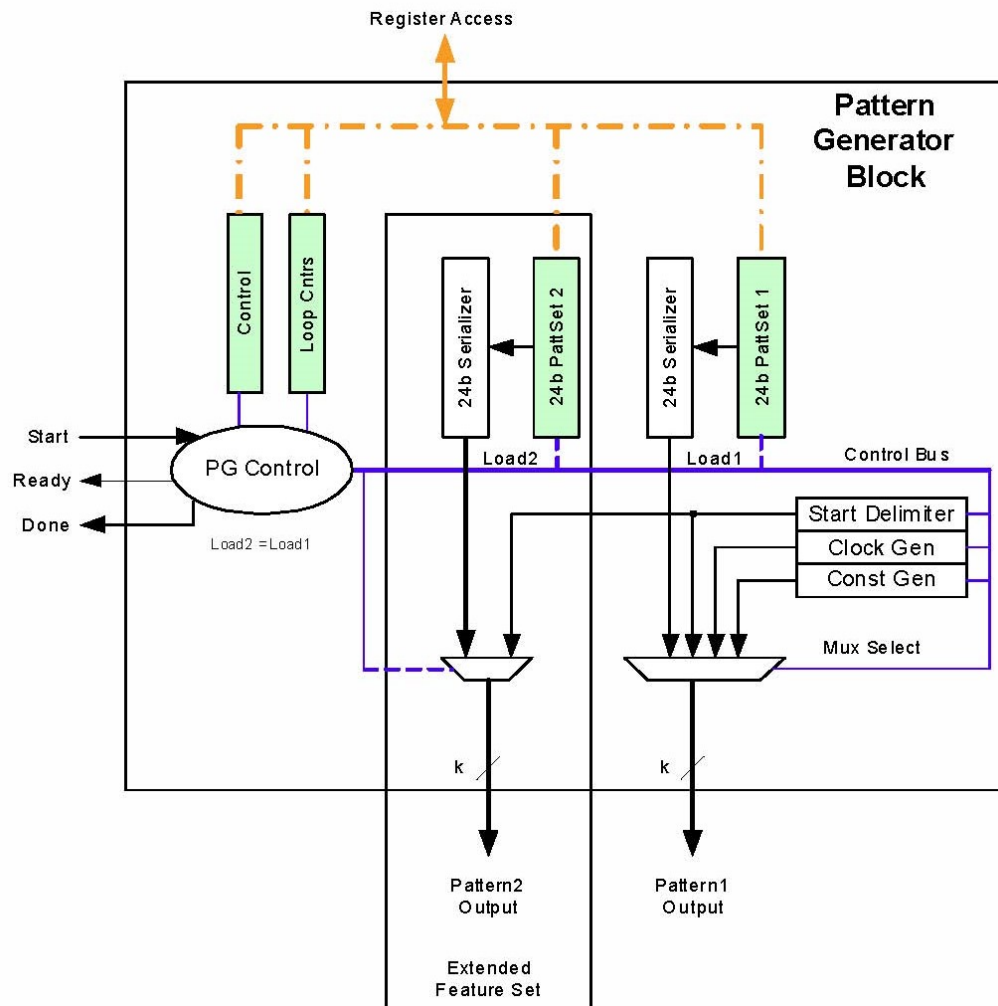
### 7.2.3 Pattern Generation

Pattern generation for the FBDIMM IBIST is based on a programmable combination of fixed and stored buffer patterns. The defaults settings fix the set of patterns to a specific set of sequences. These of course, can be overridden to needs of the validation or testing usage model. The basic building blocks of the pattern generator are shown in Figure 7-7. After the IBIST start bit is set and the training state machine passes control over to the IBIST engine, the pattern state machine sends the start delimiters and the test symbol payload. Pattern management is straight forward based on the bit settings in the control register. The order of the patterns defines which sub-pattern (pattern buffer, clocking pattern, or constant value) will be sent if the loop count contains a non-zero value. If the loop count for that sub-pattern is zero then it does not participate in the IBIST pattern set. If the loop continuous bit is set then the individual pattern loop counts are re-loaded and the entire sequence is ran again. The overall loop count is ignored when this bit is set.

After the pattern generator completes all sub-pattern loop counts and the overall loop count, the IBIST controller will release control to the training state machine where it will send two sets of two end delimiters before exiting from TS1. Once this happens the IBIST operation is complete.

The pattern generator outputs the IBIST patterns in k-bit chunks where k is the granularity of the data bus for transferring information between the core clock domain and FBDIMM physical layer.

### 7.2.3 Pattern Generation (cont'd)



**Figure 7-7 — Pattern Generator Block Diagram**

#### 7.2.3.1 Automatic Cross-talk Generation

A cross-talk pattern is induced by inverting patterns on a neighboring channel. A channel experiences the worst case cross-talk when both the left and the right neighbors are toggling exactly opposite itself. Also, a sweep of inversion patterns introduces dynamic conditions to the pattern set during run time. An automatic inversion (auto-inversion) feature is the only low cost feature that adds real-time alterations to the pattern flow that mimics real data scenarios on the link. Automatic cross-talk generation with sweep rotation is a required feature because it provides the only method to dynamically alter the pattern set on the lane while the IBIST is running.

The auto-inversion sweep feature is enabled with bit 5 of the FIBPORTCTL register. Clearing the bit allows a user to focus inversions on specific lanes for duration of the test. For inversion static stressing of the link use the FIBTXSHFT and FIBRXSHFT registers to specify each lane. The bit positions are a 1 to 1 correspondence to the lane number. If auto-inversion is enabled, the registers are shifted left at the beginning of each loop.



### 7.2.3.1 Automatic Cross-talk Generation (cont'd)

Figure 7-8 shows an example of how the auto-inversion feature works from the point of view of a Host Memory Interface's Southbound Tx lanes. During the IBIST run all lanes that are enabled for inversion will enable a mux that selects the inverted pattern set. For simplicity in this example only two lanes are enabled for inversion (FIBTXSHFT = 0x011), initially. With auto-inversion enabled, this register is constructed as a rotating shift register to continuously direct which lanes are inverting throughout the course of the IBIST test. The inversion pattern can be loaded in any configuration required for debug or characterization. The number of frames that the inversion is active on a lane is same as the pattern buffer counter.

It should be noted that since the AMB is asymmetric the loopback mode will take the data on lanes [4:0] and replicate them on northbound Tx lanes as [13:10][9:5][4:0].

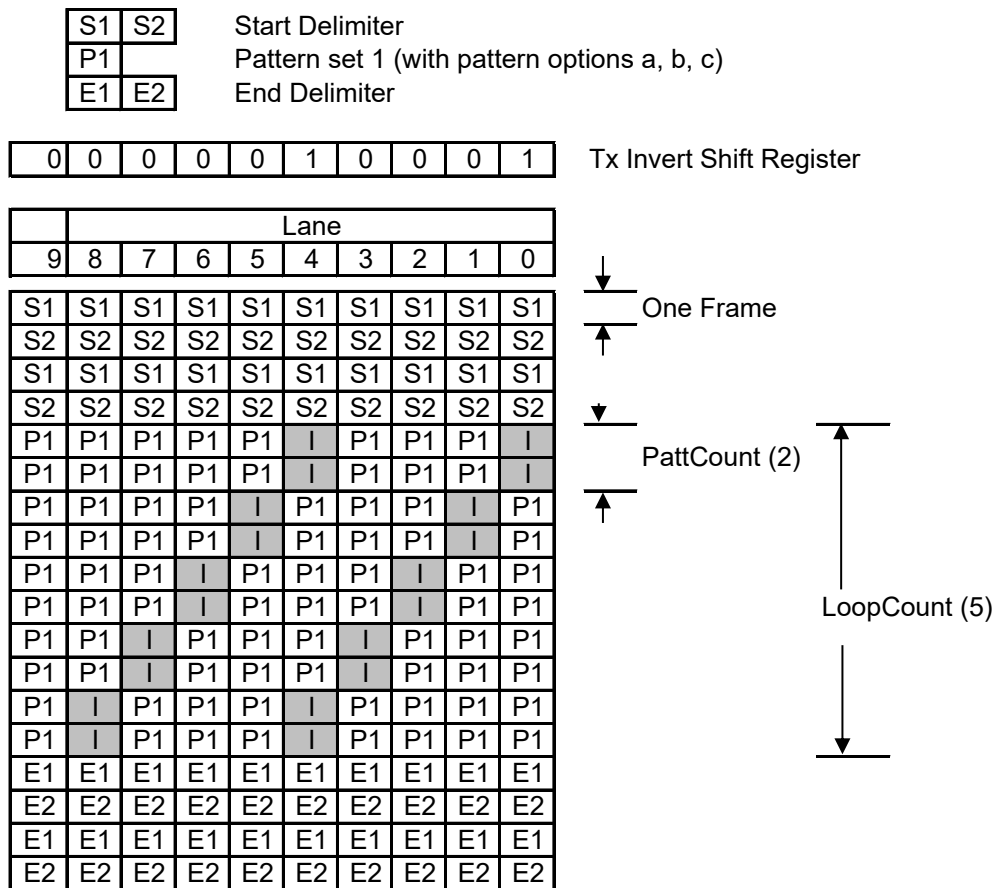


Figure 7-8 — Auto-inversion Sweep Example

### **7.2.3.2 Extended Pattern Set**

Optionally IBIST may support a second pattern buffer (PS2) to supply an alternate pattern set to a specific lane. It tracks exactly with the first pattern set in every aspect. So there is no additional logic to control it. There are no facilities to select its inverse. The cost is the second pattern buffer, an enable register for each Tx and Rx block, and a mux to select which pattern set to use. Other than these logic blocks it is probably more important to consider the routing implications of adding another k-bit bus to support the second pattern generator.

This feature allows a focused pattern set to target lanes for more intensive debug or characterization operations where a single pattern set with inversion cannot provide. For example, one particular lane with a lone pulse could be surrounded by high-frequency patterns. This cannot be performed with PS1 + Inversion.

This feature is currently designated as optional because it is nice to have. There is no evidence to require it be there for platform validation. It would be targeted towards the Host Memory Interfaces because in the future the AMBs will be made by an external vendor and Intel may need more in-depth testing capabilities.

Figure 7-9 shows an example of how the second pattern buffer operates with the auto-inversion sweep and the first pattern buffer.

### 7.2.3.2 Extended Pattern Set (cont'd)

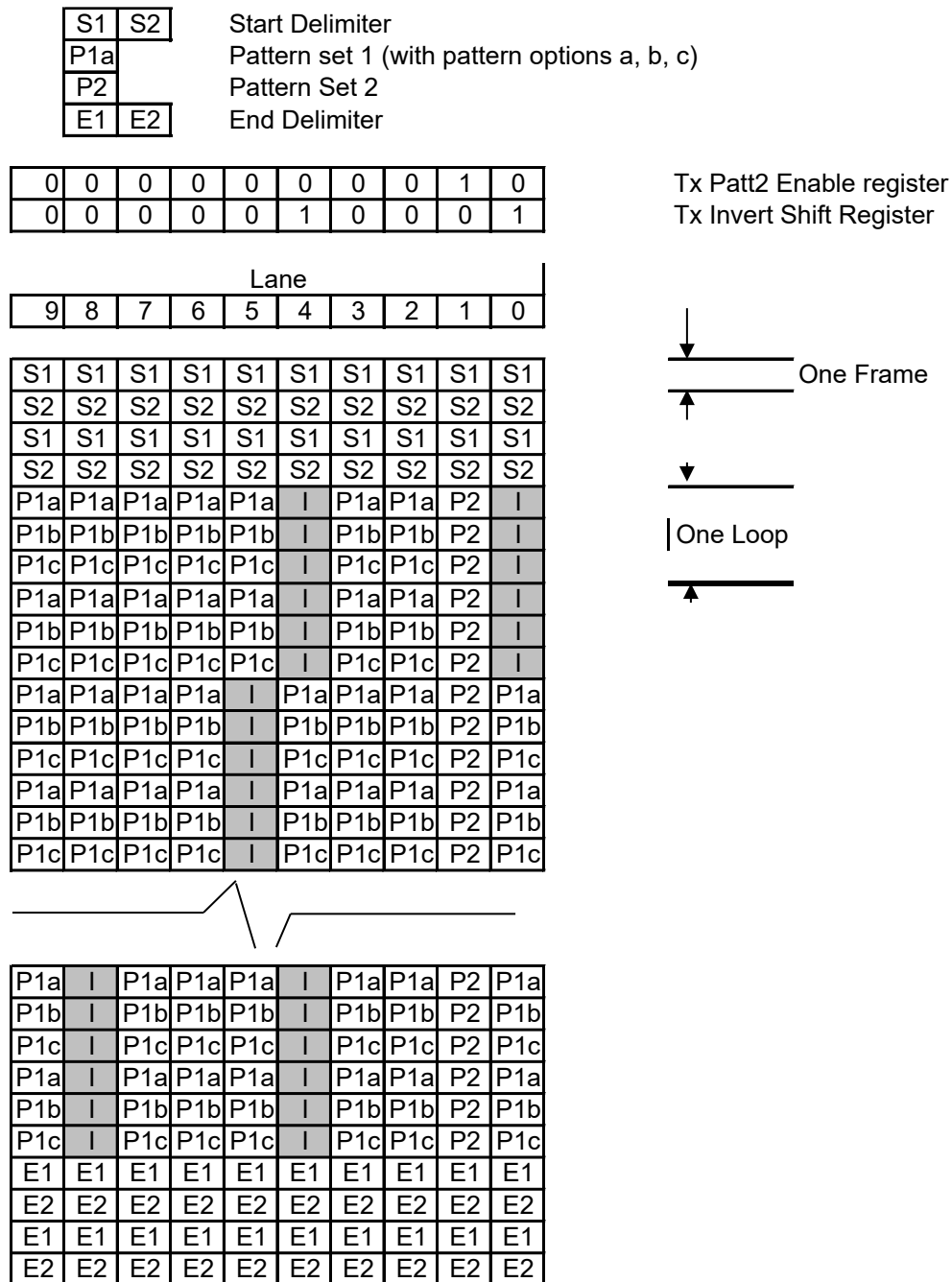


Figure 7-9 — Pattern Set 2 with Auto-Inversion Example

### 7.2.3.3 Examples of IBIST Patterns

Table 7-4 shows how these patterns can be generated.

**Table 7-4 — Patterns Generation Examples**

Pattern Type	Pattern Order	User Defined Pattern (IBPATBUF1)	Pattern Loop counter [6:0]	Clock Pattern (MOD-PERIOD)	CLK Loop Count [6:0]	Constant	Constant Loop Count [6:0]
Lone Pulse 1 106 - 0s then a lone 1 followed by high frequency transitions to maintain link. Overall loop count = 4h	100	1010_1010_1010_ 0100_0000_0000	1	Not Used	00h	0	4h
Lone Pulse 2 30 - 1s then a lone 0 followed by high frequency transitions to maintain link. Overall loop count = 4h	100	1010_1010_1010_ 1011_1011_1111	1	Not used	00h	1	1h
ISI 9 frames of High freq clk then lone 1 pulse. Overall loop count = 8h	010	0100_0000_0000_ 0101_0101_0101	1	001	4h	Not Used	0
Resonance 64 frames (768 sym) Odd fraction of clock frequency using pattern buff.	000	0000_0000_0000_ 0000_1111_1111	20h	Not used	0	Not used	0

#### 7.2.3.3.1 Lone Pulse

A lone pulse can be either a lone-one or a lone-zero pattern. The pattern is composed of a string of zeros/ones with a single one/zero followed again by a string of zeros/ones. It is in essence exactly what the description implies a single zero (or one) embedded in a long string of data of the opposite value. (Ex: 1111\_1111\_0111\_1111\_1111\_1111). When combined with the pattern buffer the constant 1/0 generator can be programmed with a long sequences of ones or zeros to setup the lone pulse. The pattern buffer contains the lone pulse bit.

### 7.2.3.3.2 Differential Common Mode Test Patterns

These are patterns that first build up enough common mode skew using constant frequency patterns followed by an ISI and repeated. These patterns can be used sequentially with different frequency patterns used to build up different common modes.

Pattern symbol definition:

[A] [B] [C] Subsection a, b, or c

1010.1111.0101: Each dot separates nibbles. Three nibbles to an FBDIMM frame (assuming a 12 bit transfer granularity).

1010.1111.0101\_1010.1111.0101: Each underscore separates an FBDIMM frame.

Example patterns:

[clock][PS1][clock]

[0101.0101.0101] [1010.0111.1111\_0101.1000.0000] [0101.0101.0101]

[0011.0011.0011] [1010.0111.1111\_0101.1000.0000] [0011.0011.0011]

[0001.1100.0111] [1010.0111.1111\_0101.1000.0000] [0001.1100.0111]

A mixed frequency ISI pattern contains changing frequency components from high to low but also has lone-pulse like stress embedded in the middle. An example is:

[0101.0011.0001\_0000.1100.1010]

### 7.2.4 Transmitter Block

The transmitter block controls the data flow to the FBDIMM electrical sub-block. It switches in different data streams based on operation mode selected. Table 7-5 shows the transmitter block operation. Transmitter block switches traffic streams when it receives the Go signal. The traffic select is based on the control signals coming in to the Tx Control. Table 7-5 lists the different modes and data stream support.

**Table 7-5 — Transmitter Block Output Select**

Operating Mode	Output
Normal Mode	Regular traffic stream
Loopback Mode	Loopback traffic stream
IBIST Mode PS1 (non-inverted)	IBIST Pattern Set 1 output
IBIST Mode PS1 (inverted)	Inverted IBIST Pattern Set 1 output
IBIST Mode PS2	IBIST Pattern Set 2 output (extended feature set)

### **7.2.5 Receiver Block**

The receiver block manages the incoming data from the FBDIMM electrical sub-block. After entry into IBIST operation, the receiver block enables the incoming data to the error detection logic. The comp bubble is the logic circuit that performs the Direct Compare method which requires the pattern generator as the reference source to compare against the incoming data.

The delimiter match logic will look for the Start delimiter sequence. Once the Start delimiter is detected, the Rx control will start comparing the incoming symbol (frame size of k bits, this spec uses a 12-bit frame size). When the delimiter match logic detects the Stop delimiter, the Rx control will stop the direct symbol comparison and assert the Done signal to IBIST control. Rx control will also stop comparison if an error is found and stop on error bit is set.

### **7.2.6 Error Detection**

Using the direct symbol compare method, a symbol-by-symbol comparison is performed to detect any error during the IBIST pattern transmission. This method assumes the receiver has access to the same IBIST pattern sequence as the transmitter generating it.

Once the receiver detects the four start delimiters, the IBIST logic starts comparing the received symbol one at a time. Symbols can be of different size depending on implementation and can be read from a transmit-buffer or regenerated by the pattern generator. An error is detected when the received symbol does not match the transmitted symbol. This method permits precise identification of where errors occur.

Once an error is detected, the error status is changed in the IBIST Control Register. Since the error status is sticky, only the first error is recorded but a lane error status register is available to record all errors that may have occurred concurrently. This is important when comprehending any adjacency issues with the links.

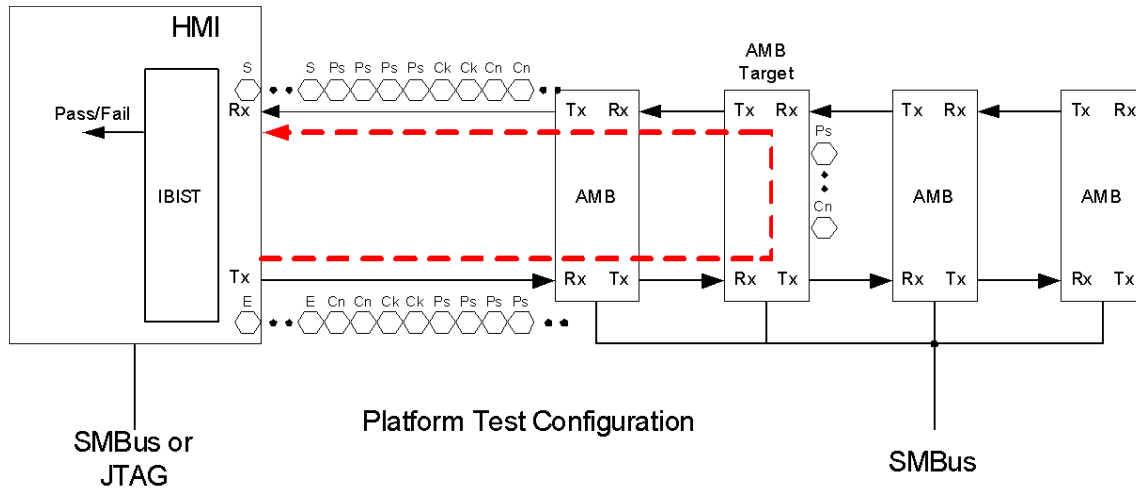
### **7.2.7 Validation and Test Usage Models**

FBDIMM IBIST supports three analog validation and component test usage models; 1) loopback board testing, 2) controller-to-controller board testing, and 3) and HVM component testing.

#### **7.2.7.1 Loopback Testing in a System**

Loopback testing is the most basic form of FBDIMM link testing. It only requires setting the IBSTR bit in the Host Memory Interface's IBIST control register. The training state machine coordinates the entry and exit conditions for the IBIST symbol payload. Figure 7-10 shows a diagram that was previously described in the introductory clause. An AMB IBIST engine does not participate in this mode.

### 7.2.7.1 Loopback Testing in a System (cont'd)



**Figure 7-10 — Standard IBIST Loopback Testing**

The following procedure lists the steps required for this test mode:

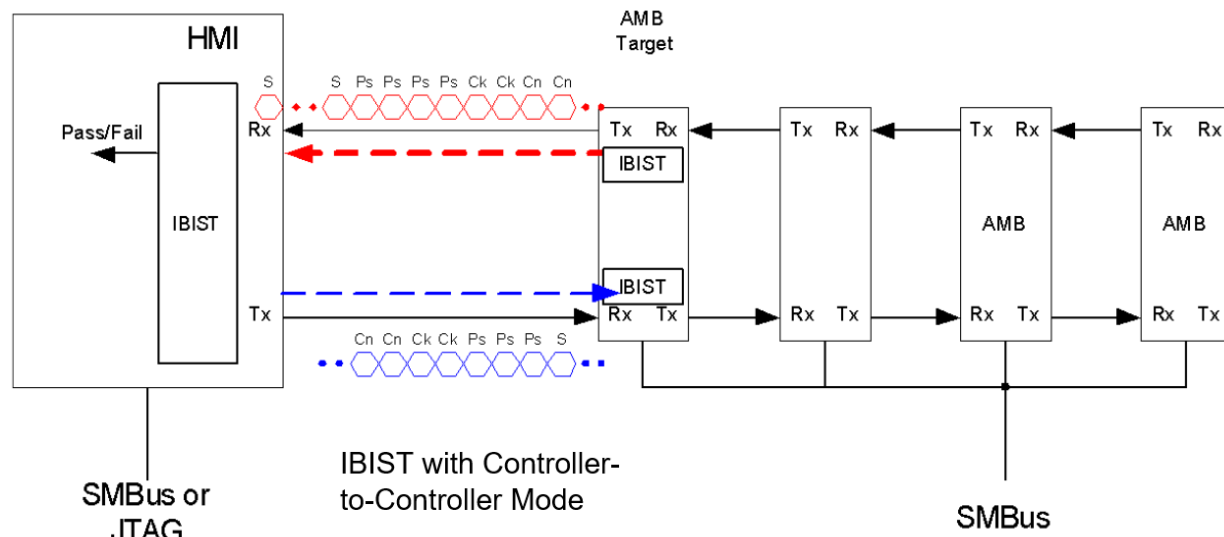
- 1) Define the pattern sequences and write the pattern control registers;
- 2) Write the target AMB DIMM slot number to the HMI link control register;
- 3) Write the IBSTR bit;
- 4) Poll IBCTL for error status;

The same procedure can be followed for Bit Error Rate testing except the Stop on Error bit must be cleared.

### 7.2.7.2 Controller-to-Controller Board Testing

This feature increases the capabilities of testing the link for unique data streams on the northbound lanes. In the loopback model previously described the southbound [4:0] lanes are replicated across the northbound link. Loopback may be sufficient in most cases. A Controller-to-Controller (C2C) mode allows the IBIST engine in the AMB to transmit on each northbound lane according to the patterns defined in that component. Since the IBIST symbol payload is framed by start and end delimiters it is possible to perform checking on each unidirectional link (HMI-SB to AMBSB and AMB-NB to HMI-NB).

### 7.2.7.2 Controller-to-Controller Board Testing (cont'd)



**Figure 7-11 — Controller-to-Controller Board Test Diagram**

A similar testing procedure as previous described for loopback applies here:

- 1) Define the pattern sequences and write the pattern control registers;
  - a) Write the pattern registers in both the HMI and target AMB;
  - b) Write the MSTRMD bit in the target AMB;
- 2) Write the target AMB DIMM slot number to the HMI link control register;
  - a) This will designate the last AMB in the link;
- 3) Write the IBSTR bit in the HMI and IBISTR and MASTRMD in last AMB.
- 4) Poll IBCTL for error status;

The bit error rate testing procedure is also similar to the loopback example.

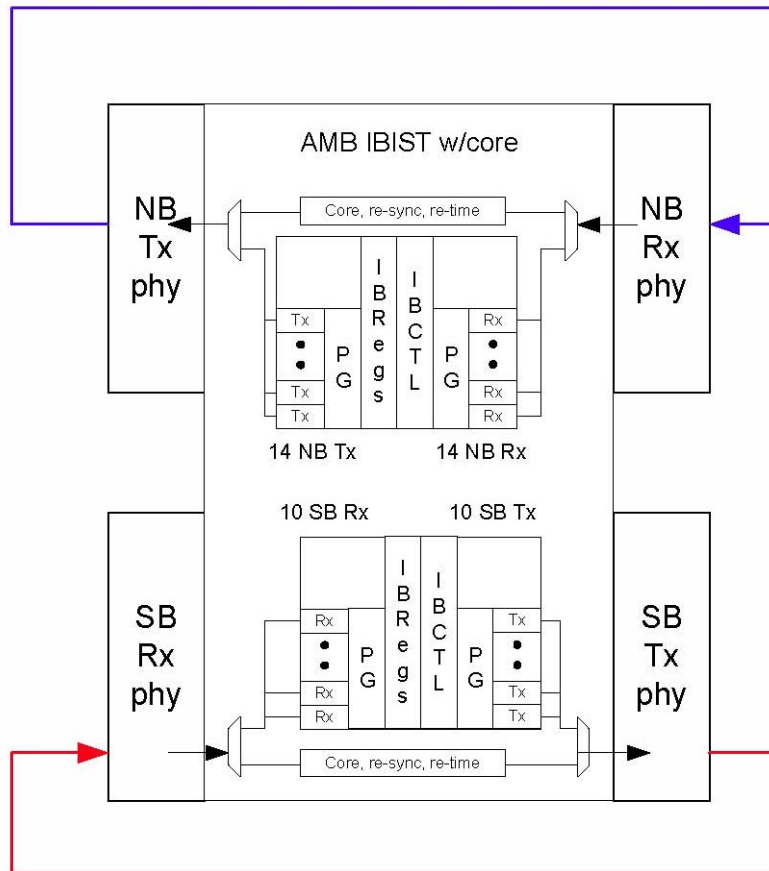
### 7.2.7.3 AMB Component Testing

Component High Volume Manufacturing utilizes the loopback function for testing the IOs at-speed to detect fabrication flaws in the physical layer. Shown in Figure 7-12 is the loopback definition of an HVM AMB component. Loopback still has the same meaning (looping back the data from Tx to Rx) except that the loopback path is a copper trace on a load board and not through a component as in the platform board paradigm as described earlier.

The MASTRMD mode and IBISTINITEN bits must be set to force the AMB into training to perform the functions of a HMI component. The targeted AMB slot number can be anything. Beside these basic differences, the same procedure can be applied.



### 7.2.7.3 AMB Component Testing (cont'd)



One possible external wire trace loopback topology

**Figure 7-12 — AMB External Wire Trace for Loopback Testing**

### 7.2.7.4 Power-On Self Test

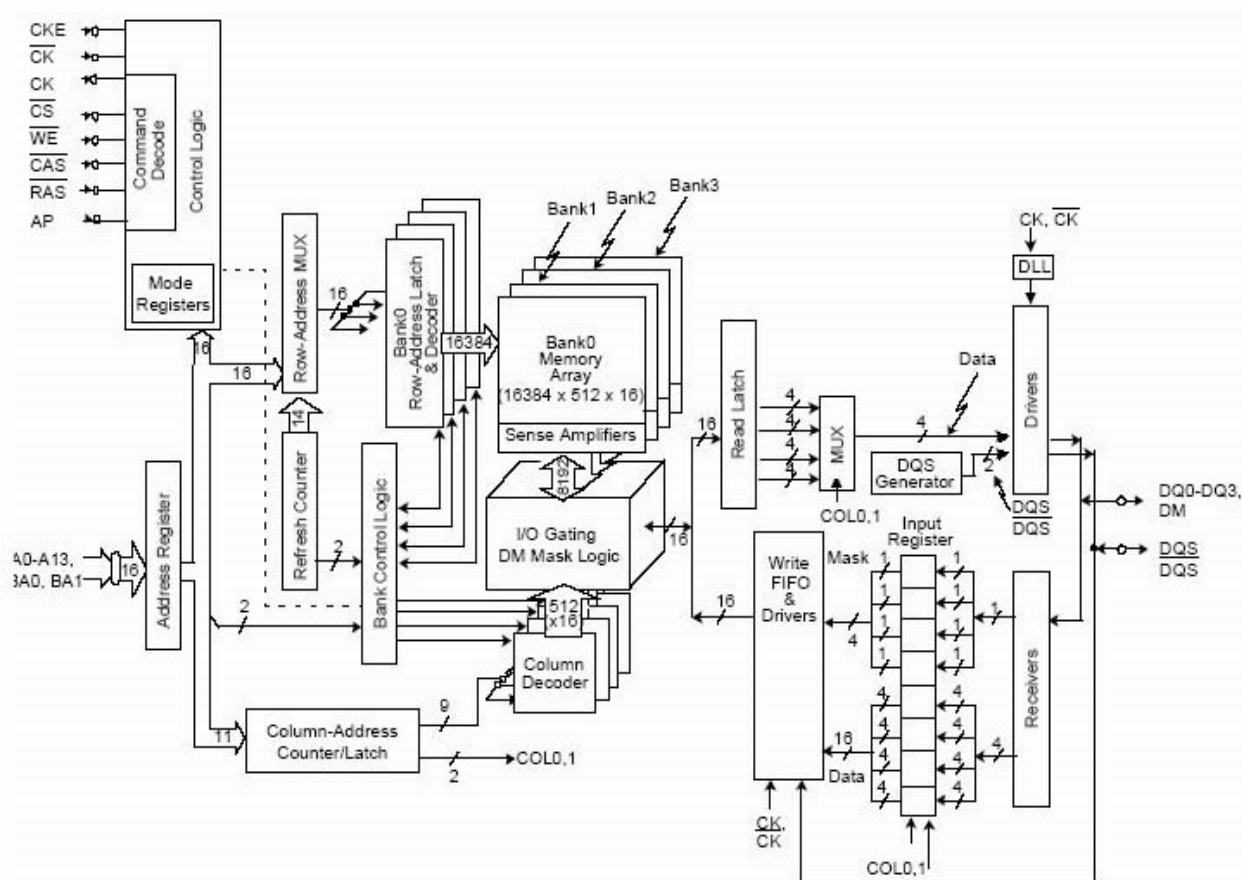
The link logic should enable a fixed mode operation of IBIST as part of the training sequences. Default register settings provide approximately 3 microsecond's worth of symbol stress testing. To enable this capability, a signal from the link training state machine is OR'd with the IBSTR internal signal. Proper coordination between the two machines is necessary to provide a seamless transition.

### 7.2.7.5 System Diagnostics

FBDIMM IBIST can be used by system diagnostic software to investigate potential problems in the system. When a link signals an uncorrectable error or too many correctable errors, the error handler (system management) could invoke IBIST on the failed link to determine if more information on the type of failure occurred. This information could be logged such that a service call would either have advance debug data or arrive with the replacement card in hand. Another usage model would be to initiate IBIST on every hot-add/remove sequence to ensure signal integrity as part of standard system checks before the system is placed back on-line.

## 8 DIMM Test and Manufacturing

Figure 8-1 shows the typical architecture of a DRAM with a 4 bit wide data path. The memory array operates at 100 to 200 MHz. With a pre-fetch of 4, there are 4 bits of data on each array access, allowing us to clock data in or out at 400 to 800 MHz. This 4 to 1 multiplexing and de-multiplexing is performed in the input register or output multiplexer.



### Figure 8-1 — DRAM Architecture

## 8.1 Transparent Mode

Transparent mode is designed to allow access to the DRAM behind the AMB. In this mode high speed pins are converted into low speed pins and mapped to DRAM pins. The objective is to allow the use of existing test equipment and manufacturing processes. The tester must be capable of operation at 200 MHz. Transparent mode offers potential improvements in test capacity over traditional DIMMs. In this mode, FBDIMM requires only 60 active pins to test the DIMM.

Data from the tester is 16 bits wide at 200 MHz (single data rate). The data rate is doubled and the width halved on the way to the DRAM (by clocking out 8 bits of data on the rising edge of the clock and the remaining 8 bits on the falling edge).

## 8.1 Transparent Mode (cont'd)

The tester will drive data to be written to the DRAM on a write pass and data to be compared on a read. DRAM data and the expect data from the tester is compared in the AMB. If the actual and expected data differs the pass/fail outputs will indicate which DRAM failed.

In transparent mode the FBDIMM input pins require 0 to 500 mV swing (half of the normal differential input voltage). Input slew rates should be approximately 5V/ns. This parameter is not critical, but it must be fast enough to be recognized by the AMB receiver. The DDR pins will operate with normal DDR2 timings and levels.

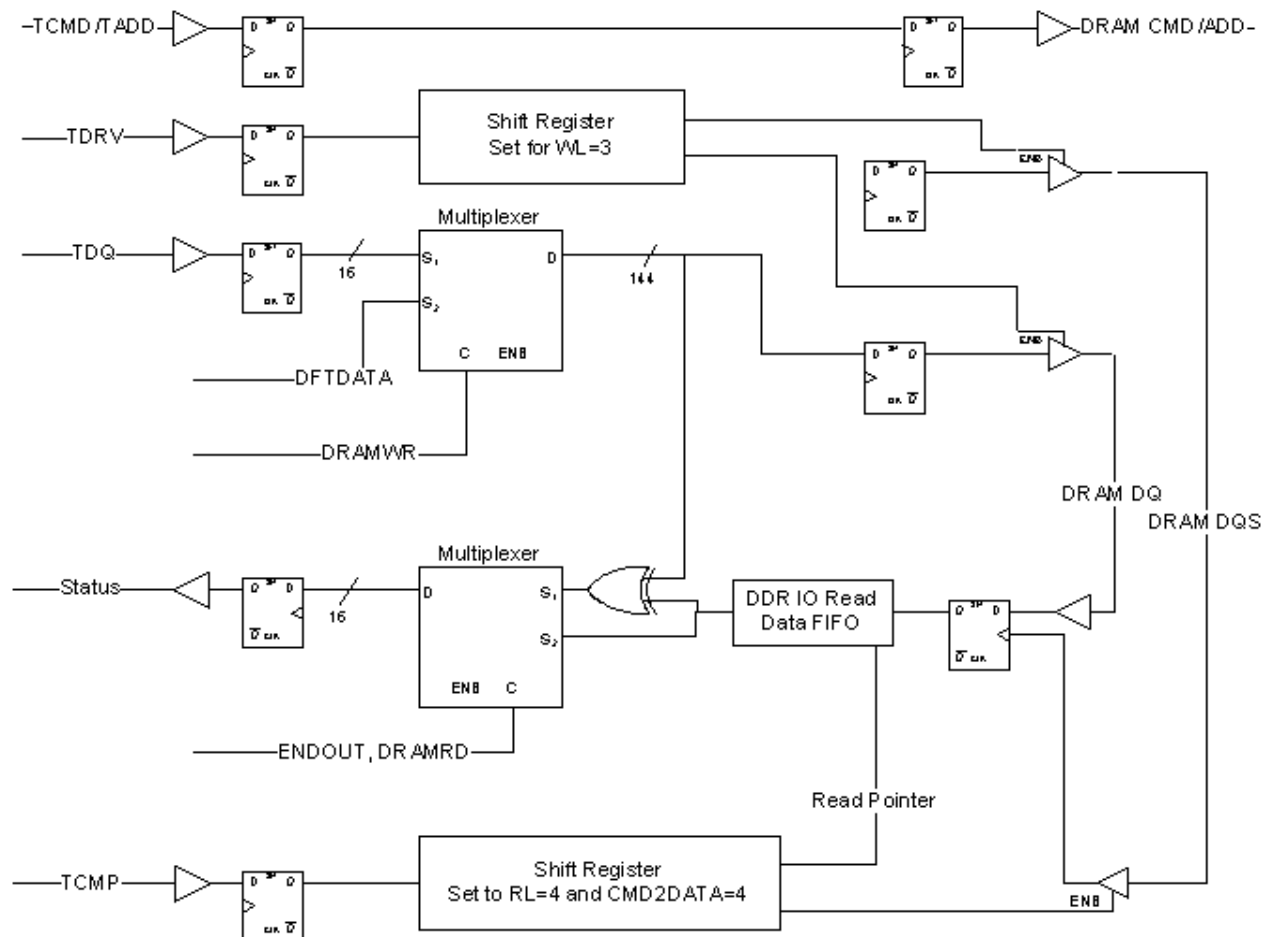
The AMB clock input pins will be used for transparent mode as well as normal mode. This also allows use of most of the existing on-chip clock distribution network.

**Table 8-1 — Transparent Mode Pin List**

DRAM Pin	Type	Quantity	Comment
CKE0, CKE1	I	2	
CS*	I	2	
ODT	I	1	
RAS*, CAS*, WE*	I	3	
BA0-BA2	I	3	
A0-A14	I	15	
TDRV	I	1	Tester signal to drive data on writes
TCMP	I	1	Tester signal to compare on reads
DQ Receiver	I	16	Send even/odd data at same time
Pass/fail	O	9 or 16	Used for pass/fail (8:0) or direct access (15:0)
Sum Receiver		44	
Sum Driver		16	9 pins are used in normal mode, 16 in direct access mode

### 8.1.1 Transparent Mode Architecture

An overview of transparent mode is shown in Figure 8-2.



**Figure 8-2 — Transparent Mode Simplified Block Diagram**

### 8.1.2 Clock Frequency and Core Timing

The DDR2 DRAM clock frequency is 200 to 400 MHz but core timings require several clocks (or ns) to complete.

For example on DDR2-667: tCL, tRP, and tRCD are 4 clocks or 12 ns, tRC is 57 ns, and tRRD is 7.5 ns.

DDR2 transactions are burst-oriented, reading or writing 4 or 8 words of data across 4 or 8 clock edges. Assuming a 4 bit burst, a x8 DRAM will transfer 32 bits on successive edges of 2 DRAM clock cycles. On the tester side of the interface the same 32 bits of data, is transferred, 16 bits at a time, over two DRAM cycles.

### 8.1.3 Edge Placement Accuracy

Command, address and data edges should be reasonably close to the appropriate clock edge but with some margin of error. DRAM setup and hold times are 400 to 600 ps, while the half cycle time is at least 1250 ps. As long as the data is within  $\frac{1}{4}$  cycle (625 ps) of the clock edge, it will not violate setup or hold. Since there will be other error terms (DIMM trace length matching, jitter, etc.) it is recommended the tester be accurate to +/- 300 ps.

### 8.1.4 Transparent Mode Timing

Normally, transparent mode will use DDR2-400 timing even if the DRAM is rated for faster operation. Optionally an AMB may support operation at frequencies higher than 200 MHz.

To set up transparent mode the appropriate AMB registers should be set to AL=0, CL=4 and CMD2DATA=4. Some AMBs may be hard-coded to these values, in which case programming has no effect. These values establish an internal timing relationship as illustrated in the following figures. Optionally other register values may be supported for special test cases.

Actual placement of DRAM read/write or other commands is dependent on the incoming signals, not the AMB register values. Figure 8-3 shows an example of a write, read, write sequence using incoming signals that correspond to WL=4 and RL=5. The timing relationships in red (normal font) are fixed relationships (established by the AMB register settings above). These edges will move together. The relationships in green (*italic font*) are the DRAM timings, which are under control of the tester.

Timing may be changed on the fly (e.g., in the middle of a test pattern) by changing the placement of edges from the tester. DRAM mode registers can be programmed on the fly as needed by including (E)MRS commands in the tester data stream. There is no need to change AMB settings during a test. The only exception is that DRAM BL may be changed on the fly but the data logging logic may get confused if DRAM BL does not match the BL expected by the data logger. All other DRAM settings such as AL and CL may be changed at any time.

[illegible]

#### 8.1.4.1 Write Timing

Figure 8-4 illustrates write timing with the tester set to WL=3, 4, and 5. There is a constant offset of three cycles from TDRV to TDQ and one cycle from TDQ to DRAM DQ. The DRAM mode registers must, of course, be set to the appropriate timing to recognize the read. For BL=8 the TDRV pulse will be 4 clocks wide rather than 2.

#### 8.1.4.2 Extended Write Timing

The TDRV pulse may be extended indefinitely in cases where it is necessary to apply constant data to the DRAM pins. As long as TDRV remains asserted the AMB will continuously propagate data from the tester TDQ inputs through to the DRAM DQ pins (delayed by 1 cycle) as indicated in Figure 8-4.

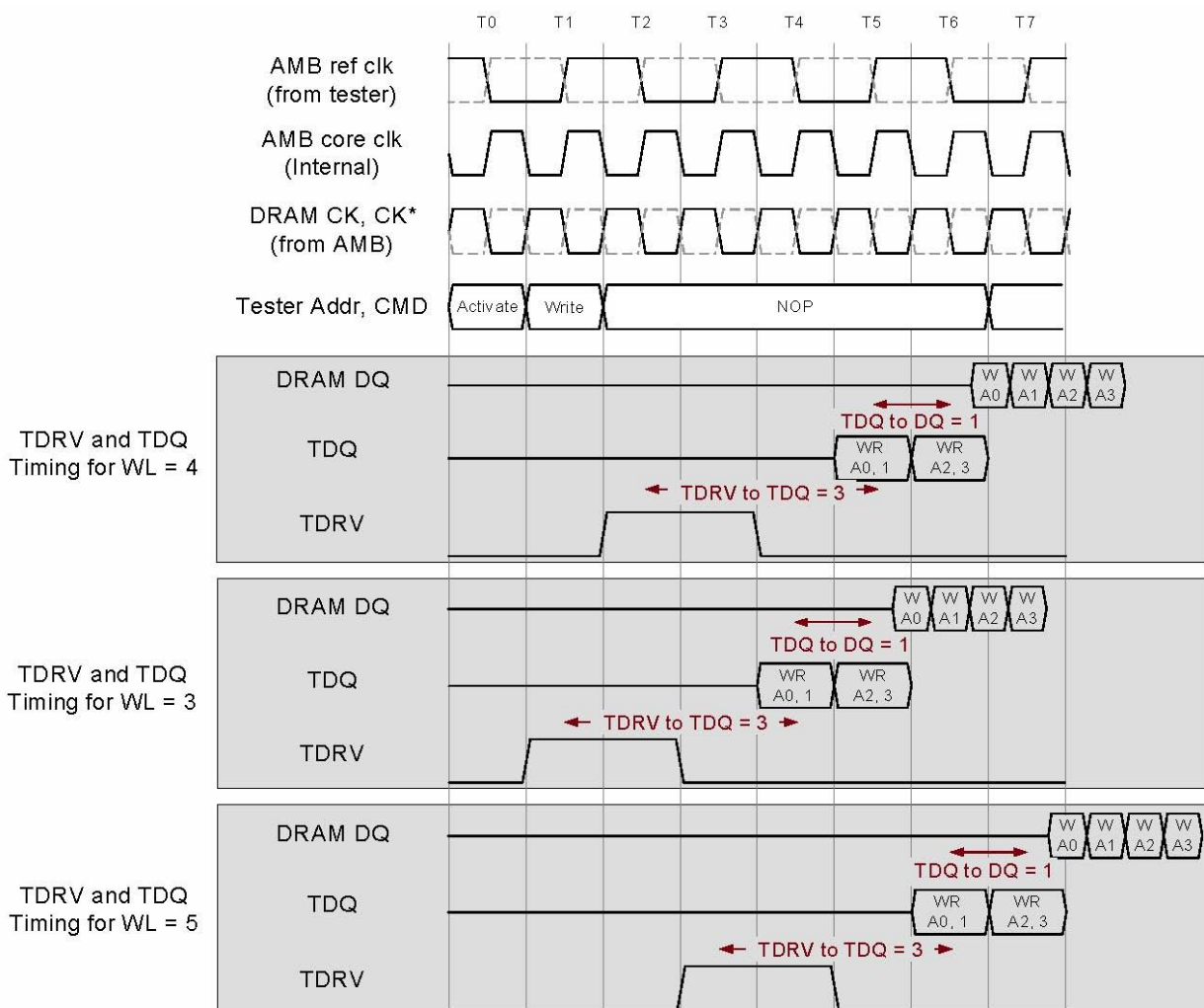


Figure 8-4 — Transparent Mode Write Timing

### 8.1.4.3 Read Timing

Figure 8-5 shows read timing with the tester set to RL=3 and RL=5. Due to complexities in the handling of read data in an AMB, there is a latency of several cycles from the TDRV pulse to TDQ and test status outputs. Specifically there is a constant latency of four cycles plus the programmed CMD2DATA value from TDRV to TDQ (8 cycles total using the AMB settings above) and one cycle from TDQ to DRAM DQ. An AMB may support shorter latencies but this is not required.

TCMP is latched on the rising edge of core clock. This initiates the read inside the AMB. In most cases the AMB will latch DRAM read data slightly before TDQ data is needed. The reason is most AMBs will load DDR data into a queue in the DRAM domain and unload the data on a core clock edge. TDQ is typically not needed until the DRAM data is in the core. The comparison of actual and expected data and propagation back to the tester will occur on the next core clock edge.

The timings in Figure 8-5 are for BL=4. When testing BL=8, the TCMP pulse should be 3 clocks wide rather than 1. Tester DQ data, DRAM data, and the status outputs will be extended appropriately to cover the burst length.

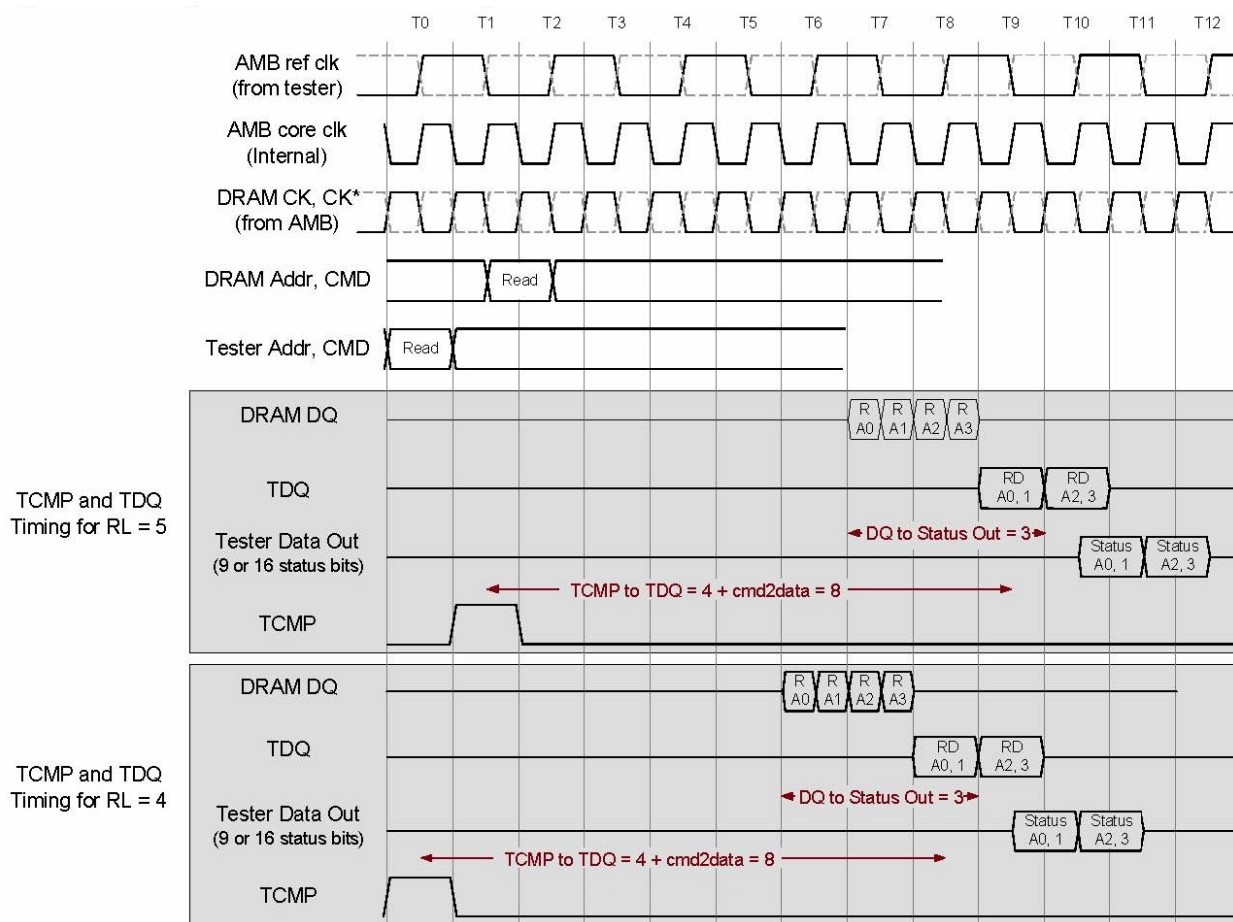


Figure 8-5 — Transparent Mode Read Timing



### 8.1.4.3 Read Timing (cont'd)

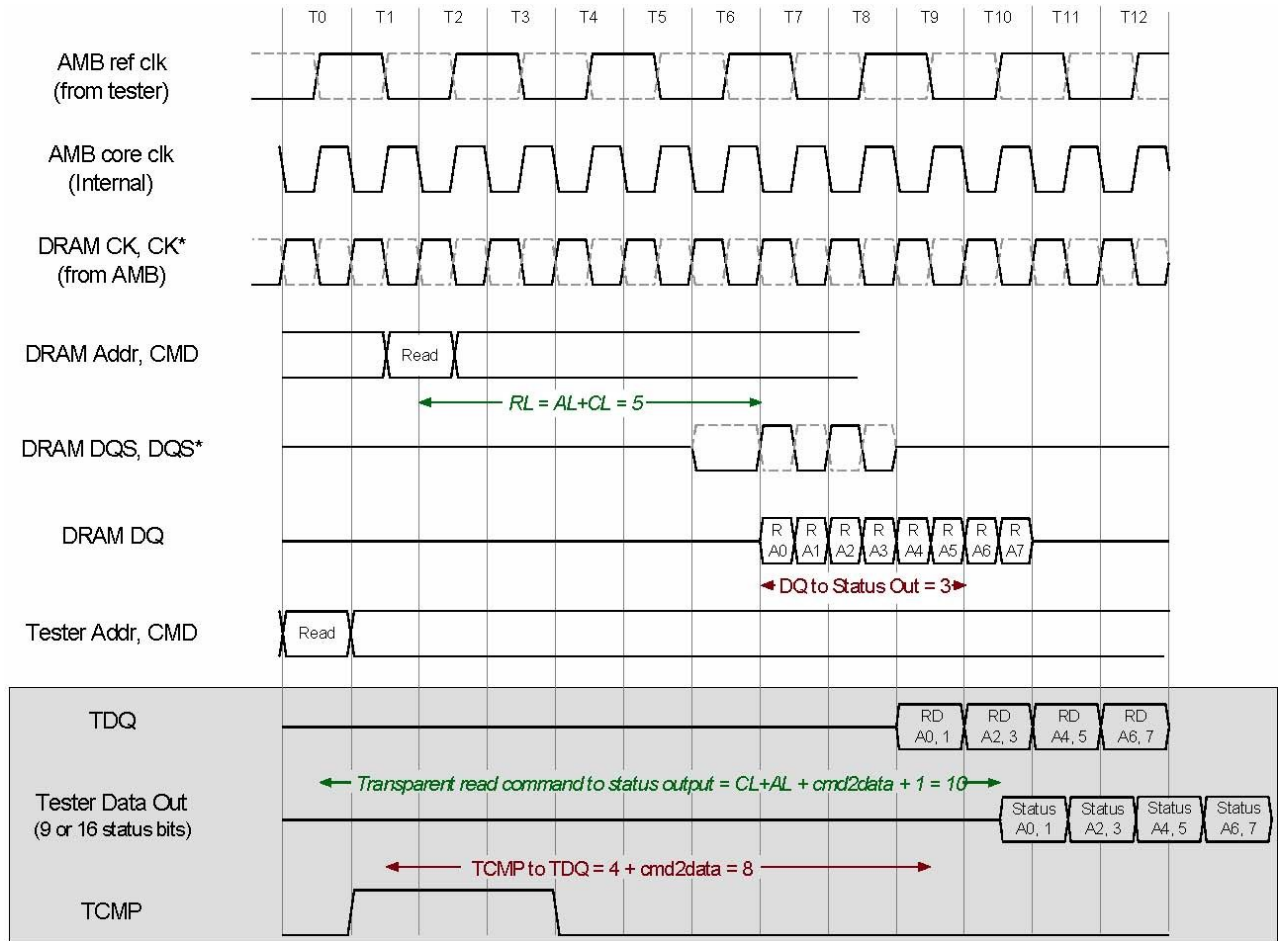


Figure 8-6 — BL=8 Read Timing

### 8.1.5 Error Reporting

By default the status pins will be the xor of actual data from the DRAM and expected data from the tester. The AMB also stores 144 bits of DQ data. If the LGFBITS control bit is cleared the 144 bits of data will be actual data read from the DRAM. Otherwise the result of the data compare is stored. In either case the tester must still provide expected data for the AMB to properly set the status pins.

Normally the test will stop when an error occurs. It is the responsibility of the tester or test program to track errors, read error registers and stop or continue as appropriate. This may require multiple iterations of the same test to execute, collect data, and re-start the test.

### 8.1.5.1 Multiple Failures

There are some implications if multiple failures occur in the same DRAM burst. Three control register bits determine how these failures should be captured. If the log first fail (LGFFAIL) bit is set, the AMB will record only the first failure in a burst. When the bit is cleared (default), the AMB will record the last failure at the burst position matching the burst position (BSTPOS) setting. If the LGFFAIL bit is set and an error is logged, no further logging will occur until the bit is cleared.

- If a failure is detected only in one half of the burst (first and second or third and fourth data words), the error pins will indicate which DRAM or DRAMs failed. The error register will indicate which data lines failed and in which data words.
- If a failure is detected in both halves of the burst (data word 1 or 2 and words 3 or 4), data from the second failure will overwrite data from the first failure. By default this is prevented by the log fail bit. The error pins will continue to operate correctly. If it is desired to collect data from a specific portion of the burst the burst position bits can be used to select an appropriate burst position to record. For a 4 bit burst it is possible to select data from the first or second half of the burst. For an 8 bit burst there are four positions to choose from. These mappings are illustrated in Figure 8-7.

Log First Fail	Burst Position		BL=4			
Bit0	Bit1	Bit0	1	2	3	4
1	x	x	144 bits*		or 144 bits*	
0	0	0	144 bits			
0	0	1			144 bits	

Log first Fail	Burst Position		BL=8							
Bit0	Bit1	Bit0	1	2	3	4	5	6	7	8
1	x	x	144 bits*		or 144 bits*		or 144 bits*		or 144 bits*	
0	0	0	144 bits							
0	0	1	144 bits							
0	1	0	144 bits							
0	1	1	144 bits							

\* the first failure will be saved in whatever burst position it occurs

**Figure 8-7 — Mapping of Burst Position Bits to Error Capture**

### 8.1.5.2 Direct Access - Testing of Individual DRAMs

In certain cases it is desirable to test one or two DRAMs. Transparent mode allows direct access to a single x8 or two x4 DRAMs. In this mode 8 DDR DQ pins are demultiplexed onto 16 SDR status pins, providing a 16 bit input data path (on TDQ) and a 16 bit output data path on the status pins.

The transparent mode configuration register has one bit (ENDOUT) to enable this mode. On reads, the DRAMRD bits will select the bytes of DRAM data to be presented on the status pins. On writes the DRAMWR bits select a DRAM to receive data from the TDQ bus. A separate register holds 8 bits of default data to be applied to non-selected DRAMs in the early/even cycles and another 8 bits for late/odd cycles. The mapping of these bits to DQ selection is illustrated in Table 8-2.

**Table 8-2 — Selection of 8 bit Data Paths when ENDOUT is Set**

		Early Data DQ Byte	Late Data DQ Byte
DRAM RD/WR	DQ		
0xF (DRAM WR only)	all bytes		
8	71:64	8	17
7	63:56	7	16
6	55:48	6	15
5	47:40	5	14
4	39:32	4	13
3	31:24	3	12
2	23:16	2	11
1	15:8	1	10
0	7:0	0	9

DRAMWR is the byte of data bus selected to receive transparent write data, and byte of data bus to be compared against transparent read data. DRAMWR allows a setting of 0xF (all ones) which sends the TDQ input data to all DQ bytes. DRAMRD is the byte of data bus selected to be output on transparent data/status pins when ENDOUT bit is set.

### 8.1.6 Transparent Mode IO Specifications

Listed below are the specifications for transparent mode input and output pins.

**Table 8-3 — Transparent Mode FBDIMM Interface Signaling Specifications**

	Minimum	Maximum	Units
I/O voltage swing	0	500	mV
Input slew rate	2		V/ns
Input to refclk (rising or falling edges) setup time	3000		ps
Input to refclk (rising or falling edges) hold time	1000		ps
Status output valid to refclk time	-1000	+1000	ps
Vref	200	300	mV
Vil (DC)	-300	200	mV
Vil (AC)	-300	150	mV
Vih (DC)	300	900	mV
Vih (AC)	350	900	mV
Voh	400		mV
Vol		100	mV
Ioh	8		mA
Iol	12		mA
Ioh: current into a 50-ohm external load to ground, with on-die transmitter termination enabled. Iol: current into a 50-ohm external load to 1.5 V supply rail, with on-die transmitter termination enabled			

### 8.1.7 IO Implementation Guidelines

#### 8.1.7.1 Dedicated Receivers

Simple one-stage receivers for the transparent mode have been added in parallel to the existing high-speed sampling receivers. The latter can be turned off during transparent mode, as well as the DRC and the phase interpolator, to save power and avoid noise. An internal VREF set to 0.25 V will be used, so the tester signals should oscillate between 0 and 0.5 V. The transparent mode RX should be turned off during normal mode, so as to save power/avoid noise.

#### 8.1.7.2 Common Clock Scheme

To avoid costly implementations using strobes and FIFOs, a common clock scheme is followed, implemented in the core, where the data capture flops reside. Since transparent mode data signals from the tester are all in phase with the 100 MHz system clock, the clock used for the capture flops has to be aligned with the external system clock (or slightly delayed, to account for the propagation delay difference between data receivers and clock receiver).

Aligning core clock and external clock can be done using the HVM mode circuitry included in the PLL. The HVM clock tree will have to feed the capture flops, and one of its leaves has to be fed back to the PLL, in order to achieve adequate synchronization.

### 8.1.7.3 Tester Interface Clock and Data Routing

The following uncertainties have to be factored in:

- Tester board (TIU) trace mismatches
- Package trace mismatches
- FBDIMM low speed RX propagation delay variations due to PVT
- Set up and hold of capture flops (typically <350 ps depending on process, voltage, and temperature)
- Clock synchronization mismatch, core clock PLL and tree jitter (approximately +/-200 ps)

At 200 MHz, there is a 5 ns data window. The potential FBDIMM low speed receiver variation is approximately 300 ps propagation variation over process, voltage, and temperature. Package and tester interface mismatches are not expected to exceed 200 ps. Flop setup variation should also be less than 200 ps. After removing 400 ps for clock uncertainties leaves 3.9 ns margin. While this is plenty of margin, some amount of trace matching should be done on the tester interface to minimize skew.

### 8.1.7.4 Outgoing Control Signals

Only the TX+ pin should connect to the tester. The data on TX- can be discarded (it will toggle at the same rate as TX+).

Terminating TX+ on the tester should be a given, as every tester offers this capability. Terminating TX- could be done on the tester, by having a TIU (tester board) with a route and a tester connection allocated for it. It may be cheaper to just have a 50 ohm resistor tied to ground on the TIU itself, from the TX- pins.

### 8.1.7.5 Crossing Clock Domains

To avoid the crossing clock domains from core clock to FBDIMM fast clocks, the transparent mode data flows directly through the Analog Front-end Unit of the TX.

### 8.1.7.6 Quad Rank Operation

Table 8-4 lists the transparent mode assignments for the added signals.

**Table 8-4 — Transparent Mode Mapping**

Link signal	DDR signal	Existing usage
SN[1]#	CS[0]#	CS[0]#
SN[1]	CS[1]#	CS[1]#
SN[5]#	CS[2]#	unused
SN[13]	CS[3]#	A15
SN[2]	ODT[0] (existing)	ODT[0]
PS[9]	ODT1	unused
PS[9]#	ODT2	unused
PS[0]	TDQ8 and XORA2	TDQ8
PS[1]	TDQ9 and XORA6	TDQ9

### 8.1.7.6 Quad Rank Operation (cont'd)

A special case applies for MRS cycles. In order to control the ECCA2 and ECCA6 bits separately from A2 and A6, PS0, and PS1 are used as XORs during MRS commands (RAS, CAS, and WE all low). PS0 and PS1 are used for data inputs during read and write commands, but since MRS commands require all pages to be precharged, the data signals are not used at that time.

The AMB will XOR the input signals to create the ECCA2 and ECCA6 in the following manner during an MRS command:

**Table 8-5 — Transparent Mode Mapping of XORA2 and XORA6**

DRAM output pin	Non-MRS commands	MRS commands
A2A and A2B output	SN[8]#	SN[8]#
A6A and A6B output	SN[12]#	SN[12]#
ECCA2 output	SN[8]#	SN[8]# XOR PS[0]
ECCA6 output	SN[12]#	SN[12]# XOR PS[1]

NOTE In Transparent mode, the standard mapping of bank addressing applies. The Mode C remapping of BA0 does NOT apply in transparent mode.

### 8.1.7.7 Usage Models

#### 8.1.7.7.1 Host Side Usage

TX: The transmitters are set the same as in normal operation (bias on, enable termination, etc.).

RX: RX+ and RX- signals will be independent. Incoming data will free-flow through the I/O (no flops). The routing distance from transparent\_rxout pins should be the same for all capture flops. All input flops should be clocked by the dedicated HVM clock. Receiver termination should be enabled.

#### 8.1.7.7.2 Tester Side Usage

The tester interface should have trace-matched data signals, to avoid skews > 1 ns. The tester should have 50 ohm terminations to ground for all TX pins in use (on-tester termination can be used where applicable). The tester should enable 50 ohm terminations to ground for all the signals sent to RX pins. This will guarantee reasonable signal integrity.

## 8.2 Memory BIST

### 8.2.1 System Level Test

The AMB supports memory self test for memory initialization during system boot up and for testing the installed memory. During DIMM manufacturing this mode may be used to apply tests at speed to test the AMB-DRAM interface. The table below describes the features of the MemBIST engine.

At system level MemBIST may be executed on multiple DIMMs simultaneously. This is expected to speed memory test during system boot.

### 8.2.2 DIMM Manufacturing

During DIMM manufacturing, there is a need to detect assembly-related defects, interface defects and, in many cases, array-related defects. Verification of proper connectivity of the AMB to DRAM can be tested functionally with a variety of addresses and data patterns. The fastest data pattern will normally be to walk a one through a field of zeros and vice versa. Addressing may be verified by walking through various combinations of page, row and column addresses. Control logic and connectivity will be verified by successful reads and writes.

In most cases, transparent mode will be used for memory core testing while BIST will be more useful for DRAM interface testing. For this reason there is no plan to specify a comprehensive BIST capability. The core design is extensible, so it is possible to add AMB supplier-specific features beyond those described below.

Testing may be initiated either in or out of band, making MemBIST compatible with motherboards, low cost ATE, or standalone equipment such as continuity testers. During motherboard testing an FBDIMM-enabled system will be required for in-band access but any motherboard or SMBus controller may be used if testing is to be performed out of band.

It is expected that traditional system test methods will be used for operating system or application-based testing of the memory subsystem. This may include existing memory stress tests, applications or other tests selected by the DIMM manufacturer.

### 8.2.3 DDR Interface Testing

As described above, the primary use of MemBIST is to detect defects in DIMM assembly and operation of the AMB and DRAM interfaces at speed.

DDR interface testing requires stress of the AMB DDR IO and the DDR IO to core path in the DRAM. The DRAM write data path will cover the receivers, input register and write FIFO. The read data path will typically include a read latch, mux (controlled by column addresses), DQS generator, DLL and output drivers. The address path is common and includes receivers; address register, control logic, row address mux, row address latch and decoder, bank control logic, column address counter/latch and column decoder.

Covering this logic will require:

- Patterns are delivered at full speed (800 MT/S).
- Incrementing and decrementing addresses. The address decoders are best tested with marching or other non-linear patterns.
- Alternating data patterns (single rotating bits, checkerboards) to detect slow nodes or capacitive coupling in the data path.
- Interface timings (nominal clock cycle time, setup, hold, pre and post-amble)
- Verification of ODT operation at speed.

## **8.2.4 MemBIST Overview**

MemBIST has a number of modes of operation, data formats and control bits. In addition MemBIST supports a variety of DRAM timings and densities. In general the required bits are grouped into a memory technology register, a control register, a data register, and an error or data reporting register. There are additional registers for setting start and end addresses for the test.

A fundamental architectural issue is that unlike transparent mode, which simply replicates 8 bits of data across the DQ bus, MemBIST can control individual bits in the 72 bit DQ bus. Another key item is the AMB operates at the DRAM address rate. Thus, the AMB provides two 72 bit words, or 144 bits for each DRAM clock cycle. For this reason the data registers in MemBIST are designed for a 144 bit data path operating on “early” and “late” phases of the DDR clock cycle. Early data will be provided on the rising edge of CK. Late data will be provided one half cycle later on the falling edge of CK. DRAM compare data is treated in a similar manner with appropriate clock alignment.

### **8.2.4.1 Memory Addressing Modes**

A memory test is characterized by a starting address, an ending address and a direction. Direction is further split into incrementing or decrementing and counting by row and/or column. For this reason the address generation logic has counters for row, column and bank addressees. Fast X addressing increments (or decrements) in the X or row direction first. When the counter wraps around, the Y or column counter is incremented or decremented. The AMB continues working row-wise through the array until the end of test. Fast Y addressing is similar except that counting occurs in the Y direction. Fast XY is a combination of the two, resulting in a diagonal addressing pattern.

Addresses in MemBIST are logical addresses, meaning they follow the order of the DRAM external address pins. This actual arrangement of bits in the array (the physical address) will differ from the logical address. Therefore, an addressing scheme or data pattern may not be applied to the array exactly as one might think. For example, accesses to logically adjacent cells will not necessarily generate physically adjacent array access. For general purpose testing such as that intended for MemBIST this is not an issue. In-depth array testing is best done on a portion of the array where the logical to physical mapping is known, or in transparent mode where one has full control of address and data sequencing.



### 8.2.4.2 Address Definition

All addresses in MemBIST have three components: a row address, a column address and a bank address. These values are concatenated in a 32 bit register. Column address 0 is not stored since the MemBIST engine has an internal 144 bit architecture. Column address 10 is used for auto-precharge (always low in MemBIST) so it is not specified in the address register.

User-defined start and end addressing is constrained to a column address modulo the burst length. For instance, at BL=4, a memory access could start at column 0. The next access would start at column 4 and so on. A burst will always end on a page boundary, eliminating boundary checking and simplifying the MemBIST sequencer and control logic. The address register bits will reflect these constraints. BL=4 will allow specification of column bits 14..2, while BL=8 will allow specification of column bits 15..3.

Address Bit																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	R	14	13	12	11	9	8	7	6	5	4	3	2	2	1	0
Row																Column										Bank					

NOTE Bit 15 "R" = Reserved for future use

**Figure 8-8 — Memory Address Definition, BL=4**

Address Bit																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	R	15	14	13	12	11	9	8	7	6	5	4	3	2	1	0
Row																Column										Bank					

NOTE Bit 15 "R" = Reserved for future use

**Figure 8-9 — Memory Address Definition, BL=8**

### 8.2.4.3 Address Generation

The address generation logic and controlling register bits allow a variety of methods to traverse the address space of the DRAM.

#### 8.2.4.3.1 Default and User Defined Counting

By default MemBIST will test the entire DRAM address space, counting first by column, then by bank and row (CBR order). If we wish to focus on a particular area of memory, a starting row, column and bank and ending row, column and bank will be used to define the address space to be tested. In addition the user must set the direction bits to establish increasing or decreasing counting and fast X (RCB order), fast Y (CRB order) or fast XY direction as needed. With either incrementing or decrementing address counting the row, column and bank starting addresses must be numerically lower than or equal to the ending address for proper operation. When using Fast X, Y or XY addressing MemBIST will traverse a single bank, defined by the starting address. Optionally MemBIST may use the end bank address to create multi-bank patterns. The user may also select a single location to be written or read.

### 8.2.4.3.1 Default and User Defined Counting (cont'd)

The test will complete when all counters reach the values specified for end address. The BIST engine does not check or adjust the addresses to determine if the values will converge on the end address. It is up to the user to specify values that will eventually be reached. If the ending address is not hit on the first traversal of the array the counters will roll over and continue counting. If for some reason the test needs to be stopped there is a termination bit setting in the control register. If this bit is set, MemBIST execution will stop at completion of the current memory read or write operation.

The user-defined starting and ending banks may be different. If the banks differ, the rows and columns in the starting bank will be tested first, followed by the second bank and so on until all banks have been tested.

### 8.2.4.3.2 Address Inversion

To further stress the DRAM there is a control bit to invert the address lines on every access. The least significant address bit is not inverted since it already toggles at the address rate. All address lines (X, Y and bank) will be inverted, creating a ping-pong access pattern. If the starting and ending bank addresses differ as described above, the bank bits will also be inverted on every other access, creating a two-bank pattern. In this case, MemBIST will activate both banks, since otherwise the AMB would be constantly activating banks. Note that two-bank behavior is limited to cases where addresses are applied column-wise, since row-wise operation requires precharge and activate on each row. As indicated in Table 8-4, it is possible to traverse 4 or 8 banks. However, only two banks (bank and bank bar) are required to be active at a given time. There is no dependency on the order of activation or precharge between banks in two bank mode. It is possible to activate bank a then bank b or bank b then bank a with no difference in the test result.

**Table 8-6 — Address Inversion**

Normal					Inverted, Single Bank					Inverted, 4 Bank				
bank	bit 3	bit 2	bit 1	bit 0	bank	not 3	not 2	not 1	not 0	bank	not 3	not 2	not 1	not 0
00	0	0	0	0	00	0	0	0	0	00	0	0	0	0
00	0	0	0	1	00	1	1	1	1	11	1	1	1	1
00	0	0	1	0	00	0	0	1	0	00	0	0	1	0
00	0	0	1	1	00	1	1	0	1	11	1	1	0	1
00	0	1	0	0	00	0	1	0	0	00	0	1	0	0
00	0	1	0	1	00	1	0	1	1	11	1	0	1	1
00	0	1	1	0	00	0	1	1	0	00	0	1	1	0
00	0	1	1	1	00	1	0	0	1	11	1	0	0	1
00	1	0	0	0	00	1	0	0	0	00	1	0	0	0
00	1	0	0	1	00	0	1	1	1	11	0	1	1	1
00	1	0	1	0	00	1	0	1	0	00	1	0	1	0
00	1	0	1	1	00	0	1	0	1	11	0	1	0	1
00	1	1	0	0	00	1	1	0	0	00	1	1	0	0
00	1	1	0	1	00	0	0	1	1	11	0	0	1	1
00	1	1	1	0	00	1	1	1	0	00	1	1	1	0
00	1	1	1	1	00	0	0	0	1	11	0	0	0	1
01	etc				01	etc				01	etc			
01	etc				01	etc				10	etc			

#### **8.2.4.3.3 Address Inversion for VTT Balancing**

FBDIMMs use a separate termination (Vtt) power supply for command/address termination. Keep in mind the AMB has two copies of the CA bus. In normal operation the AMB attempts to balance the number of high and low address lines by inverting one of the address ports. Inverting one copy of the addresses reduces Vtt power consumption. This operation is invisible to the controller and DRAM, as this is simply a static inversion of address lines.

By default this behavior is enabled for all memory accesses including during MemBIST operation. Most memory test patterns assume a particular address sequence. Vtt balancing may not be desirable in these cases. The AMB provides a control bit (DRC.BALDIS) to turn off Vtt balancing if desired. If the bit is set, no balancing will take place. If adjacent address inversion is disabled Vtt power may increase substantially, placing additional load on the system power supply.

#### **8.2.4.4 Memory Data Formatting**

In normal operation 144 bit data will be used to write and read memory. This data may be a constant value such as all zeros or ones, 5h, Ah, or a user-defined pattern. Several pre-defined data values may be selected through register settings. The data register allows definition of a 144 bit data pattern. The data will be written in two consecutive locations within a memory burst. Since a burst is at least 4 data words, early data is written to the first and third locations in the burst and late data to the second and fourth locations. An 8 bit burst will be treated in similar manner.

There are certain cases, such as initializing ECC bits in system memory that may require unique data in each data word. This requirement is supported by allocating all of 288 bits of the data register for writing data, where normally, half of the bits would be used for failure information. Thus, 288 bit mode is a write-only mode.

In addition, MemBIST has the ability to shift data. The 144 bits of DQ data will be circular shifted one bit to the left at the DRAM address rate, which is once per 144 bits. This allows the creation of diagonal patterns such as walking ones or zeros. A lone one in a field of zeros (and vice-versa) is an effective test for continuity on a DIMM DQ bus.

The final data source is a random data generator. This generator will create (CRC 32) pseudo-random patterns starting from a user-defined 32 bit seed. This type of data is useful for general memory stress patterns. The 144 bits of random data will be updated at the DRAM address rate as with the data shifter.

#### **8.2.4.5 Error Reporting and Control**

MemBIST provides a 144 bit failure accumulator. This register is automatically zeroed at the start of test. During execution these bits are “sticky” (once set they will stay set). The bit positions will indicate which bits have failed sometime during the test. This register is used primarily for detected failed data lines to facilitate DRAM replacement during DIMM manufacturing.

The first failing DQ data (after the Memory Test Address Pointer Register has counted down to zero) is recorded in the failure data register. As with other data registers this is a 144 bit register. Unlike the failure accumulator this register reflects only a single failure.

### 8.2.4.5 Error Reporting and Control (cont'd)

The address corresponding to the data in the error register (the failing address) is also recorded to aid in diagnosis of the failure.

In the default case testing will stop on a failure, leaving appropriate information in the failing address and data registers. If the test is directed to continue, data in these registers will be overwritten if there is a subsequent failure.

#### 8.2.4.5.1 Failure Address Reporting

Failure addresses are reported in a slightly different format than they are specified. The reason is there are a few more bits needed to specify the location of a failure within the burst. Column bit 0 is not needed due to the 144bit architecture of the MemBIST engine.

Address bit																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	14	13	12	11	9	8	7	6	5	4	3	2	1
Bank			Row																Column												

Figure 8-10 — Failure Address Format

#### 8.2.4.5.2 Multiple Failures

Multiple failures require the ability to isolate specific failing addresses (including location in a burst). This is accomplished with a 32 bit register that counts the number of failing 144 bit “chunks” of data.

When a failure occurs MemBIST will stop and report the failing address (including burst position), and 144 bits of failure data.

To record all failures the failure counter should be incremented and the test restarted. This process can continue in a loop until the desired end address and burst position (or end of memory) is reached. Internally MemBIST will load the failure number into a counter and count down each time it encounters a failure. MemBIST will stop at the next failure after the counter reaches zero.

To generate a complete data log of all failures, use the following procedure:

- 1) Define starting and ending addresses, address modes and data patterns, set failure number to zero.
- 2) Start the test
- 3) Read out the failing address and data
- 4) If the failing address is less than or equal to the end address, continue, otherwise exit
- 5) Increment the failure number
- 6) go to step 2

#### 8.2.4.6 DRAM Throttling

MemBIST can generate high DRAM bandwidth and consequently, high power consumption and thermal stress. Although this is manageable in a dedicated test environment, a system's power and cooling capacity may be exceeded. For this reason MemBIST allows insertion of deselect cycles on every address change. The deselections will be inserted after each read or write command except at the end of a row or page. This allows slowing down BIST execution if necessary to stay within a given power or cooling envelope.

#### 8.2.4.7 Refresh Control

The refresh interval is programmable by setting a 15 bit refresh counter. At the maximum address frequency of 400 MHz (250 ps), 3120 clocks are needed to achieve the nominal refresh interval of 7.8  $\mu$ s. The maximum interval in this case will be 81.9  $\mu$ s. The refresh and MemBIST state machines will typically be implemented as separate state machines, allowing refresh when MemBIST is not active. In normal operation refresh commands are issued from the host rather than the AMB.

**Table 8-7 — Refresh Programming**

		Spec ( $\mu$ s)	Clk Period ( $\mu$ s)	Count
<b>tREFI (15 bits)</b>	min	0	0.0025	1
	spec	7.8	0.0025	3120
	max	81.9	0.0025	32768

#### 8.2.4.8 DRAM Mode Registers

Several MemBIST operating parameters are defined in the DRAM mode registers. It is the responsibility of the user to establish corresponding conditions in the MemBIST registers. One example is CL, which is defined in the DRAM mode register and in the MemBIST control register. Behavior is unpredictable if these values do not match.

#### 8.2.4.9 DRAM Initialization

As in other operating modes, MemBIST requires the DRAM be initialized prior to the start of test. DRAM initialization is accomplished by starting the initialization engine (using the DCALCSR register). This may be done in band or out of band.

### 8.2.5 Algorithmic Testing

MemBIST supports several of the more common algorithmic tests. Several of the traversal methods are directed at basic operation such as initializing memory or verifying proper connectivity of the AMB and DRAM on a DIMM. In addition there are a few tests that are intended to perform testing of the AMB address generators and DRAM internal address decoders, multiplexers and related logic that is not otherwise testable at speed.

## 8.2.5 Algorithmic Testing (cont'd)

The examples below use the following notation:

- $\wedge$  = increasing addressing. Addresses will be counted up, starting at 0,0 or the user-defined start address.
- $\vee$  = decreasing addressing. Addresses will be counted down, starting at the end of the array or from the user-defined start address.
- r or w = Read or Write
- D or I = Data or Inverted Data
- Number = sequence of events
- ( ) = back to back operations. Example: (wD, rD) will read then write the same cell before moving to the next cell.

### 8.2.5.1 Initialization Tests

Init:  $\wedge(wD)_1$

This is a simple memory initialization algorithm. Data will be written to the array with incrementing addressing.

Read:  $\wedge(rD)_1$

This test is used to verify memory contents. One use of this is data retention testing. Init may be used to write known data in the array. The tester or system can then alter an environmental condition, or simply wait for some period of time, and then read the array to see if the data changed.

Scan:  $\wedge(wD)_1; \wedge(rD)_2; \wedge(wI)_3; \wedge(rI)_4$

The scan test writes data to the array then reads the data back. The second half of the test writes inverted data and reads it back. Scan is a 4N pattern, meaning test time will be 4 traversals, times the size of the array (rows \* columns \* banks, also known as 'N') times the average time a read or write operation.

### 8.2.5.2 Memory Stress Tests

Mats+:  $\wedge(wD)_1; \wedge(rD)_2; wI_3; v(rI_4, wD_5)$

The Mats+ algorithm initializes the array to a known data background and steps through with a read-write sequence. The algorithm is performed with both incrementing and decrementing addressing. Mats+ will detect stuck at faults and basic address decoder faults. The algorithm is order 5N.

MarchC-:  $\wedge(wD)_1; \wedge(rD)_2; wI_3; \wedge(rI_4, wD_5); v(rD_6, wI_7); v(rI_8, wD_9); v(rD)_{10}$

The MarchC- algorithm initializes the array to a known data background and steps through the array in both count-up and count-down addressing with a read-write sequence. MarchC- tests the array decoders and basic neighboring faults. As might be determined from the sequence numbering this is a 10N pattern.

## 8.2.6 DRAM Operations not Supported

MemBIST does not support the following DRAM operations:

- Auto-precharge
- Burst interrupt
- Precharge all

## 8.2.7 Quad Rank Support

Quad rank requires two additional rank select bits. The existing rank select already included 2 bits, but used as a 01 and 10 encoding. The 00 and 11 encodings are used for ranks 2 and 3.

NOTE In MEMBIST, the standard mapping of bank addressing applies. The Mode C remapping of BA0 does NOT apply in MEMBIST.

## 8.2.8 MemBIST Flow Control FSM

This FSM controls the MemBIST flow and generates read/write commands for MemBIST. This is only one example of how this functionality might be implemented.

- When MBCSR bit[31] is programmed to kick off write/read sequence, MemBIST FSM will transit out of IDLE state to either WR\_START or RD\_START. It depends on the MemBIST command execution.
- In WR\_START state, FSM will look at the decoding of DATA type selection. If LFSR data type generation is selected, FSM will go to WR\_SEED state. If not, FSM will directly go to WR\_NXTAD state.
- In WR\_SEED state, MemBIST will create 5 crc32 data sets and load into MBDATA4/5/6/7/9 from the initial seed register MBLFSRSED. When 5 sets of random data is set and loaded into MBDATA, FSM will transit out of this state to WR\_NXTAD state.
- In WR\_NXTAD state, next address of write command will be issued and transit to WR\_AVAIL state immediately.
- FSM will toggle between WR\_NXTAD and WR\_AVAIL until all writes are issued. In WR\_AVAIL state, write command will be issued. Whenever previous cycles DRAM timing meets the requirement and not the last address, FSM will transit to WR\_NXTAD. If this is the last address and timing meets, FSM will go to WR\_WAIT state.
- In WR\_WAIT state, FSM will wait for the last write timing qualified and transit do WR\_DONE state.
- If this is a write only test, then FSM will go back to IDLE state. If this is a write with read comparison test, FSM will go to RD\_START state.
- In RD\_START state, FSM will look at the decoding of DATA type selection. If LFSR data type generation is selected, FSM will go to RD\_SEED state. If not, FSM will directly go to RD\_NXTAD state.
- In RD\_SEED state, MemBIST will create 5 crc32 data sets and load into MBDATA4/5/6/7/9 from the initial seed register MBLFSRSED. When 5 sets of random data is set and loaded into MBDATA, FSM will transit out of this state to RD\_NXTAD state.
- In RD\_NXTAD state, next address of write command will be issued and transit to RD\_AVAIL state immediately.

### **8.2.8 MemBIST Flow Control FSM (cont'd)**

- FSM will toggle in between RD\_NXTAD and RD\_AVAIL until all writes are issued. In RD\_AVAIL state, write command will be issued. Whenever previous cycles DRAM timing meets the requirement and not the last address, FSM will transit to RD\_NXTAD. If this is the last address and timing meet, FSM will go to RD\_WAIT state. If this is a back to back read/write test, FSM will transit from RD\_AVAIL state to RD\_NXTWR state.
- In RD\_NXTWR state, next address of write command will be issued and transit to RD\_WRAVL state immediately.
- If this is the last address in RD\_WRAVL state, FSM will go to RD\_WAIT state. If not, FSM will go back to RD\_NXTAD state again.
- In RD\_WAIT state, FSM will wait for the last read or write timing qualified and transit to RD\_DONE state.
- In RD\_DONE state, FSM will always go to IDLE state.



## 8.2.8 MemBIST Flow Control FSM (cont'd)

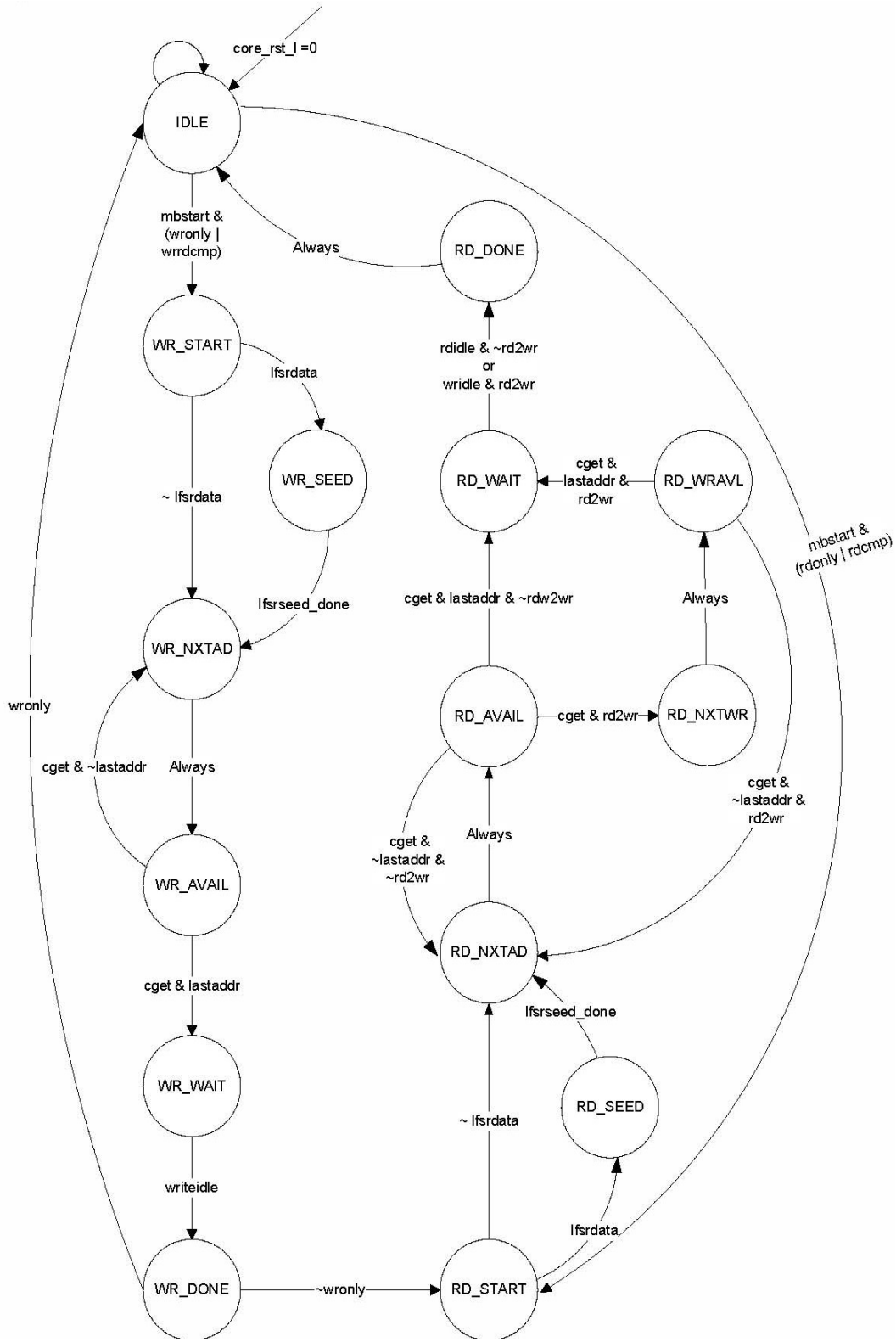
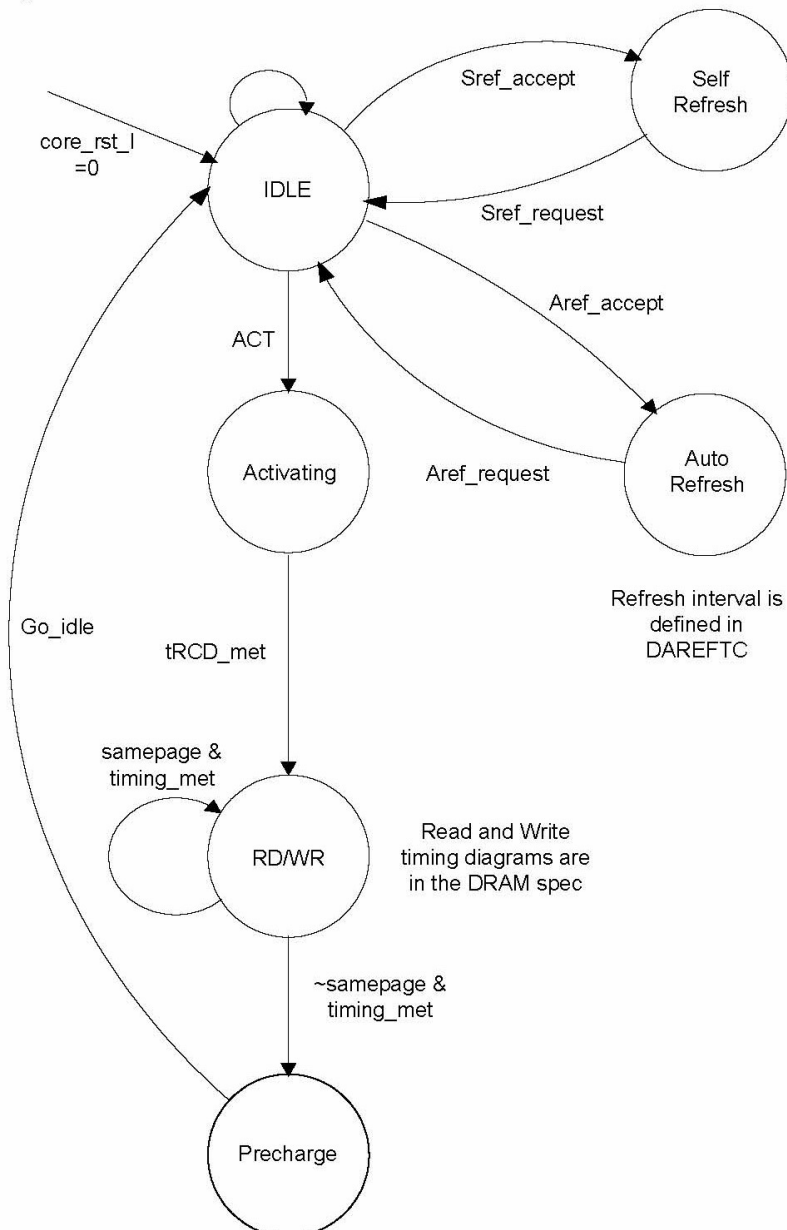


Figure 8-11 — MemBIST Flow Control State Machine

### 8.2.9 MemBIST CSFSM

This FSM creates DRAM commands for MemBIST.

- When read or write command is available and tRP/tRC timing is met, MemBIST CSFSM will transit out of IDLE state to ACT state.
- In ACT state, FSM will wait for tRCD timing parameter qualified and go to RDWT state.
- If the next coming read or write command is in same page and DRAM timing is qualified, FSM will be loop in this state. If the next command is not in the same page or there is an auto-refresh/self-refresh request, FSM will go to PRECHARGE state.
- In PRECHARGE state, FSM always goes back to IDLE state.
- In IDLE state, if there is a self refresh or an auto refresh request, FSM will wait for self refresh logic accept signal or auto refresh accept signal.



**Figure 8-12 — MemBIST Command State Machine**

## 8.2.10 MemBIST Feature Summary

As illustrated in Table 8-6, there is a minimum set of features that must be supported in a conforming MemBIST implementation. The objective is to define a standard set of features, registers and programming model for all AMBs. Additional features may be added at the AMB manufacturers discretion but these will be supplier specific.

**Table 8-8 — MemBIST Features**

<b>Address Patterns</b>	
X address bits	16
Y address bits	15
Z address (bank) bits	3
Address pattern in tests	Used defined start/end address with incrementing or decrementing FastX, FastY, FastXY support.
<b>Data Patterns</b>	
Fixed data pattern	fixed nibble data patterns (0, 3, 5, 6, 9, A, C, F)
User defined data pattern	144 or 288 bits of user defined data.
Circular data pattern	144 bit circular shift register
LFSR	LFSR (crc32)
Failure status	Pass/Fail indicator bit
Data Error logger	Last (most recent) 144 bit failure data
Error Accumulator	144 bit lane failure data accumulator
<b>DRAM Timing control</b>	
CL, tRCD, tRP, tWR, tRC	Various DDRII DRAM timings as described below
Burst Length	4 or 8
Refresh control	Programmable refresh interval (7.8mS nominal); refresh enable or disable.
<b>Execution control</b>	
Access method	all registers and settings available in band or out of band
DRAM data width	x4 or x8
DRAM initialization and mode registers	statically set by host
Pattern execution method	SMBus or Host
Failure data access	Failing data is available for read out by the host/SMBus at completion of test
Ignore failure	Ignore failures up to a specified location
Termination	Halt on error or run to completion of test
Abort	Test abort during the test
<b>Algorithm support</b>	
Scan	^(wD)1; ^(rD)2; ^ (wl)3; ^ (rl)4
Init	^(wD)1
Read	^(rD)1
Mats+	^(wD1); ^(rD2, wl3); v(rl4, wD5);
MarchC	^(wD1); ^(rD2, wl3); ^ (rl4, wD5); v(rD6, wl7); v(rl8, wD9); v(rD10);

## 8.2.11 MemBIST Registers

This clause includes more information on the data types, addressing modes and failure information available. For specific bit locations within a register refer to the FBDIMM AMB standard.

### 8.2.11.1 Memory Technology Register

The memory technology register (MTR) contains basic information about the DIMM and the DRAMs mounted on the DIMM. The row and column values will reflect the externally visible organization of the DRAM. For example, a 512 Mb DRAM with 8 bit data bus would have 14 row addresses (4K rows), 2 bank addresses (4 banks), and 11 column addresses (2K columns).

**Table 8-9 — Memory Technology Settings**

Description	Supported values
Number of ranks on the DIMM	1 (single rank) or 2 (dual rank)
DRAM data width	4 or 8 bits
DRAM Banks	4 or 8 banks
DRAM Rows	8K, 16K, 32K, 64K
DRAM Columns (external column addresses)	1K, 2K, 4K, 8K

### 8.2.11.2 DRAM Timing Control

The DRAM timing control register (DRT) controls the timing of cycles issued by the MemBIST engine. Since MemBIST is not a general purpose memory controller only the timings necessary for the supported DRAM cycle types and traversal methods are specified.

**Table 8-10 — DRAM Timing Values**

Param	Description	Target Range	Register Settings
	Read-Write delay, normally $BL/2 + 2 t_{CK}$		3 to 10 clocks
	Write-Read delay, normally $(CL-1)+BL/2+t_{WTR}$		5 to 12 clocks
t <sub>WR</sub>	Write recovery time		2 to 9 clocks
t <sub>RC</sub>	Activate to activate command period	57-60 ns	11 to 26 clocks
t <sub>RCD</sub>	Activate to read or write delay	12-15 ns	3, 4, 5 or 6 clocks
t <sub>RP</sub>	Precharge command to Activate command t <sub>CK</sub> delay	12-15 ns	3, 4, 5 or 6 clocks
t <sub>RAS</sub>	Active to precharge delay (minimum)	40-45 ns	12, 15 or 18 clocks
t <sub>RTP</sub>	Internal read to precharge delay	7.5 ns min.	2, 3, 4, or 5 clocks
CL	CAS Latency		3, 4, 5 or 6 clocks
BL	Burst Length		4 or 8

Using the above timings with various clock frequencies it is possible to create a range of DRAM timings. For example, t<sub>RCD</sub> might vary from 11 to 20 ns depending on the number of clocks selected and the clock cycle time.

### **8.2.11.3 MemBIST Control Register**

The MemBIST control register (MBCSR) determines the nature of the test to be executed. Some of the key capabilities specified in this register are addressing (fast X, fast Y or fast XY) and direction (incrementing or decrementing), the data pattern to be used, the traversal method or algorithm desired and use of the error logging features in MB\_DATA and MB\_ERR\_DATA.

### **8.2.11.4 Memory Data Register**

The memory data register (MB\_DATA) is a 320 bit register (288 + 32) for general purpose storage of memory data. The register can be used multiple ways depending on the current operating mode of the MemBIST engine. The register modes are:

Fixed data pattern, circular shift or LFSR: When running one of the fixed data patterns the register holds the most recent failure DQ bits (144 bits) or the failing addresses (up to 5\*32 bits).

144 bit user defined data: In this mode the user writes in the 144 bits of data to be written to the DRAMs.

288 bit user defined data: In this mode the user will specify 288 bits of data. There is no reporting of the 288 bit failure data.

Furthermore, the user data, circular shift and LFSR modes can report failing addresses or failing data.

In failure data mode, the failure bits are “sticky”. In this way the bits will record all DQ’s that failed at some time during the test. This allows us to run a test from start to end without stopping on failures. At the completion of test, read the failure register to determine which bits and therefore which DRAMs are failing.

#### 8.2.11.4 Memory Data Register (cont'd)

**Table 8-11 — Memory Data Register**

Register	Fixed Data	User Defined Data, Failure Address Mode	User Defined Data, Failure Data Mode	Circular Data, Failure Address Mode	Circular Data, Failure Data Mode	LFSR, Failure Address Mode	LFSR, Failure Data Mode	288 Bit User Data
MBDATA 9 [31:0]	Last Failing Address	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
		Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
		Late User Data [71:63]	Late User Data [71:63]	Late Circ Data [71:63]	Late Circ Data [71:63]	LFSR Data	LFSR Data	User Data Word 4 [71:63]
		Early User Data [71:63]	Early User Data [71:63]	Early Circ Data [71:63]	Early Circ Data [71:63]			User Data Word 3 [71:63]
MBDATA 8 [31:0]	Undefined	Last Failing Address	Undefined	Last Failing Address	Undefined	Last Failing Address	Undefined	Undefined
	Undefined		Undefined		Undefined		Undefined	Undefined
	Late Error Data [71:63]		Late Error Data [71:63]		Late Error Data [71:63]		Late Error Data [71:63]	User Data Word 2 [71:63]
	Early Error Data [71:63]		Early Error Data [71:63]		Early Error Data [71:63]		Early Error Data [71:63]	User Data Word 1 [71:63]
MBDATA 7 [31:0]	Failure Address 4	Late User Data [63:0]	Late User Data [63:0]	Late Circular Data [63:0]	Late Circular Data [63:0]	LFSR Data	LFSR Data	User Data Word 4 [63:0]
MBDATA 6 [31:0]	Failure Address 3							
MBDATA 5 [31:0]	Failure Address 2	Early User Data [63:0]	Early User Data [63:0]	Early Circular Data [63:0]	Early Circular Data [63:0]			User Data Word 3 [63:0]
MBDATA 4 [31:0]	Failure Address 1							
MBDATA 3 [31:0]	Late Error Data [63:0]	Failure Address 4	Late Error Data [63:0]	Failure Address 4	Late Error Data [63:0]	Failure Address 4	Late Error Data [63:0]	User Data Word 2 [63:0]
MBDATA 2 [31:0]		Failure Address 3		Failure Address 3		Failure Address 3		
MBDATA 1 [31:0]	Early Error Data [63:0]	Failure Address 2	Early Error Data [63:0]	Failure Address 2	Early Error Data [63:0]	Failure Address 2	Early Error Data [63:0]	User Data Word 1 [63:0]
MBDATA 0 [31:0]		Failure Address 1		Failure Address 1		Failure Address 1		

#### 8.2.11.5 Memory Failure Register

The memory error register (MB\_ERR\_DATA0, 1, 2, 3, 4, 5) register holds 144 bits of error data in five 32 bit registers. MB\_ERR\_DATA 0 and 1 contain early error data[63:0], MB\_ERR\_DATA 2 and 3 contain late error data[63:0]. MB\_ERR\_DATA 5 [7:0] contains early error data[71:64] and MB\_ERR\_DATA 5[15:8] contains late error data[71:64]. This data will be overwritten by subsequent failures, if any. To accumulate failing data bits, use MB\_DATA.

#### 8.2.11.6 LFSR Seed Register

The LFSR seed register (MB\_LFSR\_SEED) holds a 32 bit seed for the random data generator. To use this seed, the random data type must be selected in the control register. If no value is programmed MemBIST will use a default value of 0000'h.

### 8.2.11.7 Circular Shift Register

Rotating data patterns are needed for certain memory tests. One of the applications is walking a 1 or 0 across the DQ bus to detect open or shorted data lines during DIMM manufacturing test. Another anticipated use would be creating rotating or diagonal data patterns to test the DRAM decoders or array.

MemBIST provides a 144 bit circular shift register to allow generation of shifting or rotating data patterns. Rather than requiring the user to load the entire 144 bits, the LFSR seed register is used to specify a 32 bit starting pattern. The 32 bit pattern will be loaded into the least significant 32 bits (early\_data[31..0]) of the shift register at the start of test. The upper bits of the register will be cleared to all zeros at the start of test. To generate '1' data, set the data invert bit. The register will shift one position to the left every time new data is required. For example, when BL=4 the register will shift once for the first 144 bits and again for the second 144 bits of the burst. The 144 bits are organized as follows: Late ECC[7..0] Late Data[143..0] Early ECC [7..0] Early Data[143..0] <= shift direction.

### 8.2.11.8 Random Data Generator

Random patterns are useful in situations where DRAM topology is not known. A common usage would be to write and read the entire array with random data. Random data may be combined with various addressing modes to create a variety of pseudo-random tests. By default MemBIST does not support random address generation so the test will not be completely random.

The random data generator is a 32-stage LFSR conforming to the IEEE 802.3 (Ethernet) standard. Specifically the LFSR implements the polynomial:

$$P(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

At the start of random data generation the LFSR is seeded with 32 bits from the seed register as described above. These 32 bits are also copied to the least significant bits of a 144 bits data register (MB\_DATA). The upper bits of MB\_DATA are loaded with user-defined data. On subsequent updates the 144 bit data register is shifted one bit to the left and the lowest 32 bits are replaced with the next CRC code determined by: invert crc32(invert(bits 31:0)). Other implementations are possible, provided the test setup is the same and the result is a pseudo-random bit stream using the above polynomial.

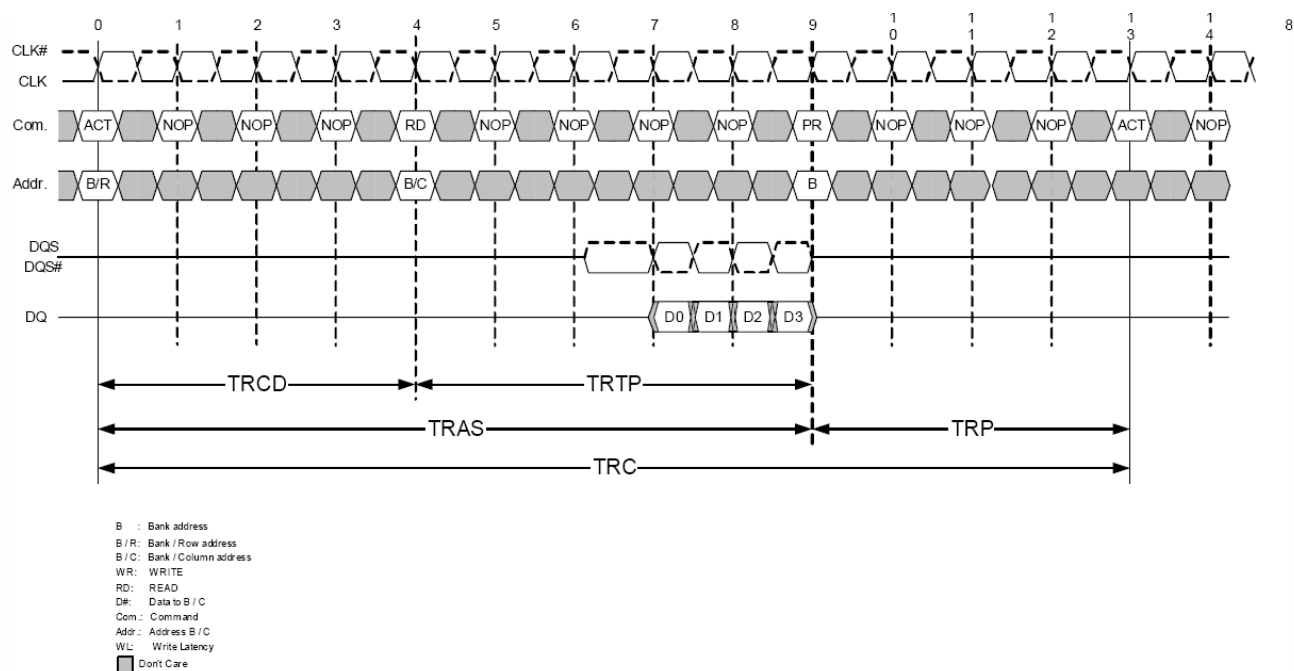
The value of the seed register and MB\_DATA registers at completion of an LFSR test sequence is implementation specific. If the test is stopped (either by normal termination or an error) and then restarted (e.g. to locate the next error) the AMB must continue with the next number in the LFSR sequence. To reset the LFSR, write a new seed in the seed register.

In most cases the Scan, Mats+ and MarchC- algorithms will be incompatible with random data. These algorithms require resetting the LFSR seed in the middle of the test or generating different LFSR sequences for back to back write/read traversals. There is no standard mechanism to alter LFSR generation in this way.

## 8.2.12 MemBIST Timing Control

MemBIST provides the capability to control the DRAM timings by setting the AMB DRT register. The settings of the timings in this register directly influence the execution of the MemBIST algorithms. Rather than just issuing RD and WR accesses, the MBIST state machine has to generate ACT and PR commands to control the data exchange to and from the DRAMs following the DDR2 SDRAM command protocol. To be able to do a more or less critical testing the pattern execution can be controlled in respect to the distance of the active commands on the DRAM command bus separated by NOP states. Data and address follow the commands accordingly.

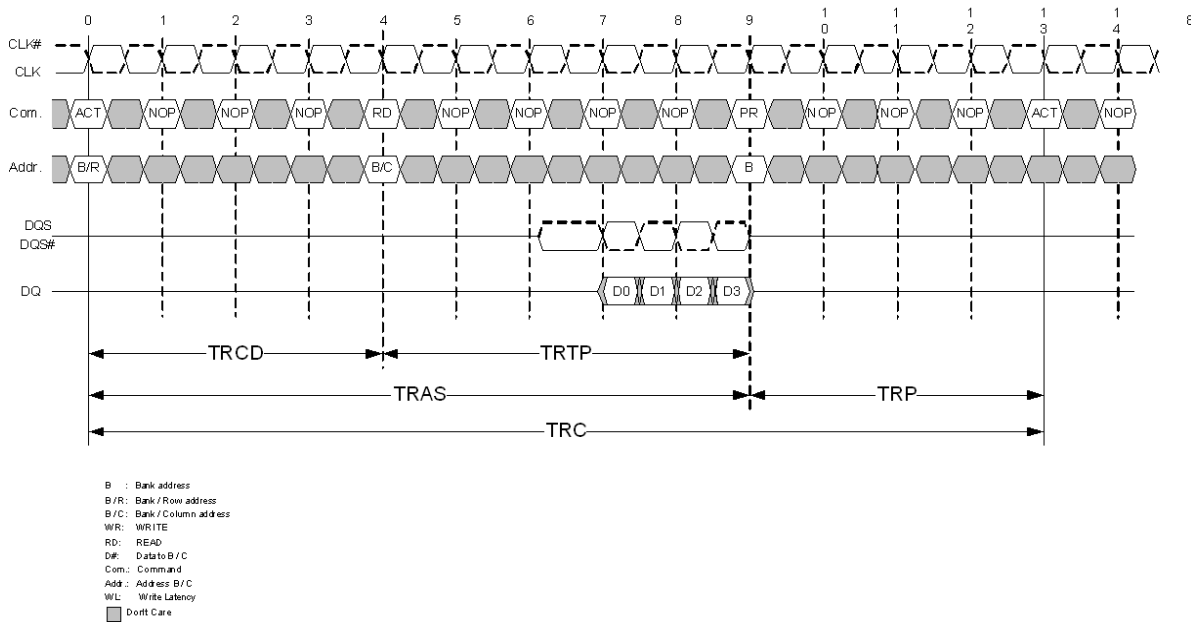
The parameters TRAS, TRTP, BBRW, BBWR, TWR, TRC, TRCD, TRP and NOPCNT, all specified in the DRT register can be varied in multiples of the clock cycle. The definition complies with the DRAM DDR2 memory component specification and indicated in figures 6-13, 6-14 and 6-15. These timing parameters have to be fulfilled for the regular AMB DRAM access as well as for stimulating the DRAM access by means of algorithmic MemBIST pattern at application near production testing. Due to this fact, it is necessary to know the relation between the DRT settings and the MemBIST pattern execution.



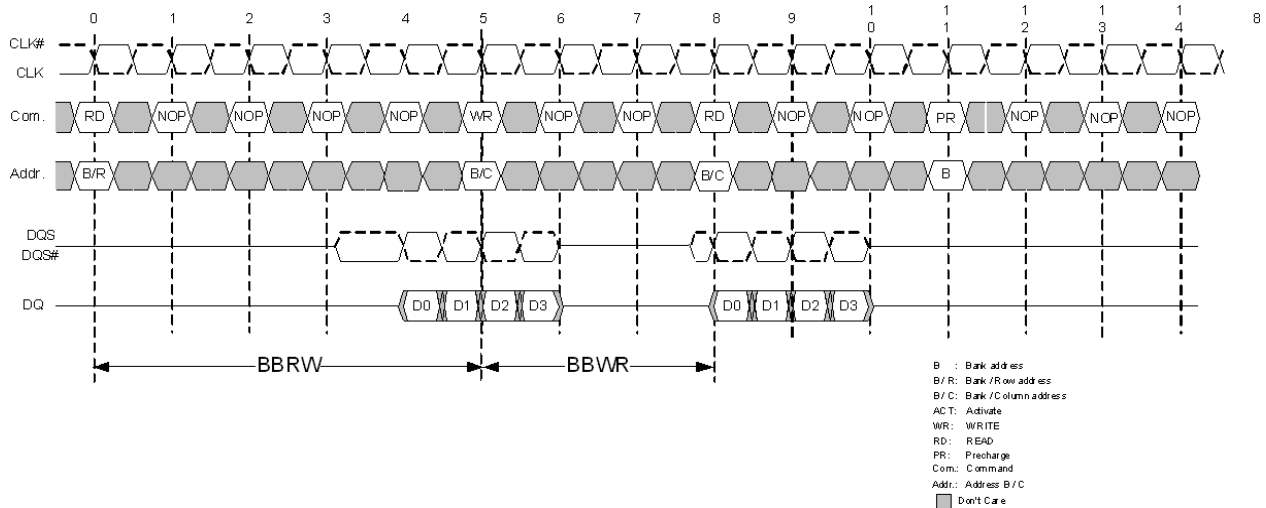
**Figure 8-13 — DRAM Write Timings (WL=2, AL=0)**



## 8.2.12 MemBIST Timing Control (cont'd)



**Figure 8-14 — DRAM Read Timings (CL=3, AL=0)**

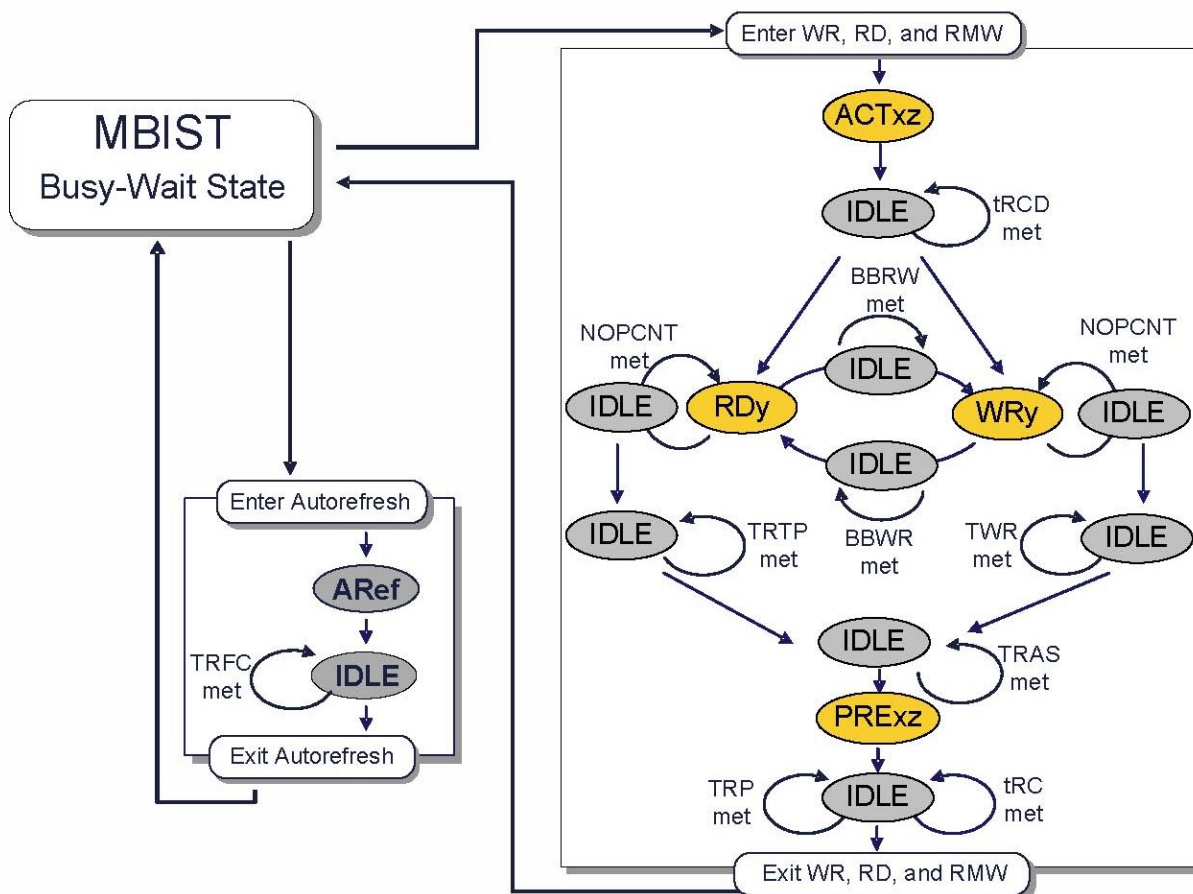


**Figure 8-15 — Read-Write-Read Timings (CL=4, AL=0, WL=3)**

Figure 8-16 shows the relationship between the operation of the finite state machine (FSM) and the timing sequence of a single MBIST access. The diagram displays the DRT's register settings depending on the MBIST execution state. Three kinds of operation modes during MBIST operation can be distinguished:

- A no operation mode called MBIST Busy-Wait State.
- A self-refresh state that is entered automatically in defined intervals for single refreshes.
- An auto refresh state that is entered in defined intervals for single refreshes, and
- A WR-RD-RMW state that is the elementary timing sequence for any sequence of MBIST pattern accesses to the DRAM.

## 8.2.12 MemBIST Timing Control (cont'd)

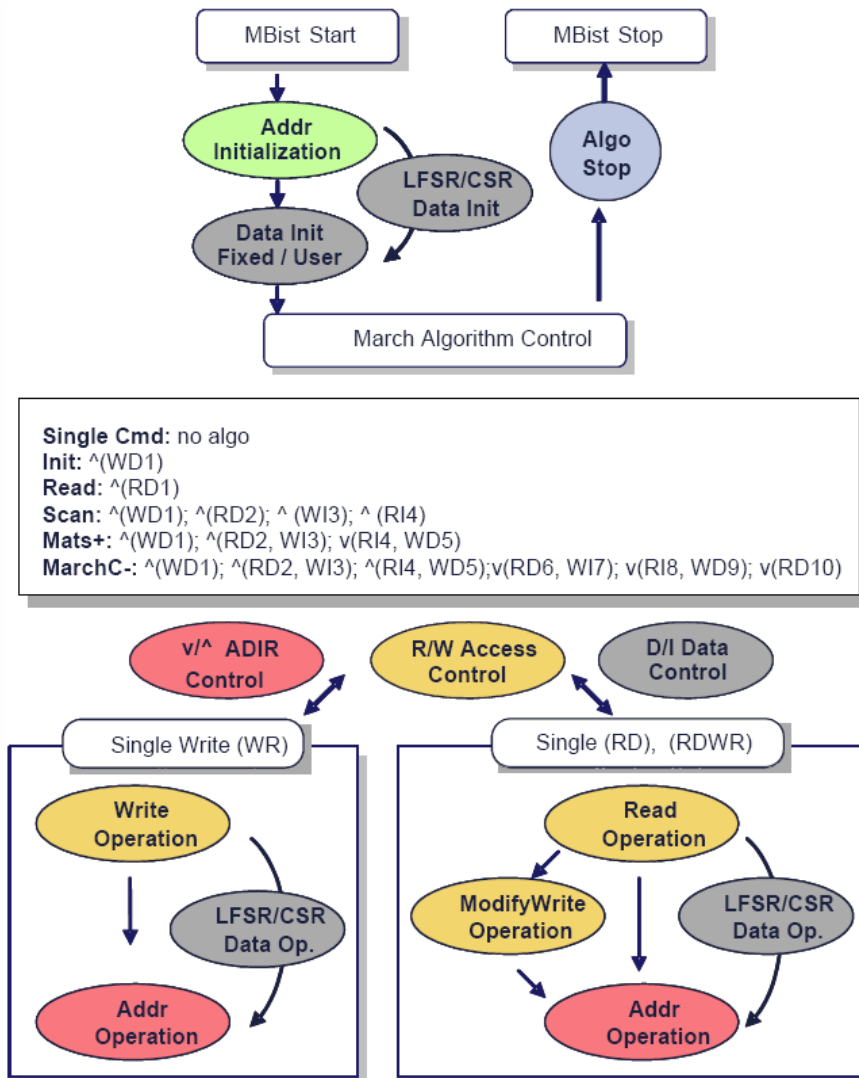


**Figure 8-16 — Relation between DRAM Timing Control Parameters and MemBIST Functionality**

For auto-refresh operation the refresh cycle time TRFC needs to be fulfilled when exiting the refresh mode. With auto-refresh enabled the AMB's refresh controller has to issue refresh interrupts in periods of TREF.

The WR-RD-RMW state is entered with an ACT command on a row x and bank z of the DRAMs. After fulfilling a time TRCD, a WR or a RD command on column address y can be issued by the MBIST engine. Depending on the algorithm selected (i.e. Init, Read, March) and the address sequencing chosen (i.e. x-fast, diagonal, y-fast, bank interleaved), a single write or read burst is issued, a single read-write burst access is being performed, or a write/read burst may be followed either by following write/read bursts during page oriented access. To slow down the data bus traffic caused by RD and WR the number of NOPCNT are inserted in between. BBRW and BBWR have to be fulfilled in a sequence of y fast march accesses. Before stopping a DRAM row access by a PRE command, either TRTP or TWR have to be met, followed by the requirement of the minimum TRAS. The next ACT can be performed after fulfilling TRP and TRC timing condition after issuing the PRE.

## 8.2.12 MemBIST Timing Control (cont'd)



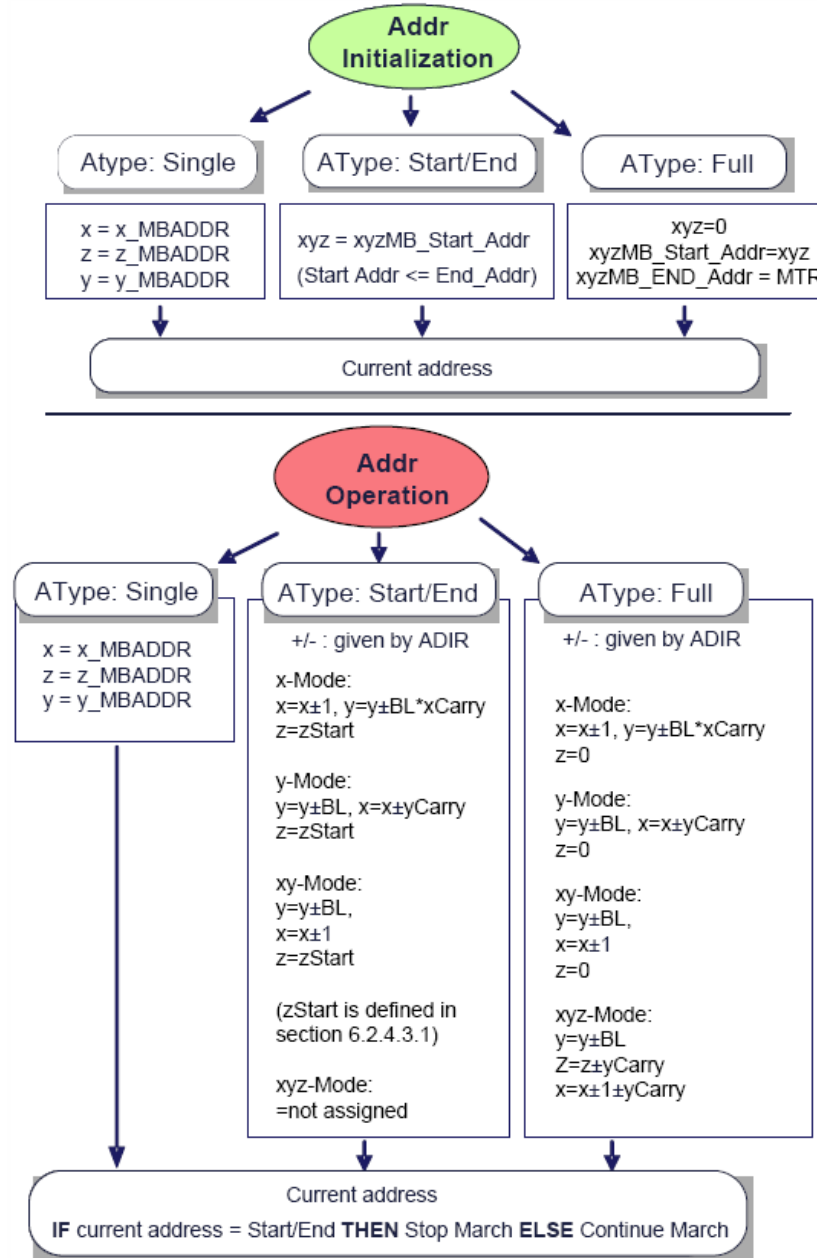
**Figure 8-17 — Schematic Representation of the MemBIST Read/Write Functionality**

The FSM of the MBIST algorithm control engine is schematically explained in Figure 8-17 and Figure 8-18. When starting MBIST, first an address initialization is completed. Depending of the AType setting in the MBCSR, the start address is being set, influenced by MB\_START\_ADDR, MB\_END\_ADDR, and the MTR register. During MBIST execution, the current address is modified according to the address-sequencing mode selected. In addition, the data path is initialized by fixed or user defined values. In case of LFSR or CSR control, the data is modified.

The algorithm control supports simple access pattern and more complex march pattern. During algorithm execution the address direction (^ or v), the data (D or I) and the access type (W, R, or RW) is controlled. Two higher-level algorithmic access types during MBIST execution can be distinguished:

- Single write operations (Init)
- Single read operations (Read) or read-modify-write operations (R, W), as executed in march patterns Mats+ and MarchC-.

## 8.2.12 MemBIST Timing Control (cont'd)



**Figure 8-18 — Address Initialization and Sequencing during MemBIST Operation**

In the first case, each write operation is followed by an address operation. The timing sequence is fulfilled as shown Figure 8-16. In the second case, at least a read is issued, that might be followed by a write command. After lone read as well as read-modify-write accesses, an address operation follows.

In case of performing a LFSR or CSR operation, the user defined data content is modified by pseudo random scrambling or circular shift right operation, respectively, after each single write or read command. MBIST stops when the start/end address is reached during pattern operation. FBDIMM DFx Spec 1.0

---

## 9 System Test

---

### 9.1 Overview

This specification describes a general Fully Buffered DIMM high speed link margin test capability. The tests provide the capability to margin transmit drive strength and receive eye timing. The voltage margin test is a required AMB feature, targeted to test a long, lossy channel. The timing margin test is a recommended AMB feature, targeted to test a short channel. The minimum required margin capability provides sufficient range to test to link failure in most cases. Margin tests are implemented and operate the same across all lanes in the channel. Additional timing margin bits are available for added granularity in more robust implementations. Unimplemented optional bits will be ignored by the AMB. Margin tests will be supported during normal channel initialization and operation with any workload, or while running IBIST, with or without MemBIST tests, to achieve maximum test flexibility. It is recommended that margin tests be controlled through the SMBus interface since normal channel register access may be unreliable during margin tests.

### 9.2 Voltage Margining

Three required normal operation NB and SB transmitter drive strengths are defined as “large”, “regular” and “small” and controlled by function 1, offset 54h, 55h (SBTXDRVNXT, NBTXDRVNXT). Voltage margining is enabled with additional bits in the same NB and SB transmit drive strength registers (SBTXDRVMAR, NBTXDRVMAR). Voltage margin levels are specified as minimum values and are not calibrated. Voltage margin levels will be binary monotonic. Smaller normal operation drive strengths may also be used as additional margin levels for larger drive strengths. Other register bits separately set the NB and SB drive de-emphasis levels and are unchanged in operation during normal and margin operation. Transmitter common mode voltage levels will remain above the minimum specification for the small voltage swing during voltage margin tests to preserve proper receiver operation and not create a false Electrical Idle detect. Link Electrical Idle detect should not be disabled during margin tests. New margin settings only take effect after the channel comes out of fast reset (channel Electrical Idle). Transmitters are not required to dynamically adjust drive strengths on the fly. Desired voltage margin settings are loaded into FBDSBCFGNXT and FBDNBCFGNXT, function 1, offsets 54h and 55h.

### 9.3 Timing Margining

The timing margin feature will introduce a sampling error at the receiver by a + or – amount. Registers in function 1, offset 5Ah and 5Bh specify the offset. If supported, the maximum error will be at least 0.25 UI, and ideally 0.35 UI. The error can be implemented as a static, single-ended error value specified by the receiver timing margin offset (NBRXMAROFF, SBRXMAROFF) with the direction, early or late, specified by the offset polarity bit (NBRXMARPOL, SBRXMARPOL), or a periodic error from early to late with a minimum 40% dwell time at the specified error value. Offsets introduced by the various error steps shall be binary monotonic, but are not required to be linear. The timing margin test is expected to be used in link resynchronization mode, not in link resampling mode. Receivers are not required to dynamically adjust error offset on the fly. New margin settings only take effect after the channel comes out of fast reset. Receivers are not required to dynamically adjust timing margin on the fly.

### **9.3 Timing Margining (cont'd)**

Desired timing margin settings are loaded into FBDSBCFGNXT and FBDNBCFGNXT, function 1, offsets 5Ah and 5Bh. If timing margining is supported, a minimum implementation requires the 3 most significant error offset bits (6:4) to be implemented. Specific implementations may optionally choose to implement additional error offset granularity using some or all of the remaining 4 error offset bits (3:0).

### **9.4 Voltage, Timing Margin Support Indication**

AMB support of any and all margin features can be tested by setting target control register bits to “1” and then reading back all bits in those registers in function 1, offsets 54h, 55h, 5Ah, 5Bh. Function bits that are supported by the AMB will return “1”, unsupported function bits will return “0”. Setting and testing of these registers must be done when the high speed channel is in Electrical Idle, through the SMBus interface.

### **9.5 Margin Test Usage**

The margin tests can be expected to cause a link failure and possible data loss, so testing is recommended when the channel is not handling user data. Test conditions are configured while the channel is in Electrical Idle through the SMBus. Access to AMB registers for test failure detection should be through the SMBus. Test failures are indicated either by failure to come out of fast reset/Electrical Idle, or failures during the initialization sequence, or by any channel errors while transferring channel traffic, or any unexpected entry into Electrical Idle.

### **9.6 Register Definitions**

Margin registers are described in the AMB component specification.

---

## **Annex A — (Informative) Differences between Revisions**

---

### **A.1 Differences between JESD82-28A.01 and JESD82-28A**

1. Terminology update to replace the following:
  - Replaced “master” with “controller “, 52 places
  - Replaced “M2M” with “C2C”, 14 places
  - Replaced “slave” with “target “, 9 places
  - Figure 7-11: changed “IBIST with Master-to-Master mode” to “IBIST with Controller-to-Controller mode”
2. Update the JEDEC logo to the latest version

### **A.2 Differences between JESD82-28A and JESD82-28**

- Added new 8.1.7.6, Quad Rank operation, pg. 69,
- Based on above renumbered subclauses as needed,
- Add new Tables 8-4 and 8-5 (pgs. 69 and 70),
- Based on above renumbered tables as needed,
- Added new 8.2.7, Quad Rank Support, pg. 79,
- Based on above renumbered subclauses as needed,
- Added new information to contents and renumbered as needed.

This page intentionally left blank





---

**Standard Improvement Form****JEDEC Standard JESD82-28A.01**

The purpose of this form is to provide the Technical Committees of JEDEC with input from the industry regarding usage of the subject standard. Individuals or companies are invited to submit comments to JEDEC. All comments will be collected and dispersed to the appropriate committee(s).

If you can provide input, please complete this form and return to:

JEDEC  
Attn: Publications Department  
3103 North 10th Street  
Suite 240 South  
Arlington, VA 22201-2107

Fax: 703.907.7583

---

**1. I recommend changes to the following:**

☐ Requirement, clause \_\_\_\_\_

☐ Test method number \_\_\_\_\_ Clause number \_\_\_\_\_

The referenced clause number has proven to be:

☐ Unclear ☐ Too Rigid ☐ In Error

☐ Other \_\_\_\_\_

---

**2. Recommendations for correction:**

---

---

---

---

---

**3. Other suggestions for document improvement:**

---

---

---

---

---

**Submitted by**

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

City/State/Zip: \_\_\_\_\_

Phone: \_\_\_\_\_

E-mail: \_\_\_\_\_

Date: \_\_\_\_\_

---

