Cortex™-A9 MBIST Controller

Revision: r3p0

Technical Reference Manual

ARM®

ARM DDI 0414G (ID072711)

Cortex-A9 MBIST Controller
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
02 April 2008	A	Non-Confidential	First release for r0p0
10 July 2008	B	Non-Confidential Restricted Access	Second release for r0p0
15 December 2008	C	Non-Confidential Restricted Access	First release for r1p0
23 September 2009	D	Non-Confidential Restricted Access	First release for r2p0
27 November 2009	E	Non-Confidential	Second release for r2p0
29 April 2010	F	Non-Confidential	First release for r2p2
21 July 2011	G	Non-Confidential	First release for r3p0

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
Cortex-A9 MBIST Controller Technical Reference Manual

Preface

This preface introduces the *Cortex-A9 MBIST Controller Technical Reference Manual*. It contains the following sections:

About this book

This book is for the Cortex-A9 MBIST controller.

In this book, the generic term MBIST controller means the Cortex-A9 MBIST controller, and Cortex-A9 processor means the Cortex-A9 processor family.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this manual, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Intended audience

This book is written for hardware engineers who are familiar with ARM technology and want to use the MBIST controller to test the RAM blocks used by the Cortex-A9 processor. The AXI protocol is not specified, but some familiarity with AXI is assumed.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to MBIST technology.

Chapter 2 *Functional Description*

Read this for a description of the Cortex-A9 processor interface to the MBIST controller and MBIST testing of the data RAM and tag RAMs. Also read this chapter for a description of the timing sequences for loading MBIST instructions, starting the MBIST test, detecting failures, and retrieving the data log.

Chapter 3 *MBIST Instruction Register*

Read this for a description on how to use the MBIST Instruction Register to configure the mode of operation of the MBIST engine.

Chapter 4 *MBIST Datalog Register*

Read this for a description of the MBIST Datalog Register.

Appendix A *Signal Descriptions*

Read this for a description of the MBIST controller input and output signals.

Appendix B *Revisions*

Read this for a description of the technical changes between released issues of this book.

Conventions

Conventions that this book can use are described in:

- *Typographical* on page vi
- *Timing diagrams* on page vi
- *Signals* on page vii.

Typographical

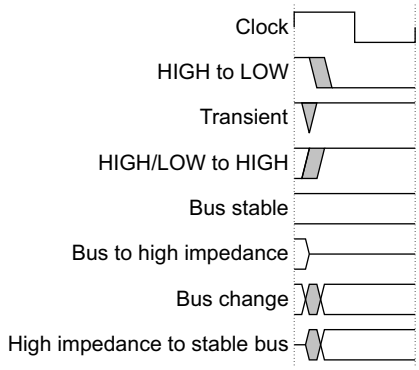
The typographical conventions are:

italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change shown in *Key to timing diagram conventions*. If a timing diagram shows a single-bit signal in this way then its value does not affect the accompanying description.

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals
- LOW for active-LOW signals.

Lower-case n At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, `http://infocenter.arm.com`, for access to ARM documentation.

See the glossary, `http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html`, for a list of terms and acronyms specific to ARM.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *Cortex™-A9 Technical Reference Manual* (ARM DDI 0388)
- *Cortex-A9 MPCore Technical Reference Manual* (ARM DDI 0407)
- *Cortex-A9 Configuration and Sign-Off Guide* (ARM DII 0146)
- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406).

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM DDI 0414G
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter describes the purpose of the MBIST controller. It contains the following sections:

1.1 About the MBIST controller

MBIST is the industry-standard method of testing embedded memories. MBIST works by performing sequences of reads and writes to the memory according to a test algorithm. Many industry-standard test algorithms exist.

An MBIST controller generates the correct sequence of reads and writes to all locations of the RAM to ensure that the cells are operating correctly. In doing this, some additional test coverage is achieved in the address and data paths that the MBIST uses. You must only use this MBIST controller with the Cortex-A9 processor to perform memory testing of the Cortex-A9 RAMs.

MBIST mode takes priority over all other modes, for example SCAN, in that the Cortex-A9 RAMs are only accessible to the MBIST controller when MBIST mode is activated.

The MBIST controller controls the MBIST testing of the Cortex-A9 RAMs through the MBIST port of the Cortex-A9 processor. Figure 1-1 shows the Cortex-A9 processor MBIST configuration.

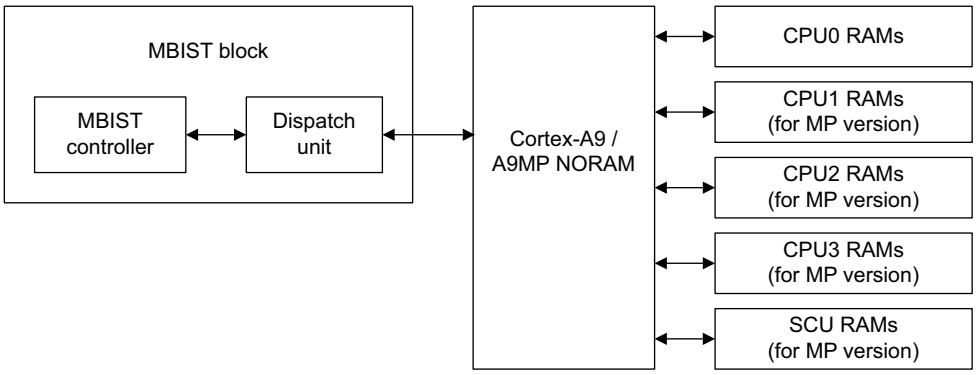


Figure 1-1 Cortex-A9 MBIST configuration

The reset signals of the Cortex-A9 processor must be HIGH in MBIST test mode. Because the MBIST test mode uses some functional paths to access RAMs, the registers on those paths must be out of reset.

See the *Cortex-A9 Technical Reference Manual* for a description of uniprocessor reset sequences.

See the *Cortex-A9 MPCore Technical Reference Manual* for a description of multiprocessor reset sequences.

1.2 MBIST controller interface

Figure 1-2 shows the nonparity configuration of the MBIST controller interface to the *Automated Test Equipmen*t (ATE) and to the MBIST interface of the Cortex-A9 processor.

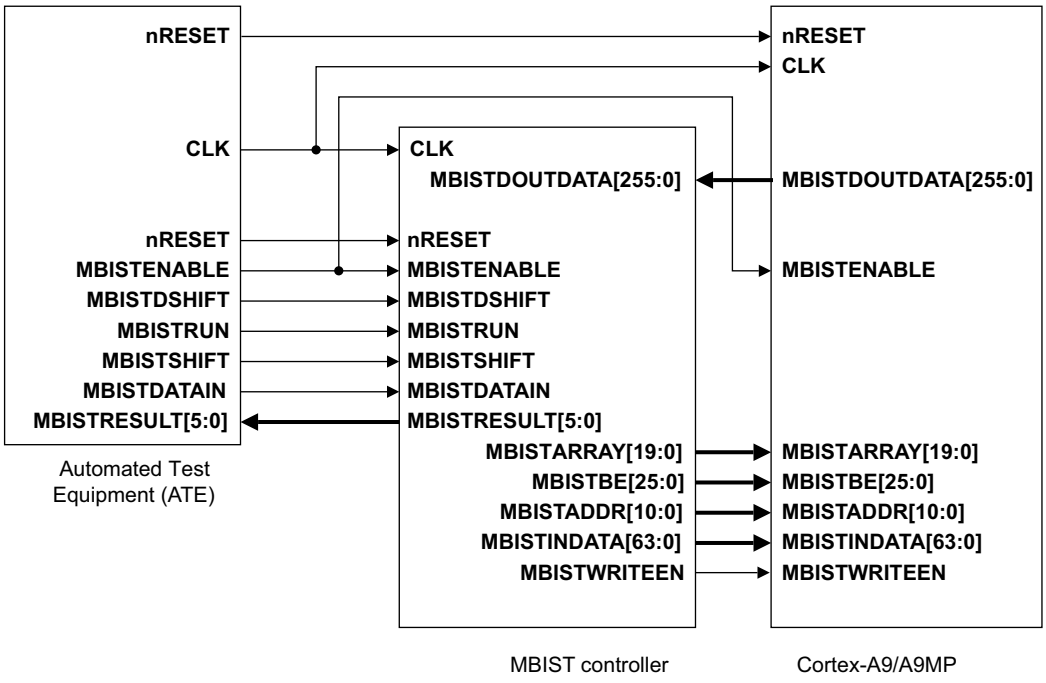


Figure 1-2 MBIST controller wiring diagram nonparity configuration

Table 1-2 on page 1-6 shows the with parity configuration of the MBIST controller interface to the *Automated Test Equipmen*t (ATE) and to the MBIST interface of the Cortex-A9 processor.

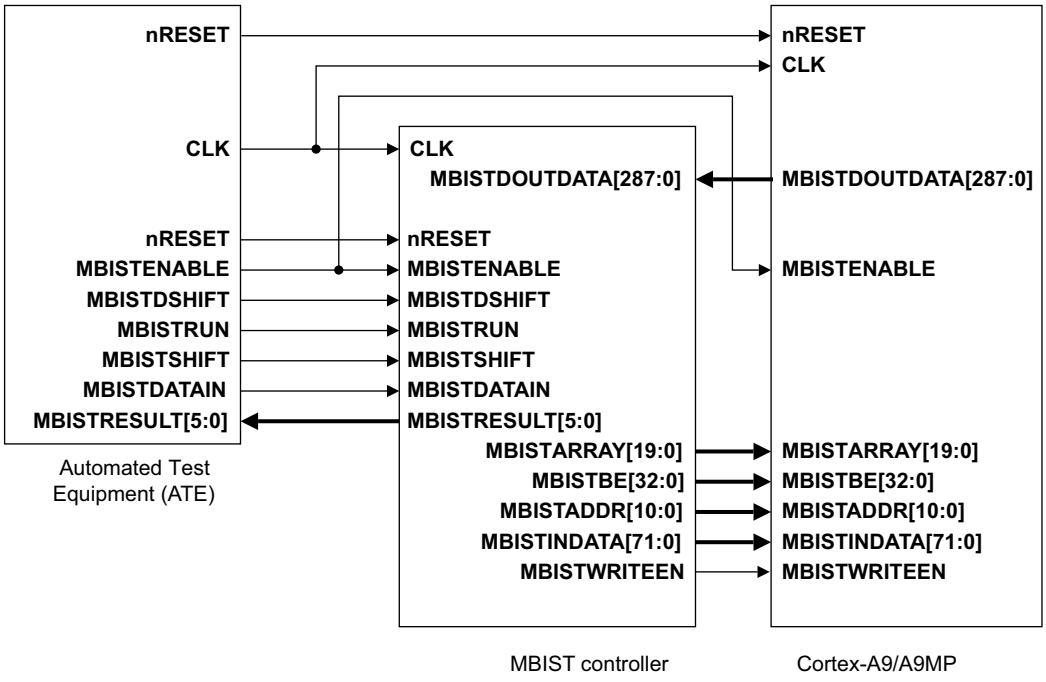


Figure 1-3 MBIST controller wiring diagram configuration with parity

Figure 1-4 shows the traditional method of accessing RAMs for MBIST.

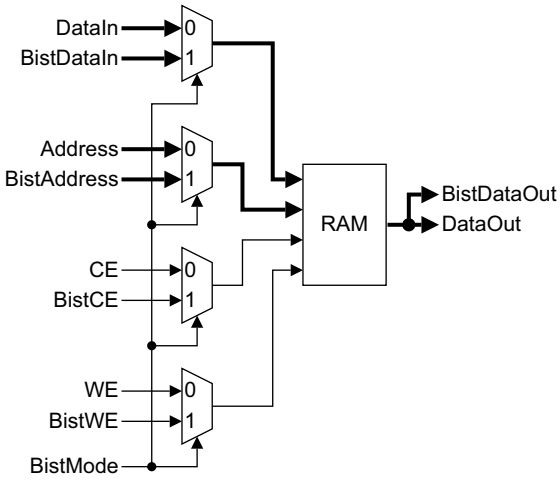


Figure 1-4 Traditional method of interfacing MBIST

Because this method significantly reduces the maximum operating frequency, it is not suitable for high-performance designs. Instead, the MBIST controller uses an additional input to the existing functional multiplexors without reducing maximum operating frequency.

Figure 1-5 on page 1-5 shows the six pipeline stages used to access the RAM arrays.

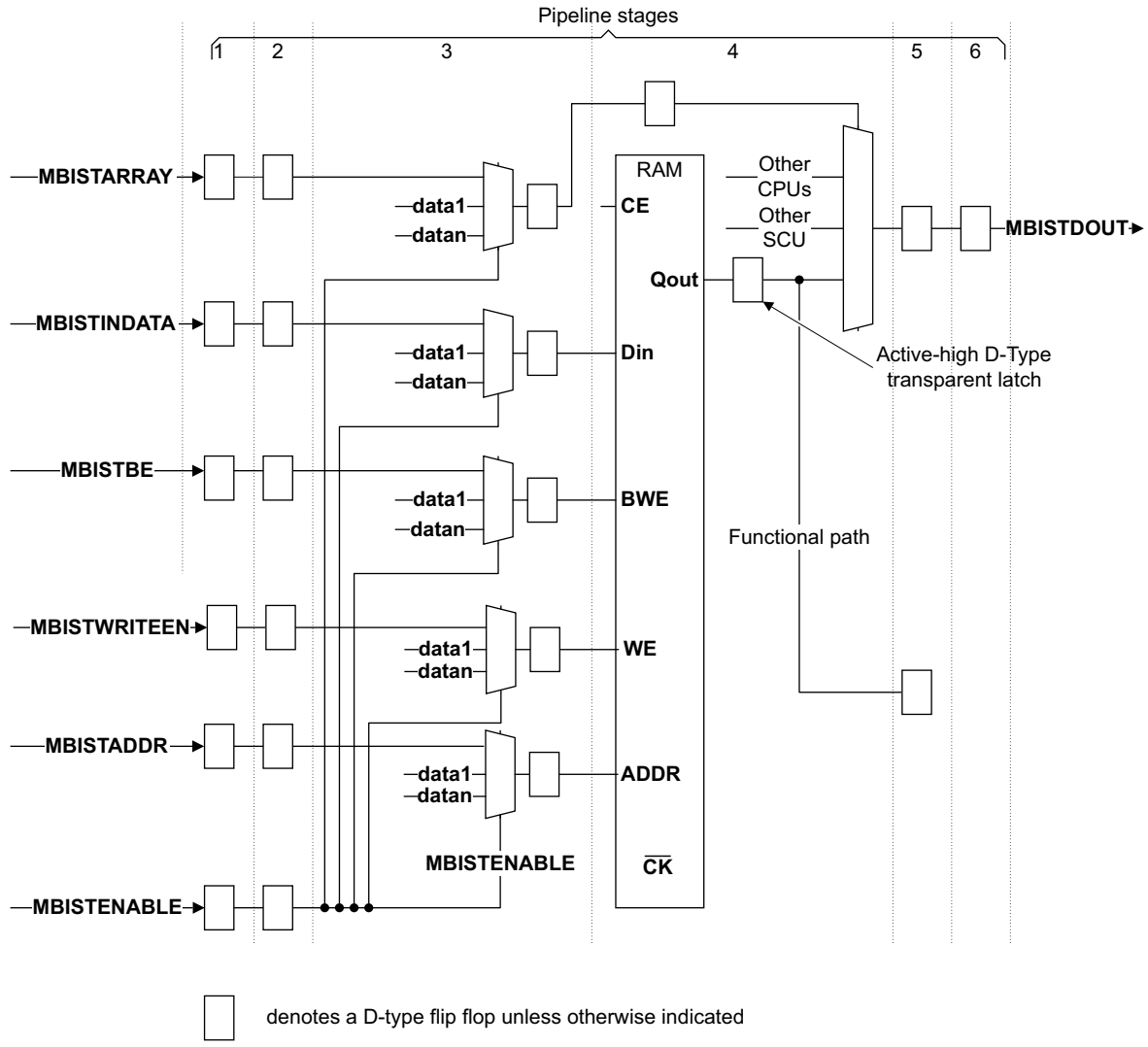


Figure 1-5 Cortex-A9 processor MBIST interface

The MBIST controller accesses memory through the MBIST interface of the Cortex-A9 processor. Table 1-1 shows the Cortex-A9 processor MBIST interface signals for a configuration without parity.

Table 1-1 Cortex-A9 processor MBIST interface signals nonparity configuration

Name	Type	Description
nRESET	Input	Global active LOW reset signal.
CLK	Input	Active HIGH clock signal. This clock drives the Cortex-A9 processor logic.
MBISTOUTDATA[255:0]	Output	Data out bus from all cache RAM blocks.
MBISTENABLE	Input	Select signal for cache RAM array. This signal is the select input to the multiplexors that access the cache RAM arrays for test. When asserted, **MBISTENABLE** takes priority over all other select inputs to the multiplexors.
MBISTARRAY[19:0]	Input	One-hot chip enables to select the RAM arrays for test.
MBISTBE[25:0]	Input	Global write enable signal for all RAM arrays.

Table 1-1 Cortex-A9 processor MBIST interface signals nonparity configuration (continued)

Name	Type	Description
MBISTWRITEEN	Input	Global write enable.
MBISTADDR[10:0]	Input	Address signal for cache RAM array.
MBISTINDATA[63:0]	Input	Data bus to the RAM arrays. Not all RAM arrays use the full data width.

Table 1-2 shows the Cortex-A9 processor MBIST interface signals for a configuration with parity.

Table 1-2 Cortex-A9 processor MBIST interface signals with parity configuration

Name	Type	Description
nRESET	Input	Global active LOW reset signal.
CLK	Input	Active HIGH clock signal. This clock drives the Cortex-A9 processor logic.
MBISTOUTDATA[287:0]	Output	Data out bus from all cache RAM blocks.
MBISTENABLE	Input	Select signal for cache RAM array. This signal is the select input to the multiplexors that access the cache RAM arrays for test. When asserted, **MBISTENABLE** takes priority over all other select inputs to the multiplexors.
MBISTARRAY[19:0]	Input	One-hot chip enables to select the RAM arrays for test.
MBISTBE[32:0]	Input	Global write enable signal for all RAM arrays.
MBISTWRITEEN	Input	Global write enable.
MBISTADDR[10:0]	Input	Address signal for cache RAM array.
MBISTINDATA[71:0]	Input	Data bus to the RAM arrays. Not all RAM arrays use the full data width.

Note

The interface of the MBIST controller communicates with both the ATE and the MBIST interface of the Cortex-A9 processor. See Appendix A *Signal Descriptions* for descriptions of the MBIST controller interface signals. See the *Cortex-A9 Processor Technical Reference Manual* for more information about the MBIST interface.

1.3 Product revisions

This section summarizes the differences in functionality between the releases of this MBIST controller.

r0p0 - r1p0 There are no functionality changes. You must use the correct corresponding revision of MBIST controller with corresponding processor revision. For example, use an r1p0 processor with an r1p0 MBIST controller.

r1p0 - r2p0 You must use the correct corresponding revision of MBIST controller with corresponding processor revision. Use an r2p0 processor with an r2p0 MBIST controller.

r2p0 - r2p2 Documentation updates only.

r2p2 - r3p1 Documentation updates only.

Chapter 2
Functional Description

This chapter contains a functional overview of the MBIST controller example design, and a description of its functional operation. It includes timing sequences for loading instructions, starting the MBIST engine, detecting failures, and retrieving the data log. It contains the following sections:

2.1 Functional overview

This section describes the interface between the MBIST controller and the RAMs that the MBIST controller tests:

- *MBIST controller interface*
- *MBISTINDATA and MBISTOUTDATA mapping* on page 2-5
- *MBIST controller implementation* on page 2-10.

2.1.1 MBIST controller interface

The MBIST controller has one MBIST port. See Appendix A *Signal Descriptions*. Only one set of RAMs is accessed by the MBIST controller at any time.

For a Cortex-A9 MPCore implementation the following pins that must be tied LOW in MBIST mode:

- **DBGEN**
- **NIDEN**
- **SPIDEN**
- **SPNIDEN**
- **EDBGRQ**
- **DBGRESTART**
- **PSELDBG**
- **CPUCLKOFF**
- **PERIPHCLKEN**
- **ACLKENS**
- **INCLKENM0**
- **INCLKENM1**
- **OUTCLKENM0**.

MAXCLKLATENCY must be driven to some value. In other words, it cannot be undriven.

For a Cortex-A9 uniprocessor implementation the following pins that must be tied LOW in MBIST mode:

- **DBGEN**
- **NIDEN**
- **SPIDEN**
- **SPNIDEN**
- **EDBGRQ**
- **DBGRESTART**
- **PSELDBG**
- **ACLKENM0**
- **ACLKENM1**
- **AWREADYM0**
- **WREADYM0**
- **BVALIDM0**
- **RVALIDM0**.

MAXCLKLATENCY must be driven to some value. In other words, it cannot be undriven.

Table 2-1 shows the MBIST interface signals

Table 2-1 MBIST interface signals

Name	Type	Description
nRESET	Input	Global active LOW reset signal.
CLK	Input	Active HIGH clock signal. This clock drives the Cortex-A9 processor logic.
MBISTOUTDATA[255:0]	Output	Data out bus from all cache RAM blocks.
MBISTENABLE	Input	Select signal for cache RAM array. This signal is the select input to the multiplexors that access the cache RAM arrays for test. When asserted, **MBISTENABLE** takes priority over all other select inputs to the multiplexors.
MBISTARRAY[19:0]	Input	One-hot chip enables to select the RAM arrays for test.
MBISTBE[25:0]	Input	Global write enable signal for all RAM arrays.
MBISTWRITEEN	Input	Global write enable.
MBISTADDR[10:0]	Input	Address signal for cache RAM array.
MBISTINDATA[63:0]	Input	Data bus to the RAM arrays. Not all RAM arrays use the full data width.
MBISTRUN	Input	Launches memory testing.
MBISTDSHIFT	Input	Shift enable for the selected Dispatch Unit Data Log Register.
MBISTSHIFT	Input	Provides serial load of the *MBIST Instruction Register* (MBIR).

You can use the MBIST controller for testing the Cortex-A9 processor compiled RAMs. You can also choose to design your own MBIST controller.

For the MBIST to run correctly on the Cortex-A9 processor if the dormant/power off wrappers are implemented, you must set the signals on the Cortex-A9 processor interface as Table 2-2 shows.

Test RAMs for symmetric CPU configurations in parallel, For example, you can test all tag RAMs in symmetric designs in parallel. Send the same data to each CPU using **BISTINDATA** and the result is read out in parallel on **BISTOUTDATA**.

For non-uniform configurations you must test the RAMs for each CPU separately.

Table 2-2 Cortex-A9 signal settings for MBIST

Signal name	Setting
CPURAMCLAMP[3:0]	4’b0000
CPUCLAMP[3:0]	4’b0000
RVALIDM1	1'b0
SCURAMCLAMP	1’b0

Table 2-3 shows the interfaces between the MBIST controller and the RAMs that MBIST tests in a configuration without parity.

Table 2-3 RAM arrays without parity and MBIST controller interfaces

RAM Name	MBISTARRAY bit	MBISTINDATA bits	MBISTBE bits	MBISTOUTDATA bits	Max address bits
SCU tag RAM way 3	[19]	[22:0]	[22:0]	[54:32]	[8:0]
SCU tag RAM way 2	[19]	[22:0]	[22:0]	[22:0]	[8:0]
SCU tag RAM way 1	[18]	[22:0]	[22:0]	[54:32]	[8:0]
SCU tag RAM way 0	[18]	[22:0]	[22:0]	[22:0]	[8:0]
Douter RAM	[17]	[11:0]	[11:0]	[11:0]	[8:0]
Data data RAM way 3 (arrays 3,7)	[16]	[63:0]	[7:0]	[63:0]	[10:0]
Data data RAM way 2 (arrays 2,6)	[15]	[63:0]	[7:0]	[63:0]	[10:0]
Data data RAM way 1 (arrays 1,5)	[14]	[63:0]	[7:0]	[63:0]	[10:0]
Data data RAM way 0 (arrays 0,4)	[13]	[63:0]	[7:0]	[63:0]	[10:0]
Data tag RAM array 3	[12]	[25:0]	[25:0]	[57:32]	[8:0]
Data tag RAM array 2	[12]	[25:0]	[25:0]	[25:0]	[8:0]
Data tag RAM array 1	[11]	[25:0]	[25:0]	[57:32]	[8:0]
Data tag RAM array 0	[11]	[25:0]	[25:0]	[25:0]	[8:0]
TLB RAM array 1	[10]	[60:0]	-	[60:0]	[5:0]
TLB RAM array 0	[9]	[60:0]	-	[60:0]	[5:0]
Global History Buffer arrays 0,1,2,3	[8]	[15:0]	[15:0]	[15:0]	[8:0]
Instruction data RAM array 7 (way 3 high)	[7]	[63:32]	-	[63:32]	[10:0]
Instruction data RAM array 6 (way 3 low)	[7]	[31:0]	-	[31:0]	[10:0]
Instruction data RAM array 5 (way 2 high)	[6]	[63:32]	-	[63:32]	[10:0]
Instruction data RAM array 4 (way 2 low)	[6]	[31:0]	-	[31:0]	[10:0]
Instruction data RAM array 3 (way 1 high)	[5]	[63:32]	-	[63:32]	[10:0]
Instruction data RAM array 2 (way 1 low)	[5]	[31:0]	-	[31:0]	[10:0]
Instruction data RAM array 1 (way 0 high)	[4]	[63:32]	-	[63:32]	[10:0]

Table 2-3 RAM arrays without parity and MBIST controller interfaces (continued)

RAM Name	MBISTARRAY bit	MBISTINDATA bits	MBISTBE bits	MBISTOUTDATA bits	Max address bits
Instruction data RAM array 0 (way 0 low)	[4]	[31:0]	-	[31:0]	[10:0]
Instruction tag RAM array 3	[3]	[21:0]	-	[53:32]	[8:0]
Instruction tag RAM array 2	[3]	[21:0]	-	[21:0]	[8:0]
Instruction tag RAM array 1	[2]	[21:0]	-	[53:32]	[8:0]
Instruction tag RAM array 0	[2]	[21:0]	-	[21:0]	[8:0]
BTAC RAM target array 1	[1]	[63:32]	-	[63:32]	[7:0]
BTAC RAM control array 1	[1]	[27:0]	-	[27:0]	[7:0]
BTAC RAM target array 0	[0]	[63:32]	-	[63:32]	[7:0]
BTAC RAM control array 0	[0]	[27:0]	-	[27:0]	[7:0]

2.1.2 MBISTINDATA and MBISTOUTDATA mapping

This section describes how the different RAM arrays are mapped on **MBISTINDATA** and **MBISTOUTDATA**:

- *Instruction data and Data data RAMs*
- *Instruction tag, Data tag, and SCU tag RAMs* on page 2-6
- *Outer RAM* on page 2-8
- *Branch Target Address Cache RAM* on page 2-9
- *TLB RAM* on page 2-9
- *Global History Buffer RAMs* on page 2-10.

Instruction data and Data data RAMs

The Instruction data RAMs are selected using bits **MBISTARRAY[7:4]**. The Data data RAMs are selected using bits **MBISTARRAY[16:13]**. This cache is byte-writable. All write enable bits must be controllable separately.

Both caches consist of eight RAM arrays for storing the associated data. See *RAM arrays without parity and MBIST controller interfaces* on page 2-4 for more information. For both data caches two arrays are tested in parallel and the same data is sent to each CPU. For `n=0` to `n=3` Data in and Data out buses are mapped as Figure 2-1 and Figure 2-2 on page 2-6 show.

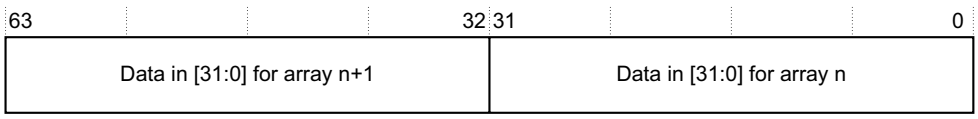


Figure 2-1 MBISTINDATA[63:0] format for Instruction data RAM and Data data RAM

The width of **MBISTINDATA** in a configuration with parity is **MBISTINDATA[71:0]**.

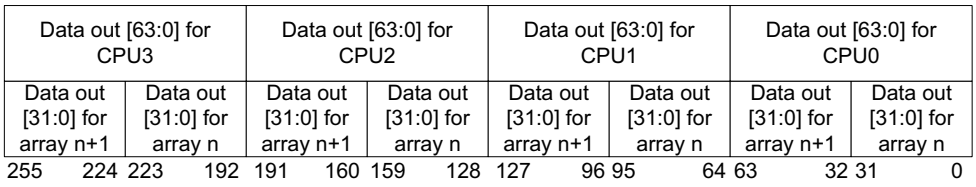


Figure 2-2 MBISTOUTDATA[255:0] format for Instruction data RAM and Data data RAM

In a configuration with parity the width of **MBISTOUTDATA** is **MBISTOUTDATA[287:0]**

Instruction data RAMs have a word write enable, controlled by **MBISTWRITEEN** when in BIST mode. Data data RAMs have a byte write enable, controlled by **MBISTBE[3:0]** as Table 2-4 shows.

Table 2-4 Data data RAM byte write enable control

MBISTBE bit	Description
0	Byte 0, bits [7:0]
1	Byte 1, bits [15:8]
2	Byte 2, bits [23:16]
3	Byte 3, bits [31:24]

Instruction tag, Data tag, and SCU tag RAMs

Instruction tag RAMs, Data tag RAMs and SCU tag RAMs all consist of four arrays per CPU. Data tag RAMs and SCU tag RAMs are identical in structure. Two arrays are tested in parallel for each CPU.

Table 2-5 shows the **MBISTARRAY** bits used to select each tag RAM.

Table 2-5 MBISTARRAY bit usage for tag RAMs

MBISTARRAY bits	Description
[3:2]	Select the Instruction tag array
[12:11]	Select the Data tag array
[19:18]	Select the SCU tag array

Figure 2-3 shows the data mapping on **MBISTINDATA** for Instruction tag RAM.

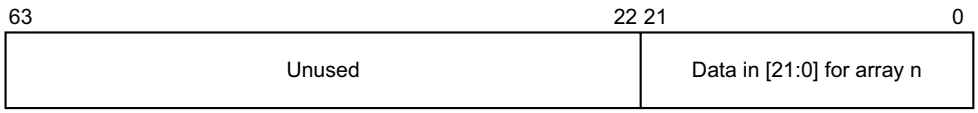


Figure 2-3 MBISTINDATA[63:0] format for Instruction tag RAM

Figure 2-4 on page 2-7 shows the data mapping on **MBISTOUTDATA** for Instruction tag RAM.

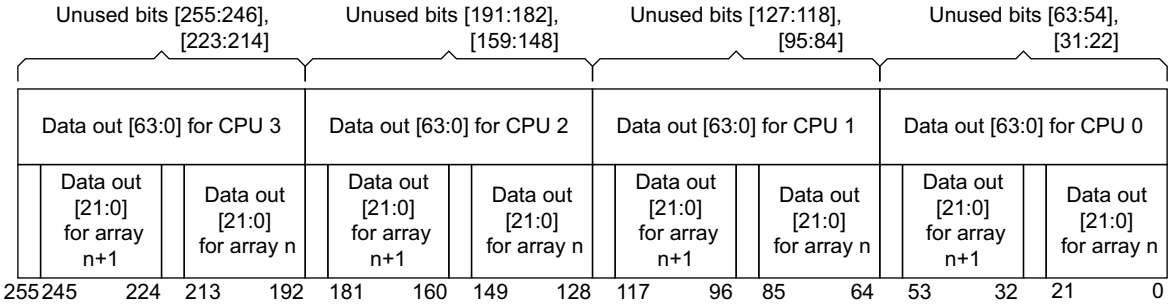


Figure 2-4 MBISTOUTDATA[255:0] format for Instruction tag RAM

Figure 2-5 and Figure 2-6 show the data mapping on **MBISTINDATA** and **MBISTOUTDATA** buses for Data tag RAM.

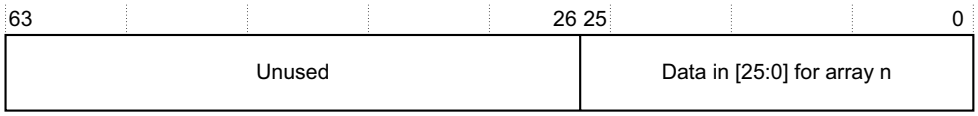


Figure 2-5 MBISTINDATA[63:0] format for Data tag RAM

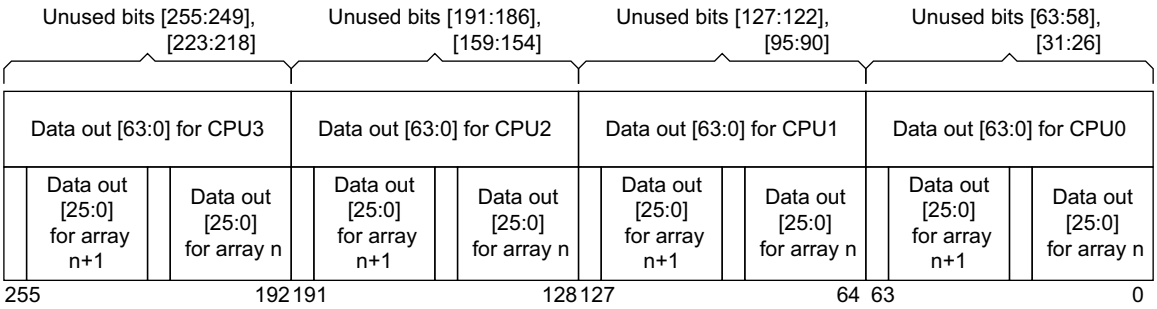


Figure 2-6 MBISTOUTDATA[255:0] format for Data tag RAM

Figure 2-7 shows the data mapping on **MBISTINDATA[63:0]** for SCU tag RAM.

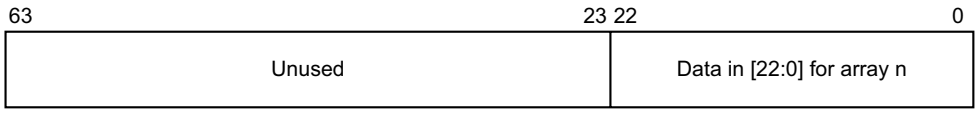


Figure 2-7 MBISTINDATA[63:0] format for SCU tag RAM

Figure 2-8 shows the data mapping on **MBISTOUTDATA[255:0]** for SCU tag RAM.

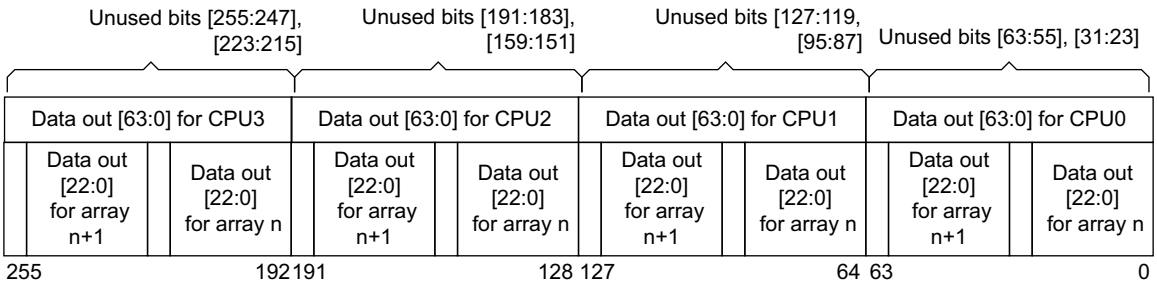


Figure 2-8 MBISTOUTDATA[255:0] format for SCU tag RAM

Figure 2-9 shows the data mapping on **MBISTINDATA[63:0]** for GHB tag RAM.

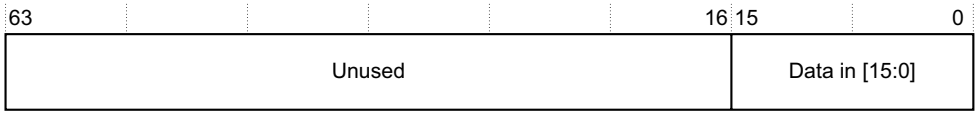


Figure 2-9 MBISTINDATA[63:0] format for GHB tag RAM

Figure 2-10 shows the data mapping on **MBISTOUTDATA[255:0]** for GHB tag RAM.

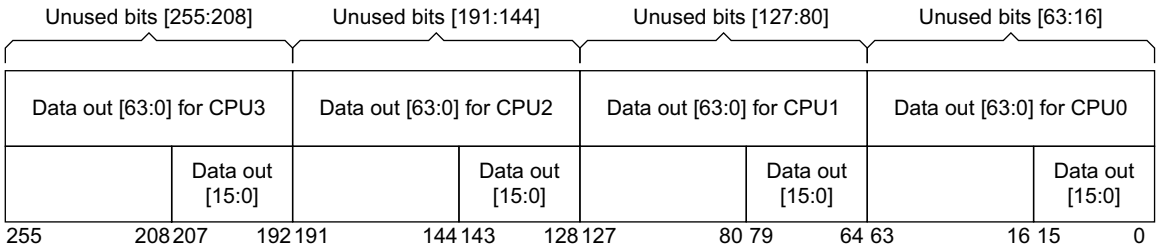


Figure 2-10 MBISTOUTDATA[255:0] format for GHB tag RAM

Table 2-6 shows the **MBISTBE** bits used to control the tag RAMs.

Table 2-6 Tag RAM control

RAM type	Write enable	MBISTBE bits
SCU tag RAM	Bit-write enable	[22:0]
Data tag RAM	Bit-write enable	[25:0]

Outer RAM

Outer RAM consists of one array per CPU. It is a bit-writable RAM. Bit-write enables must be controllable separately.

MBISTARRAY[17] selects the Outer RAM array.

Figure 2-11 shows the data mapping on **MBISTINDATA** for Outer RAM.

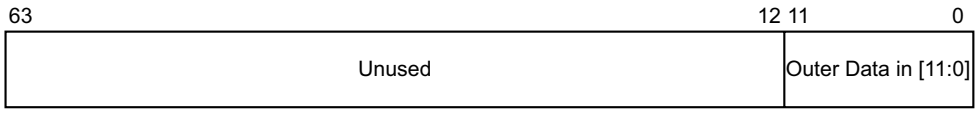


Figure 2-11 MBISTINDATA[63:0] format for Outer RAM

Figure 2-12 shows the data mapping on **MBISTOUTDATA[255:0]** for Outer RAM.

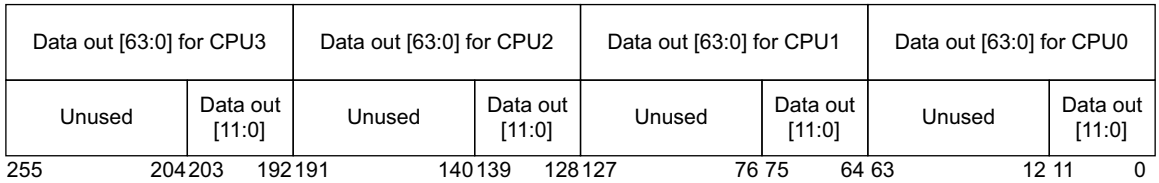


Figure 2-12 MBISTOUTDATA[255:0] format for Outer RAM

Branch Target Address Cache RAM

Branch Target Address Cache (BTAC) RAMs consist of two arrays, one for control and one for target. The target array is always 32 bits wide.

MBISTARRAY[1:0] selects the BTAC arrays. They are word-writable, controlled by **MBISTWRITEEN** when in BIST mode.

Figure 2-13 shows the data mapping for BTAC RAM.

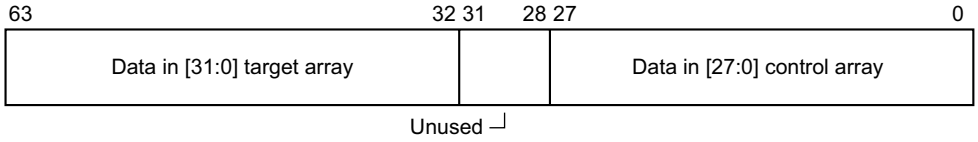


Figure 2-13 MBISTINDATA[63:0] format for BTAC RAM

Figure 2-14 shows the data mapping on **MBISTOUTDATA[255:0]** for BTAC RAM.

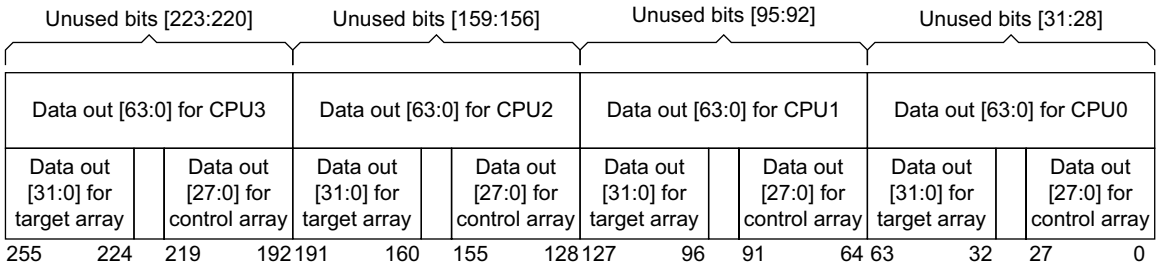


Figure 2-14 MBISTOUTDATA[255:0] format for BTAC RAM

TLB RAM

TLB RAM consists of two arrays. **MBISTARRAY[10:9]** selects these arrays. The TLB arrays are word-writable, controlled by **MBISTWRITEEN** when in BIST mode.

Figure 2-15 shows the data mapping on **MBISTINDATA[63:0]** for TLB RAM.

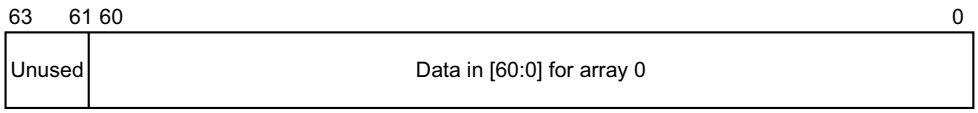


Figure 2-15 MBISTINDATA[63:0] format for TLB RAM

Figure 2-16 shows the **MBISTOUTDATA** data mapping for TLB RAM.

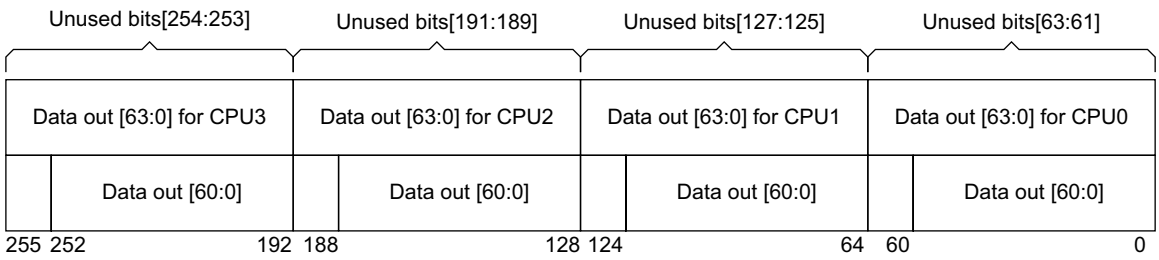


Figure 2-16 MBISTOUTDATA[255:0] format for TLB RAM

Global History Buffer RAMs

Global History Buffer (GHB) RAM consists of four arrays that are four bits wide. Address space is 512 words. **MBISTARRAY[8]** selects the GHB arrays.

The GHB arrays are bit-writable, controlled by **MBISTBE[11:0]** when in BIST mode.

Figure 2-17 shows the data mapping on the **MBISTINDATA** bus for GHB tag RAM.

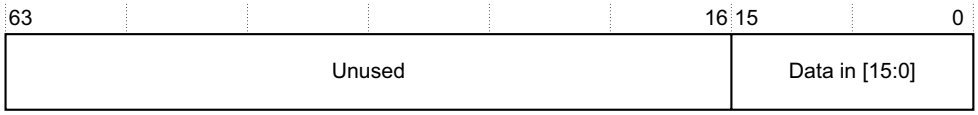


Figure 2-17 MBISTINDATA[63:0] format for GHB tag RAM

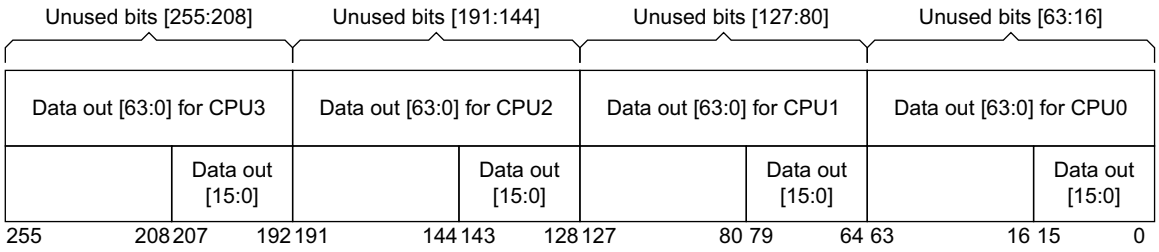


Figure 2-18 MBISTOUTDATA[255:0] format for GHB tag RAM

2.1.3 MBIST controller implementation

The MBIST controller block shown in Figure 2-19 contains two major blocks:

- MBIST controller
- dispatch unit.

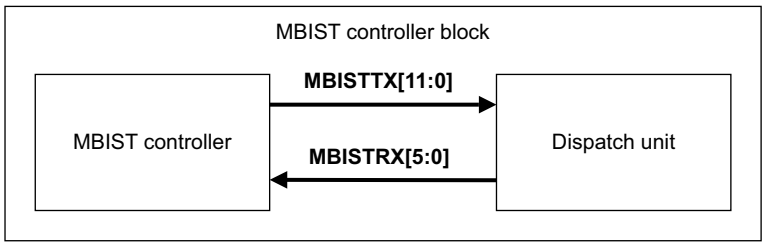


Figure 2-19 MBIST controller block

This section describes:

- *MBIST controller and dispatch unit interface* on page 2-11
- *MBIST controller block top level I/O* on page 2-12.

MBIST controller and dispatch unit interface

The MBIST controller and the dispatch unit communicate using the following signals:

MBISTTX[11:0]

This signal is an output of the MBIST controller that goes to the dispatch unit. Table 2-7 shows the signals.

Table 2-7 MBISTTX signals

MBISTTX bit	Description
0	Reset address
1	Increment address
2	Access sacrificial row, used during bang patterns
3	Invert data/instruction data in
4	Checkerboard data
5	Write data
6	Read data
7	Yfast/nXfast
8	Direction
9	Enable bitmap mode
10	Increment go/no go dataword selection
11	Latency stall control

When the instruction shift is enabled, data shifts between the two parts of the BIST engine are on bit 3. In run test mode, this bit is used as invert data information. The **MBISTTX[11:0]** interface is ARM-specific and intended for use only with the MBIST controller.

MBISTRX[5:0]

This signal is an output of the dispatch unit that goes to the MBIST controller. The behavior of **MBISTRX[5:0]** is ARM-specific and is intended for use only with the MBIST controller. The address expire signal is set when both the row and column address counters expire. Table 2-8 shows the signals.

Table 2-8 MBISTRX signals

MBISTRX bit	Description
0	Real-time error flag
1	Shadow pipeline empty
2	CPU0 address/instruction data out/fail data out
3	CPU1 address/instruction data out/fail data out
4	CPU2 address/instruction data out/fail data out
5	CPU3 address/instruction data out/fail data out

MBIST controller block top level I/O

The top level I/O of the MBIST controller contains the Cortex-A9 processor interface. See Appendix A *Signal Descriptions* and the inputs and outputs shown in Table 2-9.

Table 2-9 MBIST controller top level I/O

Signal	Direction	Function	Value, MBIST mode	Value, function mode
MBISTDATAIN	Input	Serial data in	Toggle	0
MBISTDSHIFT	Input	Data log shift	Toggle	0
nRESET	Input	MBIST reset	Toggle	0[a]
MBISTRESULT[5:0]	Output	Output status bus	Strobe	-
MBISTRUN	Input	Run MBIST test	Toggle	0
MBISTSHIFT	Input	Instruction shift	Toggle	0
MBISTENABLE	Input	MBIST path enable	Toggle	0
SE	Input	ATPG signal	0	0

a. **nRESET** and **MBISTENABLE** must be LOW in functional mode.

The following signals have additional information:

SE Preservation of array state is required when performing multiload *Automatic Test Pattern Generator* (ATPG) runs or when performing **IDDQ** testing. After performing MBIST tests to initialize the arrays to a required background, the ATPG test procedures must assert **SE** during all test setup cycles in addition to load/unload. Any clocking during **IDDQ** capture cycles must have array chip select signals constrained.

MBISTRESULT[5:0]

During tests, the **MBISTRESULT[1]** signal indicates failures. You can operate using two modes, by configuring bit [5] of the engine control section of the instruction register. If bit [5] is set, the **MBISTRESULT[1]** signal is asserted for a single cycle for each failed compare. If bit [5] is not set, the **MBISTRESULT[1]** signal is sticky, and is asserted from the first failure until the end of the test.

At the completion of the test, the **MBISTRESULT[0]** signal goes HIGH. The **MBISTRESULT[5:2]** signal indicates that an address expire for the processor under test has occurred and enables you to measure sequential progress through the test algorithms.

2.2 Functional operation

The functional operation is described in:

- *Timing*
- *Bitmap mode* on page 2-15.

2.2.1 Timing

A 58-bit instruction, loaded serially at the start of each test, controls the operation of the MBIST controller. Chapter 3 *MBIST Instruction Register* describes how to write the instruction.

The timing diagrams in this section show the clock running at two different speeds:

- the slower clock relates to the clock driven by your ATE
- the faster clock relates to the clock driven by an on-chip *Phase Locked Loop* (PLL).

If you do not have an on-chip PLL, both clocks relate to the clock driven by your ATE.

See the *Cortex-A9 Technical Reference Manual* and *Cortex-A9 MPCore Technical Reference Manual* for timing information about the processors.

Timing diagrams in the following sections show the procedures for operating the MBIST controller:

- *Instruction load*
- *Starting MBIST*
- *Failure detection* on page 2-14
- *Data log retrieval* on page 2-14.

Note

The MBIST controller must be reset between each array tested.

Instruction load

To load an MBIST instruction, drive **MBISTSHIFT** HIGH. At the next rising clock edge, the 58-bit shift sequence begins as Figure 2-20 shows. To enable data input from the ATE, the PLL is in bypass mode, and the clock is not running at test frequency.

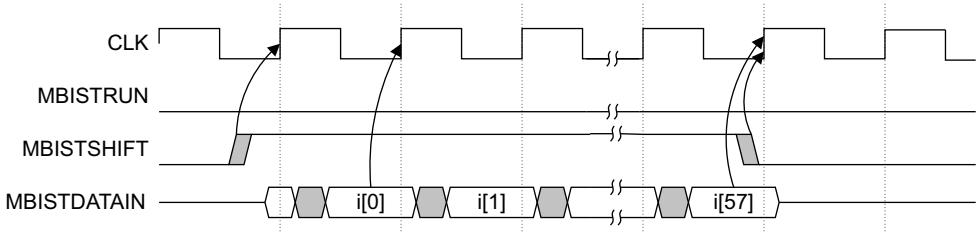


Figure 2-20 Loading the MBIST controller instruction

Starting MBIST

After loading the MBIST instruction, drive **MBISTSHIFT** LOW and disable **CLK**. With **CLK** disabled, drive **MBISTRUN** HIGH and, after an **MBISTRUN** setup time, start the PLL at the test frequency as Figure 2-21 on page 2-14 shows.

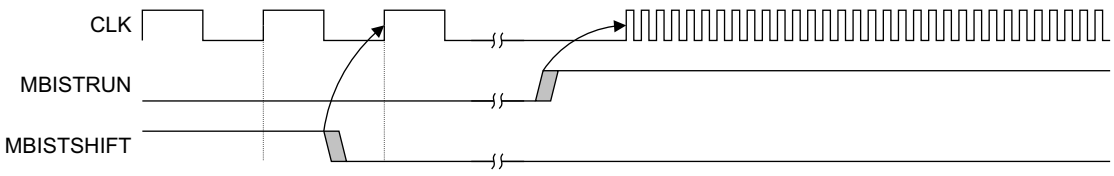


Figure 2-21 Starting the MBIST test

Failure detection

The **MBISTRESULT[1]** flag goes HIGH two **CLK** cycles after the controller detects a failure, as Figure 2-22 shows. It stays HIGH if sticky fail is enabled. If stop-on-fail is enabled, the **MBISTRESULT[0]** flag goes HIGH two cycles later.

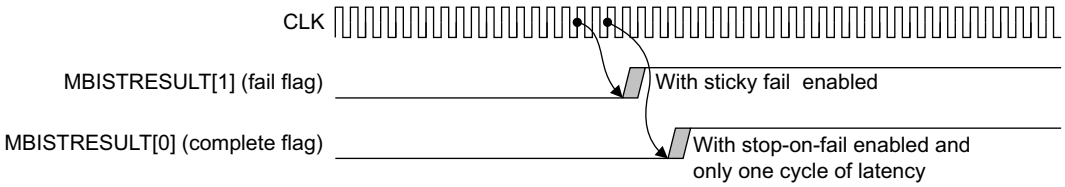


Figure 2-22 Detecting an MBIST failure

Note

To ensure that the ATE can observe a failure at test speed, specify a sticky fail in the MBIST instruction. See *Control field, MBIR[51:46]* on page 3-5.

Data log retrieval

During a test, the MBIST controller automatically logs the first detected failure. If required, you can retrieve the data log at the end of the test to generate failure statistics. Figure 2-23 on page 2-15 and Figure 2-24 on page 2-15 show the method of retrieving a data log.

Note

MBISTRESULT[2] is the serial data output for instructions and the data log for CPU0.

MBISTRESULT[3] is the serial data output for instructions and the data log for CPU1.

MBISTRESULT[4] is the serial data output for instructions and the data log for CPU2.

MBISTRESULT[5] is the serial data output for instructions and the data log for CPU3.

After the **MBISTRESULT[0**] flag goes HIGH, stop the test by putting the PLL in bypass mode and driving **MBISTRUN** LOW as Figure 2-23 on page 2-15 shows. To begin shifting out the data log on **MBISTRESULT[5:2]**, drive **MBISTDSHIFT** HIGH. The **MBISTRESULT[1]** error flag goes LOW two cycles after **MBISTRUN** goes LOW. Data begins shifting out on **MBISTRESULT[5:2]** two cycles after **MBISTDSHIFT** goes HIGH.

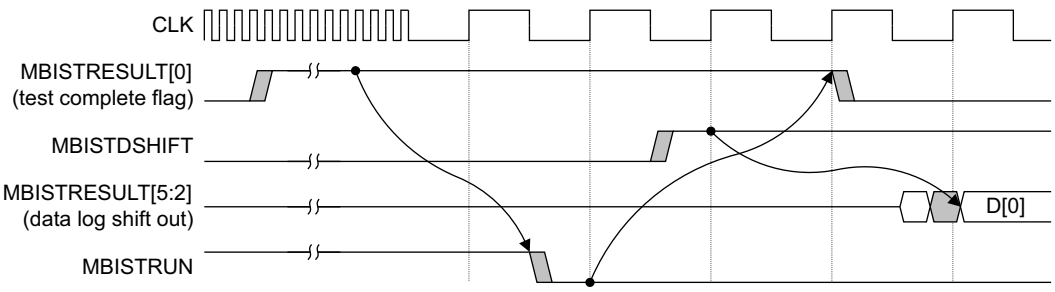


Figure 2-23 Start of data log retrieval

In Bitmap mode, **MBISTRUN** can stay high. Asserting **MBISTDSHIFT** clears **MBISTRESULT[1]**, fail flag, and stalls the controller during datalog retrieval.

When the last data log bit shifts out, drive **MBISTDSHIFT** LOW as Figure 2-24 shows.

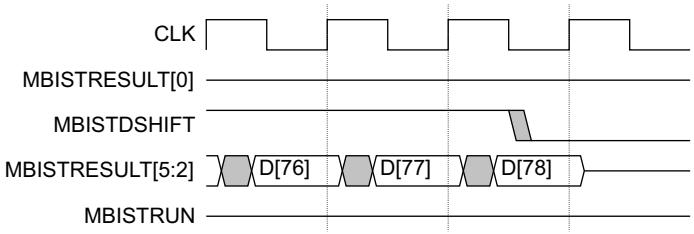


Figure 2-24 End of data log retrieval

Table 2-10 shows the format of the data log.

Table 2-10 Data log format

Bits	Description
[78:68]	Address of the failing location.
[67:4]	Failing data bits. These bits are set for faulty bits and cleared for passing bits.
[3:0]	The data seed used in the test. See *DataWord field, MBIR[27:24]* on page 3-8.

The address contained in the data log refers to the full address of the failing location as it appears on the **MBISTADDR[10:0]** port of the MBIST interface of the Cortex-A9 processor.

See also Chapter 4 *MBIST Datalog Register*.

2.2.2 Bitmap mode

In bitmap mode, you can identify all failing locations in a RAM. Each time a failure occurs, the controller stops executing the test and waits for you to begin shifting out the data log as Figure 2-25 on page 2-16 shows.

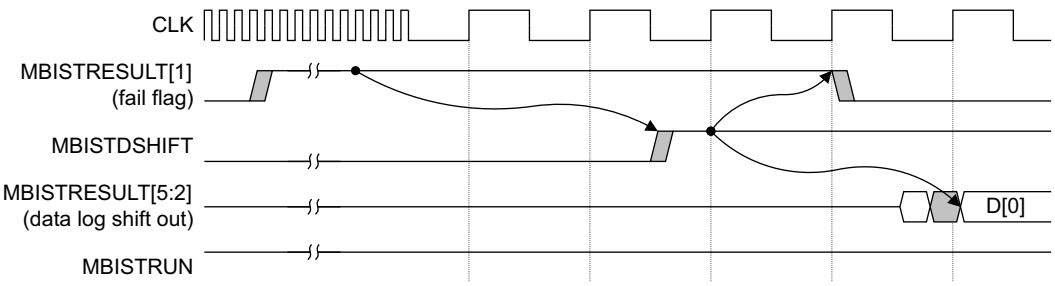


Figure 2-25 Start of bitmap data log retrieval

After you finish shifting and drive **MBISTDSHIFT** LOW, the controller then resumes testing where it stopped as Figure 2-26 shows. This process continues until the test algorithm completes. A fault can cause a failure to occur several times during a given test algorithm. The fault might be logged multiple times depending on the number of reads performed by the algorithm and the exact nature of the fault.

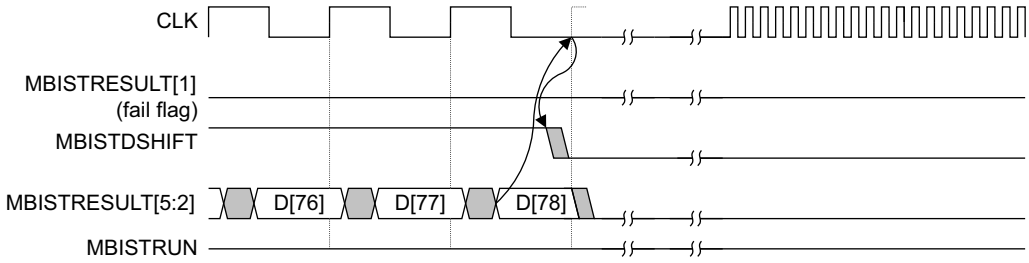


Figure 2-26 End of bitmap data log retrieval

Loading a new instruction resets bitmap mode.

Chapter 3
MBIST Instruction Register

This chapter describes how to use the *MBIST Instruction Register* (MBIR) to configure the mode of operation of the MBIST controller. It contains the following sections:

3.1 About the MBIST instruction register

The MBIST executes loaded instructions stored in the *MBIST Instruction Register* (MBIR). The MBIR is 58 bits wide and divided into a control unit part of 18 bits and a dispatch unit part of 40 bits.

The MBIR is loaded through serial port **MBISTDATAIN** of the control unit when **MBISTSHIFT** is asserted. When **MBISTSHIFT** is asserted again, the control unit passes **MBISTDATAIN** serially to the dispatch unit through its **MBISTTX[3]** port.

Figure 3-1 shows the control unit part of the MBIR.

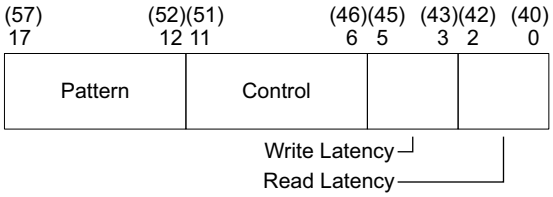


Figure 3-1 MBIST instruction register control unit

The control unit contains the following fields:

Pattern Specifies the test algorithm.

Control Specifies MBIST mode of operation and sticky or nonsticky fail flag.

Write latency Specifies the number of cycles to enable a RAM write.

Read latency Specifies the number of cycles to enable a RAM read.

Figure 3-2 shows the dispatch unit part of the MBIR.

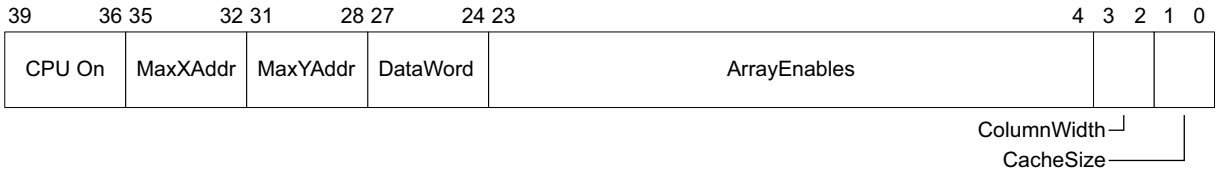


Figure 3-2 MBIST instruction register dispatch unit

The dispatch unit contains the following fields:

CPU On Controls the data comparison for the CPUs under test.

MaxXAddr Specifies the number of bits in the X-address counter.

MaxYAddr Specifies the number of bits in the Y-address counter.

DataWord Data seed to be used during test. These 4 bits are replicated 16 times to form 64 bits of data.

ArrayEnables Specifies the RAM under test.

ColumnWidth Specifies 4, 8, 16, or 32 columns per block of RAM.

CacheSize Specifies a cache size of 16KB, 32KB, or 64KB.

Field descriptions on page 3-3 describes the MBIR fields in more detail.

3.2 Field descriptions

The following sections describe the MBIR fields:

- *Pattern field, MBIR[57:52]*
- *Control field, MBIR[51:46]* on page 3-5
- *Read Latency and Write Latency fields, MBIR[42:40] and MBIR[45:43]* on page 3-5
- *CPU On field, MBIR[39:36]* on page 3-6
- *MaxXAddr and MaxYAddr fields, MBIR[35:32] and MBIR[31:28]* on page 3-7
- *DataWord field, MBIR[27:24]* on page 3-8
- *ArrayEnables field, MBIR[23:4]* on page 3-8
- *ColumnWidth field, MBIR[3:2]* on page 3-9
- *CacheSize field, MBIR[1:0]* on page 3-10.

3.2.1 Pattern field, MBIR[57:52]

The MBIST controller is supplied with industry-standard pattern algorithms and a bit-line stress algorithm. You can group algorithms together to create a specific memory test methodology for your product.

Table 3-1 describes the supported algorithms, and *Pattern specification* describes their use. The N values in the table indicate the number of RAM accesses per address location and give an indication of the test time when using that algorithm.

Table 3-1 Pattern field encoding

Pattern MBIR[57:52]	Algorithm name	N	Description
b000000	Write Solids	1N	Write a solid pattern to memory
b000001	Read Solids	1N	Read a solid pattern from memory
b000010	Write Checkerboard	1N	Write a checkerboard pattern to memory
b000011	Read Checkerboard	1N	Read a checkerboard pattern from memory
b000100	March C+ (x-fast)	14N	March C+ algorithm, incrementing X-address first
b001011	March C+ (y-fast)	14N	March C+ algorithm, incrementing Y-address first
b000101	Fail Pattern	6N	Tests memory failure detection capability
b000110	Read Write March (x-fast)	6N	Read write march pattern, incrementing X-address first
b000111	Read Write March (y-fast)	6N	Read write march pattern incrementing Y-address first
b001000	Read Write Read March (x-fast)	8N	Read write read march pattern, incrementing X-address first
b001001	Read Write Read March (y-fast)	8N	Read write read march pattern, incrementing y-address first
b001010	Bang	18N	Bit-line stress pattern
b111111	Go/No-Go	30N	See Table 3-2 on page 3-4

Pattern specification

This section describes the MBIST test patterns. An x-fast pattern increments or decrements the X-address counter first. A y-fast pattern increments or decrements the Y-address counter first. *MaxXAddr and MaxYAddr fields, MBIR[35:32] and MBIR[31:28]* on page 3-7 describes the X-address and Y-address counters.

The first four patterns are useful for data retention or I_{DDQ} testing.

Write Solids This initializes the RAM with the supplied data seed.

Read Solids This reads each RAM location once, expecting the supplied data seed.

Write Checkerboard

This initializes the RAM with a physical checkerboard pattern created by alternating the supplied data seed and its inverse.

Read Checkerboard

This reads back the physical checkerboard pattern created by alternating the supplied data seed and its inverse.

For the next set of patterns, the following notation describes the algorithm:

0 represents the data seed.

1 represents the inverse data seed.

r represents a read operation.

w represents a write operation.

incr Increment address starting with 0 until address = addrmax.

decr Decrement address starting with addrmax until address = 0.

March C+ (x-fast or y-fast)

This is the industry-standard March C+ algorithm:

```
(w0) (r0, w1, r1) (r1, w0, r0) decr (r0, w1, r1) decr (r1, w0, r0) (r0)
```

Read Write March (x-fast or y-fast)

```
(w0) (r0, w1) decr (r1, w0) (r0)
```

Read Write Read March (x-fast or y-fast)

```
(w0) (r0, w1, r1) decr (r1, w0, r0) (r0)
```

Bang This test is always performed in x-fast. It executes multiple consecutive writes and reads effectively stressing a bit-line pair. While this pattern does detect stuck-at faults, its primary intent is to address the analog characteristics of the memory. In the following algorithm description, row 0 indicates a read or write of the data seed to the sacrificial row, this is always the first row of the column being addressed.

```
(w0) (r0, w1, w1(row 0) x 6) (r1 x 5, w0(row 0), r1, w0) (r0)
```

Go/No-Go If you do not want to implement your own memory test strategy, use the Go/No-Go test pattern that performs the algorithms that Table 3-2 shows.

Table 3-2 Go/No-Go test pattern

Sequence	Algorithm	Data
1	Write Checkerboard	Data seed
2	Read Checkerboard	Data seed
3	Write Checkerboard	Data seed

Table 3-2 Go/No-Go test pattern (continued)

Sequence	Algorithm	Data
4	Read Checkerboard	Data seed
5	Read Write Read March (y-fast)	0x6
6	Bang	0xF

This test suite provides a comprehensive test of the arrays. The series of tests in Go/No-Go are the result of the experience in memory testing by ARM memory test engineers.

3.2.2 Control field, MBIR[51:46]

This 6-bit control field specifies the MBIST function. Table 3-3 shows how the five LSB bits of the Control field affect the behavior of the MBIST controller.

Table 3-3 Control field encoding (five LSB bits)

Control MBIR[50:46]	Behavior	Description
bx00000	Default	Test runs to completion. If **MBIR[51]** is 0, sticky fail present after first failure.
bx00001	Stop-on-fail	End of test on failure.
bx00011	Bitmap mode	Enables logging of each failure. See *Bitmap mode* on page 2-15.

MBIR[51] selects a nonsticky or sticky fail flag, **MBISTRESULT[1]**:

- When **MBIR[51]** is set, the fail bit toggles in real time. It goes HIGH for failing comparisons and LOW for passing comparisons.

  Note

  Setting **MBIR[51]** can cause the fail bit to toggle at the test frequency. ARM recommends that you do not set **MBIR[51],** when the external pin or the ATE cannot follow the test frequency.

- When **MBIR[51]** is cleared, the fail bit is sticky. It remains HIGH after the first failure until a new MBIST instruction shifts in or until the data log shifts out.

3.2.3 Read Latency and Write Latency fields, MBIR[42:40] and MBIR[45:43]

The Read Latency and Write Latency fields of the MBIR are used to specify the read and write latency of the RAM under test. Read and write latencies are the numbers of cycles that the RAM requires to complete read and write operations. For example, in a write to a RAM with a write latency of two cycles, the RAM inputs are valid for a single cycle. The next cycle is a NOP cycle with the chip enable negated. Similarly, in a read from a RAM with a read latency of three cycles, the RAM inputs are valid for a single cycle. After two NOP cycles, the read data is valid on the RAM outputs.

Note

Even if the RAM under test uses the same latency for both read and write operations, you must still program both the read latency and write latency fields of the MBIR with the same value.

Table 3-4 shows the latency settings for read operations.

Table 3-4 Read latency field encoding

Read Latency MBIR[42:40]	Number of cycles per read operation
b000	1
b001	2
b010	3
b011	4
b100	5
b101	6
b110	7
b111	8

Table 3-5 shows the latency settings for write operations.

Table 3-5 Write latency field encoding

Write Latency MBIR[45:43]	Number of cycles per write operation
b000	1
b001	2
b010	3
b011	4
b100	5
b101	6
b110	7
b111	8

3.2.4 CPU On field, MBIR[39:36]

The CPU On field controls data comparison for the CPUs under test.

Setting one of the **MBIR[39:36]** bits to 0 disables the data comparison for the relevant CPU. Table 3-6 shows the CPU mapping of this field.

Table 3-6 MBIR[39:36] CPU mapping

MBIR bit	CPU
39	CPU3
38	CPU2
37	CPU1
36	CPU0

3.2.5 MaxXAddr and MaxYAddr fields, MBIR[35:32] and MBIR[31:28]

You can determine the number of address bits you must specify for a RAM from the MBIR fields:

- *MaxXAddr*
- *MaxYAddr* on page 3-8.

This enables you to specify your address range in two dimensions. The two-dimensional specification represents the topology of the physical implementation of the RAM more accurately. The dimensions are controlled by two separate address counters, the X-address counter and the Y-address counter. One counter can be incremented or decremented only when the other counter has expired. The chosen test algorithm determines the counter that moves faster.

Use this procedure to determine how many bits to assign to the X-address and Y-address counters:

1. Determine the column width of the RAM array. The Y-address must have at least that many bits for the column select.
2. Determine how many address bits the RAM requires. See the *Cortex-A9 Processor Configuration and Sign-off Guide* for more information.

MaxXAddr

The MaxXAddr field specifies the number of X-address counter bits to use during test. Table 3-7 shows the MaxXAddr settings.

Table 3-7 MaxXAddr field encoding

MaxXAddr MBIR[35:32]	Number of counter bits
<b0010	Unsupported
b0010	2
b0011	3
b0100	4
b0101	5
b0110	6
b0111	7
b1000	8
b1001	9
b1010	10
>b1010	Reserved

MaxYAddr

The MaxYAddr field specifies the number of Y-address counter bits to use during test. Table 3-8 shows the MaxYAddr settings.

Table 3-8 MaxYAddr field encoding

MaxYAddr MBIR[31:28]	Number of counter bits
<b0010	Unsupported
b0010	2
b0011	3
b0100	4
b0101	5
b0110	6
b0111	7
b1000	8
b1001	9
b1010	10
>b1010	Reserved

3.2.6 DataWord field, MBIR[27:24]

DataWord is the 4-bit data seed field that supplies the background data for the test algorithm at instruction load.

Note

In the Go/No-Go algorithm, the Read Write Read March (y-fast) and Bang algorithms do not use the data seed value. Table 3-2 on page 3-4 shows the data that the Go/No-Go algorithm uses.

The data seed enables you to select values stored into arrays for $\mathbf{I_{DDQ}}$ ATPG, or to select data words to search for unexpected sensitivities during march or bit-line stress tests. The MBIST engine replicates the four bits of data 16 times to give the full 64 bits of data required on the **MBISTDIN[63:0]** port of the MBIST interface.

3.2.7 ArrayEnables field, MBIR[23:4]

Table 3-9 shows how each bit in the ArrayEnables field selects the cache RAM array to be tested. You can select only one array at a time. Selecting multiple arrays produces unpredictable behavior.

Table 3-9 ArrayEnables field encoding

ArrayEnables MBIR[23:4]	RAM name
b00000000000000000001	BTAC RAM control array 0 and target array 0
b00000000000000000010	BTAC RAM control array 1 and target array 1
b00000000000000000100	Instruction tag RAM arrays 0 and 1

Table 3-9 ArrayEnables field encoding (continued)

ArrayEnables MBIR[23:4]	RAM name
b00000000000000001000	Instruction tag RAM arrays 2and 3
b00000000000000010000	Instruction data RAM way 0 (block 0 and 1)
b00000000000000100000	Instruction data RAM way 1 (block 2 and 3)
b00000000000001000000	Instruction data RAM way 2 (block 4 and 5)
b00000000000010000000	Instruction data RAM way 3 (block 6 and 7)
b00000000000100000000	Global History Buffer
b00000000001000000000	TLB RAM array 0
b00000000010000000000	TLB RAM array 1
b00000000100000000000	Data tag RAM arrays 0 and 1
b00000001000000000000	Data tag RAM arrays 2and 3
b00000010000000000000	Data data RAM way 0 (block 0 and 4)
b00000100000000000000	Data data RAM way 1 (block 1and 5)
b00001000000000000000	Data data RAM way 2 (block 2 and 6)
b00010000000000000000	Data data RAM way 3 (block 3 and 7)
b00100000000000000000	Douter RAM
b01000000000000000000	SCU tag RAM arrays 0 and 1
b10000000000000000000	SCU tag RAM arrays 2 and 3

3.2.8 ColumnWidth field, MBIR[3:2]

The ColumnWidth field specifies the number of columns in each block of RAM in the array under test. The column address is always encoded in the least significant bits of the RAM address, so the number of columns determines the number of bits used. This information is important for the correct operation of certain MBIST operations, for example bit-line stress testing and writing a true physical checkerboard pattern to the array.

Table 3-10 shows the supported column widths along with the number of LSB address bits used for each and the MBIR encodings required to select them.

Table 3-10 ColumnWidth field encoding

ColumnWidth MBIR[3:2]	Number of columns	Number of address bits
b00	4	2
b01	8	3
b10	16	4
b11	32	5

3.2.9 CacheSize field, MBIR[1:0]

The CacheSize field specifies the size of the cache in your implementation of the module. Table 3-11 shows the supported cache sizes.

Table 3-11 CacheSize field encoding

CacheSize MBIR[1:0]	Cache size
b00	16KB
b10	32KB
b11	64KB
b01	Reserved[a]

a. Mapped internally to 16KB

Chapter 4
MBIST Datalog Register

This chapter describes the MBIST Datalog Register. It contains the following sections:

4.1 About the MBIST Datalog Register

The MBIST Datalog Register records information about failing arrays. Figure 4-1 shows the register format for a configuration with parity.

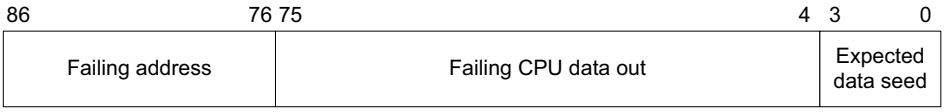


Figure 4-1 MBIST Datalog Register format for a configuration with parity

Figure 4-2 shows the register format for a nonparity configuration.

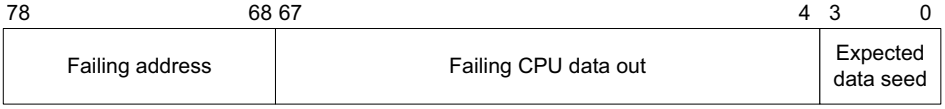


Figure 4-2 MBIST Datalog Register format for a nonparity configuration

Field descriptions on page 4-3 describes the register fields in detail.

The datalogs for all CPUs are dumped in parallel through **MBISTRESULT[5:2]** with:

- **MBISTRESULT[5]** for CPU3
- **MBISTRESULT[4]** for CPU2
- **MBISTRESULT[3]** for CPU1
- **MBISTRESULT[2]** for CPU0.

4.2 Field descriptions

These are the fields in the MBIST Datalog Register in a configuration with parity:

Datalog[86:76] 11 bits that contain the failing address.

Datalog[75:4] 72 bits that contain an XOR between failing data and correct data. All bits at1'b1 are failing.

Datalog[3:0] 4 bits that contain the expected data seed.

These are the fields in the MBIST Datalog Register in a configuration without parity:

Datalog[78:68] 11 bits that contain the failing address.

Datalog[67:4] 64 bits that contain an XOR between failing data and correct data. All bits at1'b1 are failing.

Datalog[3:0] 4 bits that contain the expected data seed.

Appendix A
Signal Descriptions

This appendix describes the MBIST controller signals. It contains the following sections:

A.1 MBIST controller interface signals

Table A-1 shows the MBIST controller interface signals in a nonparity configuration.

Table A-1 MBIST controller interface signals in a nonparity configuration

Signal	Type	Description
MBISTOUTDATA[255:0]	Input	MBIST data out, from Cortex-A9 processor.
MBISTADDR[10:0]	Output	MBIST address
MBISTARRAY[19:0]	Output	MBIST RAM one-hot chip enables. See Table A-3.
MBISTINDATA[63:0]	Output	MBIST data in, to Cortex-A9 processor
MBISTBE[25:0]	Output	MBIST write enable.
MBISTWRITEEN	Output	Global write enable.

Table A-2 shows the MBIST controller interface signals in a configuration with parity.

Table A-2 MBIST controller interface signals in a configuration with parity

Signal	Type	Description
MBISTOUTDATA[287:0]	Input	MBIST data out, from Cortex-A9 processor.
MBISTADDR[10:0]	Output	MBIST address.
MBISTARRAY[19:0]	Output	MBIST RAM one-hot chip enables. See Table A-3.
MBISTINDATA[71:0]	Output	MBIST data in, to Cortex-A9 processor.
MBISTBE[32:0]	Output	MBIST write enable.
MBISTWRITEEN	Output	Global write enable.

Table A-3 shows the MBISTARRAY one-hot chip enables.

Table A-3 MBISTARRAY one-hot chip enables

MBISTARRAY bit	RAM name
0	BTAC RAM control array 0 and target array 0
1	BTAC RAM control array 1 and target array 1
2	Instruction tag RAM arrays 0 and 1
3	Instruction tag RAM arrays 2 and 3
4	Instruction data RAM way 0 (blocks 0 and 1)
5	Instruction data RAM way 1 (blocks 2 and 3)
6	Instruction data RAM way 2 (blocks 4 and 5)
7	Instruction data RAM way 3 (blocks 6 and 7)
8	Global History Buffer
9	TLB RAM array 0
10	TLB RAM array 1

Table A-3 MBISTARRAY one-hot chip enables (continued)

MBISTARRAY bit	RAM name
11	Data tag RAM arrays 0 and 1
12	Data tag RAM arrays 2 and 3
13	Data data RAM way 0 (blocks 0 and 4)
14	Data data RAM way 1 (blocks 1 and 5)
15	Data data RAM way 2 (blocks 2 and 6)
16	Data data RAM way 3 (blocks 3 and 7)
17	Douter RAM
18	SCU tag RAM arrays 0 and 1
19	SCU tag RAM arrays 2 and 3

A.2 Miscellaneous signals

Table A-4 shows the miscellaneous signals.

Table A-4 Miscellaneous signals

Signal	Type	Description
nRESET	Input	Global active LOW reset signal
CLK	Input	Clock
MBISTDATAIN	Input	Serial data in
MBISTDSHIFT	Input	Data log shift
MBISTRESETN	Input	MBIST reset
MBISTRESULT[5:0]	Output	Output status bus
MBISTRUN	Input	Run MBIST test
MBISTSHIFT	Input	Instruction shift
MBISTENABLE	Input	MBIST mode enable

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

Table B-1 Differences between issue A and issue B

Change	Location
No technical changes	-

Table B-2 Differences between issue B and issue C

Change	Location
Updated description about the MBIST controller	*About the MBIST controller* on page 1-2
Updated MBIST controller signals	Figure 1-2 on page 1-3
Updated MBIST controller interface signals	Figure 1-4 on page 1-4
Updated signal names and settings	Table 2-2 on page 2-3
Updated bit information for the MBIST controller interfaces	Table 2-3 on page 2-4
Clarified data in for Instruction tag RAM	Figure 2-3 on page 2-6
Clarified data out for Instruction tag RAM	Figure 2-4 on page 2-7
Clarified Tag RAM control	Table 2-6 on page 2-8
Updated TLB RAM description	*TLB RAM* on page 2-9
Updated Branch Target Address Cache RAM description	*Branch Target Address Cache RAM* on page 2-9

Table B-3 Differences between issue C and issue D

Change	Location
MBIST Controller diagram with parity added	Figure 1-3 on page 1-3
MBIST signals with parity configuration added	Table 1-2 on page 1-6
Information about MBISTINDATA and MBISTOUTDATA sizes in configurations with parity added.	*MBISTINDATA and MBISTOUTDATA mapping* on page 2-5
Arrows removed from pattern descriptions	*Pattern specification* on page 3-3
Datalog failing data out size now 72 bits	Figure 4-1 on page 4-2 *Field descriptions* on page 4-3
MBIST datalog Register formats for configurations with parity and without parity added.	*About the MBIST Datalog Register* on page 4-2
Field descriptions for configurations with parity and without parity added.	*Field descriptions* on page 4-3
MBIST Controller interface signals with parity added	Table A-2 on page A-2

Table B-4 Differences between issue D and issue E

Change	Location
No technical changes	-

Table B-5 Differences between issue D and issue F

Change	Location
Cortex-A9 Multiprocessor TRM added to further reading	*ARM publications* on page vii
References to reset sequences in TRMs included	*About the MBIST controller* on page 1-2
Lists of pins to tie LOW added	*MBIST controller interface* on page 2-2
Bit field descriptions and titles harmonized	*MBISTINDATA and MBISTOUTDATA mapping* on page 2-5
Corrections to graphics and text	*Instruction data and Data data RAMs* on page 2-5
	Instruction tag, Data tag, and SCU tag RAMs on page 2-6
	Outer RAM on page 2-8
	Branch Target Address Cache RAM on page 2-9
	TLB RAM on page 2-9
	Global History Buffer RAMs on page 2-10
Cross-reference about processor timing added	*Timing* on page 2-13
Corrections to graphics	Figure 2-20 on page 2-13
	Figure 2-26 on page 2-16
Bit field and title harmonized	*CPU On field, MBIR[39:36]* on page 3-6
MBIR values corrected	Table 3-6 on page 3-6

Table B-6 Differences between issue F and issue G

Change	Location	Affects
Added note about resetting MBIST between arrays tested.	*Timing* on page 2-13	All revisions
Updated Instruction load figure	Figure 2-20 on page 2-13	All revisions
Updated timing diagram for data log retrieval	Figure 2-23 on page 2-15	All revisions
Updated Data log retrieval description	*Data log retrieval* on page 2-14	All revisions
Updated timing diagrams for bitmap mode	Figure 2-25 on page 2-16	All revisions
	Figure 2-26 on page 2-16	
Updated field descriptions for MBIST Datalog register	Figure 4-1 on page 4-2	All revisions
	Field descriptions on page 4-3	
Updated **MBISTOUTDATA** signal description in a parity configuration	Table A-2 on page A-2	All revisions
Update algorithm for Bang pattern description	page 3-4	All revisions