ARM11 Memory Built-In Self Test Controller

Technical Reference Manual

ARM DDI 0289B

ARM11 Memory Built-In Self Test Controller
Technical Reference Manual

Release Information

The table below shows the release state and change history of this document.

Change history

Date	Issue	Change
14 April 2003	A	First release
28 April 2003	B	Addition of ARM11 to product name

Proprietary Notice

Confidentiality Status

This document is Open Access. This document has no restriction on distribution.

Product Status

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

Web Address

`http://www.arm.com`

Contents

ARM11 Memory Built-In Self Test Controller Technical Reference Manual

List of Tables
ARM11 Memory Built-In Self Test Controller Technical Reference Manual

List of Figures
ARM11 Memory Built-In Self Test Controller Technical Reference Manual

Preface

This preface introduces the *ARM11 Memory Built-In Self Test Controller Technical Reference Manual*. It contains the following sections:

- *About this book* on page x
- *Feedback* on page xiii.

About this book

This book provides a description of the ARM11 *Memory Built-In Self Test* (MBIST) Controller.

Intended audience

This book is written for hardware engineers who are familiar with ARM technology and want to use the ARM11 MBIST Controller to test embedded memory on ARM11-based devices.

Using this book

This book is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an overview of the ARM11 MBIST Controller.

Chapter 2 ***Functional Description***

Read this chapter for timing and data log retrieval information for the ARM11 MBIST Controller.

Chapter 3 ***ARM11 MBIST Controller Instruction Register***

Read this chapter for a description of the ARM11 MBIST Controller Instruction Register and associated bit assignments.

Appendix A ***Signal Descriptions***

Read this chapter for descriptions of the ARM11 MBIST Controller signals.

Appendix B ***Integration with the ARM1136 Processor***

Read this chapter for a description of using the ARM11 MBIST Controller with an ARM1136 processor.

Appendix C ***Integration with the ETB11***

Read this chapter for a description using the ARM11 MBIST Controller with the *ARM11 Embedded Trace Buffer* (ETB11).

Typographical conventions

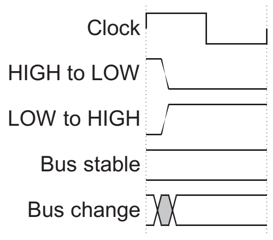
The following typographical conventions are used in this book:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
`monospace`	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`monospace italic`	Denotes arguments to commands and functions where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.

Timing diagram conventions

This manual contains timing diagrams. The figure below explains the components used in these diagrams. Any variations are clearly labeled when they occur. Therefore, no additional meaning must be attached unless specifically stated.



Key to timing diagram conventions

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

Other conventions

This document uses other conventions. They are described in the following sections:

- *Signals* on page xii
- *Bits and bytes* on page xii
- *Numbers* on page xii.

Signals

When a signal is described as being asserted, the level depends on whether the signal is active HIGH or active LOW. Asserted means HIGH for active high signals and LOW for active low signals:

Prefix n	Active LOW signals are prefixed by a lowercase n except in the case of AHB or APB reset signals. These are named **HRESETn** and **PRESETn** respectively.

Bits and bytes

Suffix b	Indicates bits.
Suffix B	Indicates bytes.
Byte	Eight bits.

Numbers

Suffix K	Indicates an amount of memory. It means 1024.
Suffix M	When used to indicate an amount of memory means $1024^2 = 1048576$. When used to indicate a frequency means 1000000.
Prefix 0x	Indicates hexadecimal.
Prefix b	Indicates binary.

Further reading

This section lists publications from both ARM Limited and third parties that provide additional information on developing code for the ARM family of processors.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the ARM Frequently Asked Questions list.

Feedback

ARM Limited welcomes feedback on both the ARM11 MBIST Controller, and its documentation.

Feedback on the ARM11 MBIST Controller

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

- the product name
- a concise explanation of your comments.

Feedback on this book

If you have any comments on this book, send email to `errata@arm.com` giving:

- the document title
- the document number
- the page number(s) to which your comments apply
- a concise explanation of your comments.

General suggestions for additions and improvements are also welcome.

Chapter 1
Introduction

This chapter describes the ARM11 MBIST Controller. It contains the following sections:

1.1 Overview

In recent years, MBIST has become the industry-standard method of testing embedded memories. MBIST works by performing sequences of reads and writes to the memory according to an algorithm. Many industry-standard algorithms exist.

An MBIST Controller is used to generate the correct sequence of reads and writes. The ARM11 MBIST Controller can be used with some ARM products to perform embedded memory testing. ARM11 MBIST Controllers are currently available for the following products:

- ARM1136JF-S processor
- ARM1136J-S processor
- ARM1136J-S *PrimeXsys Platform* (PXP) uses the ARM11 MBIST Controller
- ETB11.

This *Technical Reference Manual* (TRM) provides a high-level description of the common features of these ARM11 MBIST Controllers. Information that is specific to individual ARM11 MBIST Controllers for each of the supported products is provided in Appendix B *Integration with the ARM1136 Processor* and Appendix C *Integration with the ETB11*.

1.2 MBIST ports

The traditional method of gaining access to a RAM for MBIST is shown in Figure 1-1.

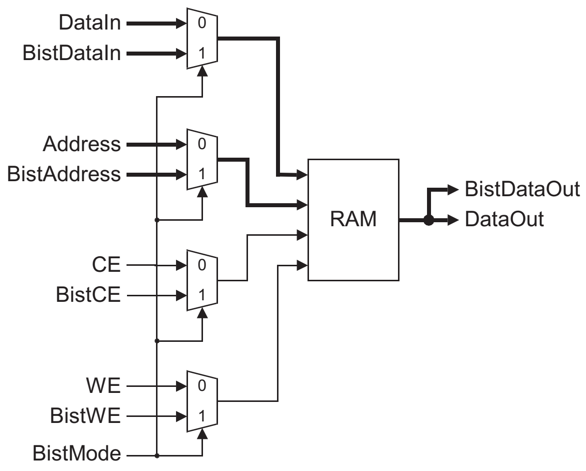


Figure 1-1 Traditional method of interfacing MBIST

This is not suitable for use in high-performance designs because the maximum operating frequency is reduced significantly. Instead, an additional input to existing multiplexors is added without reducing maximum operating frequency. Figure 1-2 on page 1-4 shows four of the pipeline stages used to access the RAM blocks. This number can vary depending on the product used.

——— **Note** ————

The method shown in Figure 1-1 has the advantage of having the two cycle register-to-register path that accesses the RAM blocks using the same path in memory BIST mode as in functional mode.

————————

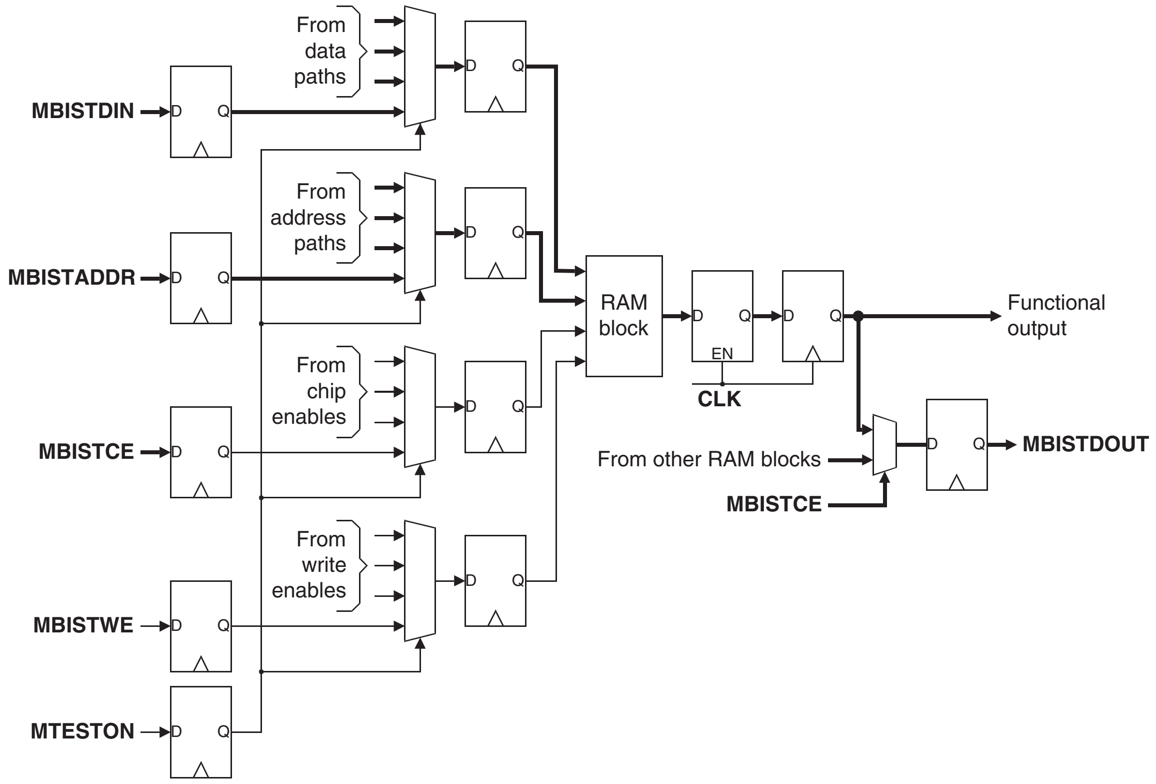


Figure 1-2 ARM11 processor MBIST interface

The ARM11 MBIST Controller accesses the memory in this way through the MBIST interface. This contains the ports listed in Table 1-1.

Table 1-1 ARM11 MBIST Controller interface signals

Name	Type	Description
MTESTON	Input	Switches multiplexors to give access to the RAM blocks. Must be HIGH during MBIST mode.
MBISTDIN	Input	Data bus to the RAM blocks. Not all RAM blocks are the full width.
MBISTADDR	Input	RAM block address signals. Not all RAM blocks use the full address width.

Table 1-1 ARM11 MBIST Controller interface signals (continued)

Name	Type	Description
MBISTCE	Input	Selects RAM blocks. One hot-chip enable for each of the RAM blocks. This signal can be a bus if required.
MBISTWE	Input	Global write enabling signal to all of the RAM blocks.
MBISTDOUT	Output	Data out bus for all of the RAM blocks. Multiple RAM blocks can be selected if output bits do not overlap.

Chapter 2
Functional Description

This chapter provides timing and data log retrieval information for the ARM11 MBIST Controller. It contains the following sections:

- *Timing* on page 2-2
- *Bitmap mode* on page 2-6.

2.1 Timing

Timing diagrams showing the procedure for operating the ARM11 MBIST Controllers are provided in the following sections:

- *Introduction*
- *Instruction load*
- *Starting MBIST* on page 2-3
- *Fail detection* on page 2-3
- *Data log retrieval* on page 2-4.

2.1.1 Introduction

The ARM11 MBIST Controller uses a 40-bit instruction to control its operation. This is described in Chapter 3 *ARM11 MBIST Controller Instruction Register*. It is loaded serially at the start of each test.

ARM Limited recommends that you run your MBIST tests at the full frequency of the design. The timing diagrams in this section show the clock running at two different speeds:

- the slower clock relates to the clock driven by your *Automated Test Equipment* (ATE)
- the faster clock relates to the clock driven by an on-chip PLL.

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

2.1.2 Instruction load

Figure 2-1 shows the method used to load an instruction into an ARM11 MBIST Controller.

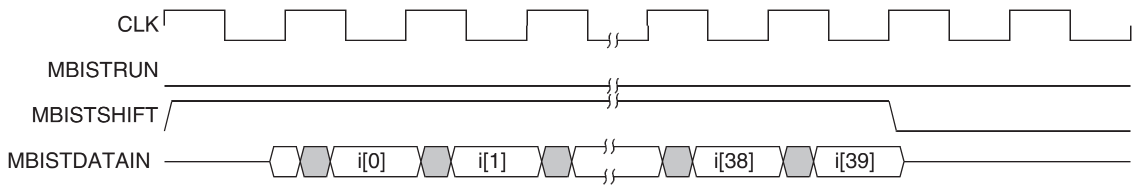


Figure 2-1 Loading the ARM memory BIST instruction

The instruction is loaded with the PLL bypassed to enable data to be input from the ATE.

2.1.3 Starting MBIST

Figure 2-2 shows how the MBIST test is started.

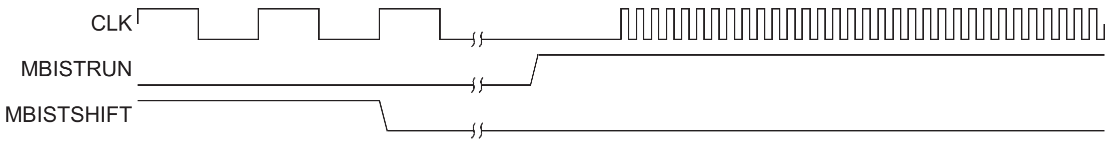
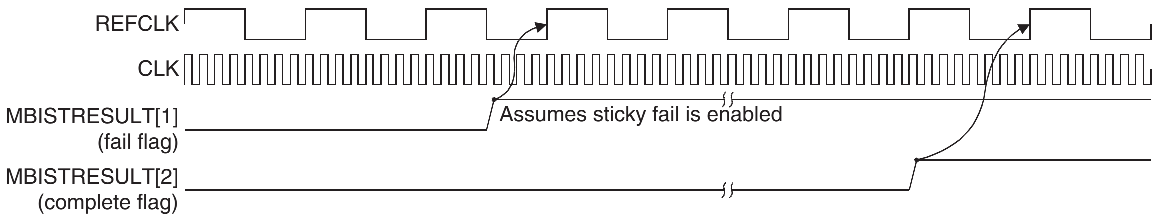


Figure 2-2 Starting the memory BIST test

You must normally run this using the PLL as the source of the clock to ensure the memories are tested at-speed.

2.1.4 Fail detection

Figure 2-3 shows how fails can be detected by the ATE using the ARM11 MBIST Controller.



REFCLK
CLK
MBISTRESULT[1] (fail flag)
MBISTRESULT[2] (complete flag)
Assumes sticky fail is enabled

Figure 2-3 Detecting a failure during memory BIST

Note

When running the ARM11 MBIST Controller at speed, you are advised to use a Sticky Fail Flag in the control field of the instruction to ensure the failure can be observed by the ATE.

2.1.5 Data log retrieval

During a test, the first failure to be detected is automatically logged by the ARM11 MBIST Controller. If required, you can retrieve this at the end of the test to generate failure statistics. The method of retrieving a data log is shown in Figure 2-4 and Figure 2-5.

Note

MBISTRESULT[0] acts as shift out for instructions and data log.

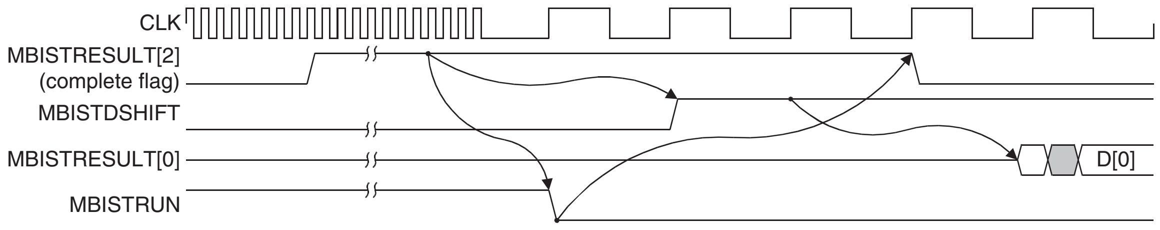


Figure 2-4 Start of data log retrieval

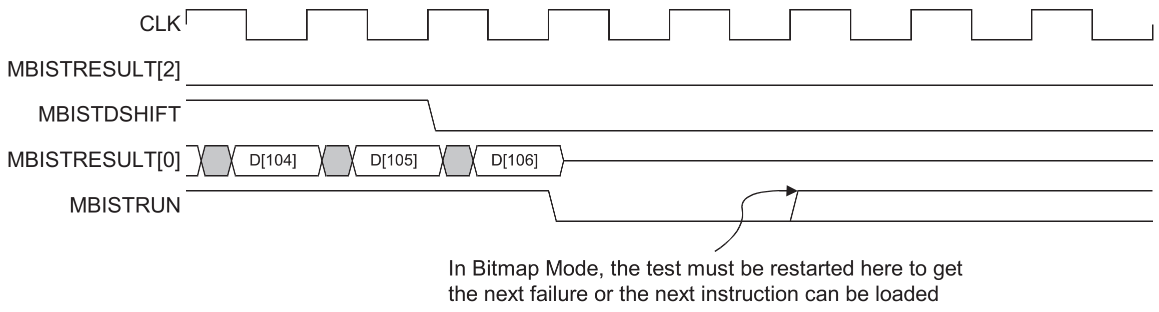

CLK
MBISTRESULT[2]
MBISTDSHIFT
MBISTRESULT[0]
MBISTRUN
D[104]
D[105]
D[106]
In Bitmap Mode, the test must be restarted here to get the next failure or the next instruction can be loaded

Figure 2-5 End of data log retrieval

Table 2-1 shows the data log.

Table 2-1 Format of the data log

Bits	Description
[106:91]	The enables specified in the instruction. See *Enables* on page 3-3.
[90:87]	The data seed specified in the instruction. See *Data seed* on page 3-3.
[86:64]	Address of the failing location.
[63:0]	Data, bits are set for faulty bits.

The address contained in the data log refers to the failing location in the address space defined by the enable field of your current instruction. Because multiple RAMs are sometimes tested by a single enable, you must use the most significant bits of the address in the data log to determine exactly which RAM contained the fault. Contact ARM Limited if you require more information.

2.2 Bitmap mode

Bitmap mode can be used to identify all failing locations in a RAM. You must run the test multiple times. You then count the number of fails detected. The first failure is logged the first time the test is run. The first failure is ignored the next time that it is run and the second failure is logged. This can continue until the test passes, and all failures have been found. A fault can cause a failure for several times and therefore might be logged multiple times depending on the number of reads performed by the selected algorithm and the exact nature of the fault.

Bitmap mode is reset by loading a new instruction.

Note

Bitmap mode is designed for failure analysis only. You must not use it during production testing. When Bitmap mode is not used, the first detected failure is recorded in the data log so that it can be retrieved at the end of the test.

Chapter 3
ARM11 MBIST Controller Instruction Register

This chapter describes the ARM11 MBIST Controller Instruction Register and associated bit assignments. It contains the following sections:

3.1 Instruction Register

You can use the *MBIST Instruction Register* (MBIR) to test varying sized arrays in embedded system RAM. It comprises the following fields:

- Pattern
- Control
- x-Locations (the number of locations in the column (x) direction)
- y-Locations (the number of locations in the row (y) direction)
- Data seed
- Enables.

The fields are described in *Field descriptions* on page 3-3.

Figure 3-1 shows the MBIR bit assignments.

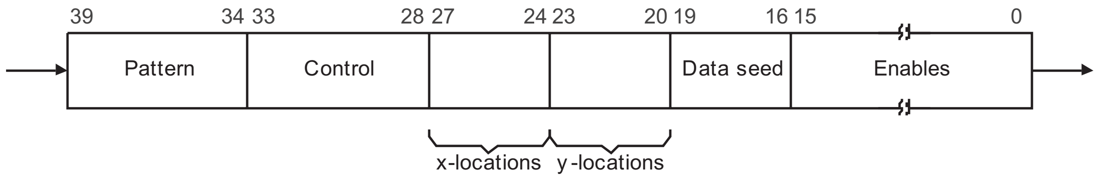


Figure 3-1 Memory BIST Instruction Register bit assignments

3.2 Field descriptions

Descriptions of the MBIR fields are provided in:

- *Enables*
- *Data seed*
- *Number of locations*
- *Control* on page 3-6
- *Pattern* on page 3-7.

3.2.1 Enables

The enables field selects which RAM is tested. See Appendix B *Integration with the ARM1136 Processor* and Appendix C *Integration with the ETB11* for specific information about the use of the field with your ARM11 MBIST Controller.

3.2.2 Data seed

You specify the data seed used during the test algorithm at instruction load.

Note

This does not apply during the Go/No-Go memory algorithm. See Table 3-5 on page 3-9 for the Go/No-Go data used.

The data seed enables you to select values stored into arrays for IDDQ ATPG, or to select data words to search for unexpected sensitivities during march or bit-line stress tests. Only four bits of data are specified and this is replicated 16 times to give the full 64-bits of data required.

3.2.3 Number of locations

The number of locations you must specify for a RAM can be determined from:

- x-Locations
- y-Locations.

This enables you to specify your address range in two dimensions. This represents the topology of the physical implementation of the RAM more accurately. For example, a RAM can have the topology shown in Figure 3-2 on page 3-4.

Note

The maximum number of locations that you can test is 8M. This is not 8M. It means that there can be 8M addresses. The maximum memory size depends on the word size.

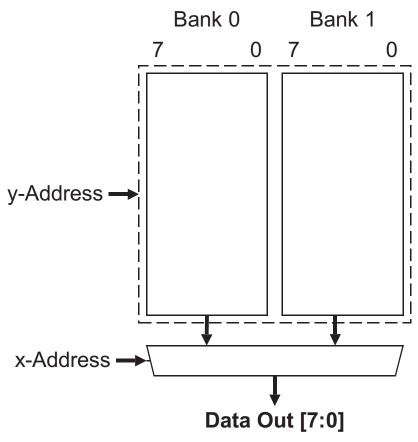


Figure 3-2 Example RAM topology

The RAM in Figure 3-2 uses a single bit for the x-Address to select between the two columns. The remaining bits of the address select a row and are called the y-Address.

The address output from each of the ARM11 MBIST Controllers is formed by concatenating the y-Address with the x-Address. This is done depending on the value of x-Locations as shown in Figure 3-3.

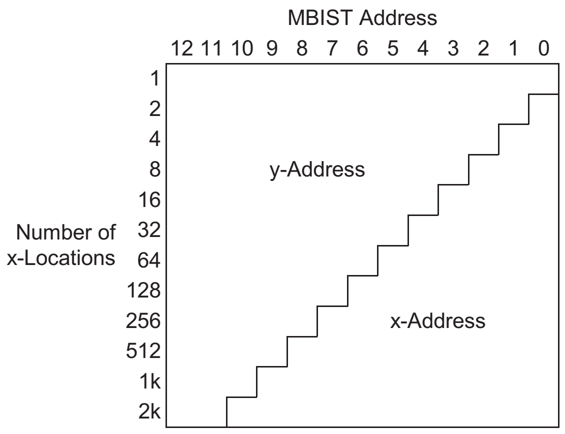


Figure 3-3 MBIST location addressing

x-Locations

The x-Locations specify the number of column locations to use during test. Table 3-1 describes the register settings and resulting address sizes.

Table 3-1 Register settings and resulting address sizes

x-Locations	Number of locations
b0000	1
b0001	2
b0010	4
b0011	8
b0100	16
b0101	32
b0110	64
b0111	128
b1000	256
b1001	512
b1010	1k
b1011	2k
>b1011	Undefined

y-Locations

The y-Locations specify the number of column locations to use during test. Table 3-2 describes the register settings and resulting address sizes.

Table 3-2 Register settings and resulting address sizes

y-Locations	Number of locations
b0000	1
b0001	2
b0010	4

Table 3-2 Register settings and resulting address sizes (continued)

y-Locations	Number of locations
b0011	8
b0100	16
b0101	32
b0110	64
b0111	128
b1000	256
b1001	512
b1010	1k
b1011	2k
b1100	4k
>b1100	Undefined

3.2.4 Control

The Control field is used to specify the MBIST function. Table 3-3 describes the behavior of the engine control field.

Bit 5 of the Control field (bit 33 of the MBIR) designates the behavior of the MBISTOUT failure bit:

- When MBISTOUT is set then the MBISTRESULT fail bit follows the runtime failure bit
- When MBISTOUT is not set then the MBISTOUT failure bit follows the sticky failure register.

Table 3-3 Behavior of the engine control field

MBIR[32:28]	Behavior	Description
b00000	Default	Test runs to completion, sticky fail present after first fail
b00001	Stop on fail	Indicates early end of test
b00011	Bitmap mode	Places controller into interactive mode for each failure, see *Bitmap mode* on page 2-6

3.2.5 Pattern

Industry standard patterns and an ARM created bit-line stress pattern are provided for use in MBIST. You can group algorithms together to create a specific memory test methodology for a manufacturer.

Table 3-4 describes the supported patterns and *Pattern specification* on page 3-8 describes their use.

The Go/No-Go pattern is recommended. This is described in *Pattern specification* on page 3-8.

Table 3-4 Supported patterns

MBIR[39:34]	Name	N	Description
b000000	Write Solids	1N	Write a solid pattern to the memory
b000001	Read Solids	1N	Read a solid pattern from the memory
b000010	Write Checkerboard	1N	Write a checkerboard pattern to the memory
b000011	Read Checkerboard	1N	Read a checkerboard pattern from the memory
b000100	March C+ (x-fast)	14N	March C+ Algorithm, incrementing the x-address first
b001011	March C+ (y-fast)	14N	March C+ Algorithm, incrementing the y-address first
b000101	Fail Pattern	6N	Tests memory failure detection
b000110	Read Write March (x-fast)	6N	Read write march pattern, incrementing the x-address first
b000111	Read Write March (y-fast)	6N	Read write march pattern, incrementing the y-address first
b001000	Read Write Read March (x-fast)	8N	Read write read march pattern, incrementing the x-address first
b001001	Read Write Read March (y-fast)	8N	Read write read march pattern, incrementing the y-address first
b001010	Bang	18N	Bitline stress pattern
b111111	Go/No-Go	30N	See Table 3-5 on page 3-9.

The definitions for the abbreviations used in Table 3-4 are:

y-fast	Target cell moves along bit-lines before moving to next column.
x-fast	Target cell moves across bit-line pairs before row or word-line.
N	Number of addressable entries.

Pattern specification

This section describes the patterns that can be selected for MBIST. Patterns are sometimes described as x-fast or y-fast. This indicates if the x-Address or the y-Address is to be incremented first. x-Address and y-Address are described in *Number of locations* on page 3-3.

Write Solids This is performed x-fast. This pattern is 1N, writing only.

Read Solids This is performed x-fast. This pattern is 1N, reading only.

Write Checkerboard

This is performed x-fast. This pattern is 1N, writing only.

Read Checkerboard

This is performed x-fast. This pattern is 1N, reading only.

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

This is performed x-fast or y-fast with the following sequence, where 0 represents the data seed, and 1 represents the inverse data seed:

```
⇑(w0) ⇑ (r0, w1, r1) ⇑ (r1, W0, r0) ⇓ (r0, w1, r1) ⇓ (r1, w0, r0) ⇑ (r0)
```

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

This is performed x-fast or y-fast with the following sequence, where 0 represents the data seed, and 1 represents the inverse data seed:

```
⇑(w0) ⇑ (r0, w1) ⇓ (r1, w0) ⇑ (r0)
```

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

This is performed x-fast or y-fast with the following sequence, where 0 represents the data seed, and 1 represents the inverse data seed:

```
⇑(w0) ⇑ (r0, w1, r1) ⇓ (r1, w0, r0) ⇑ r0)
```

Bang This test is always performed x-fast. It executes consecutive multiple writes and reads on a bit-line pair.

This pattern does detect stuck-at faults, but its primary intent is to address the analog characteristics of the memory.

Go/No-Go It is recommended that you use the Go/No-Go test pattern that performs the algorithms shown in Table 3-5 if you do not want to implement your own memory test strategy.

Table 3-5 Go/No-Go algorithm

Sequence	Algorithm	Data seed
1	Write Checkerboard	`0x5`
2	Read Checkerboard	`0x5`
3	Write Checkerboard	`0xA`
4	Read Checkerboard	`0xA`
5	Read Write Read March (y-fast)	`0xA`
6	Bang	`0x0`

This test suite provides a comprehensive test of the arrays. The series of tests in Go/No-Go are the result of past experiences of memory testing by internal ARM memory test engineers. The Data seed supplied in the instruction is ignored during this test and the data used instead is as shown in Table 3-5.

Appendix A
Signal Descriptions

This appendix describes the ARM Memory BIST signals. It contains the following sections:

A.1 Signal descriptions

You can use the ARM Memory BIST controllers with the following macrocells:

- ARM1136
- ETB11.

The Memory BIST Controllers use the Memory BIST Interface implemented on these two macrocells. For more information, see:

- *ARM1136 Implementation Guide*
- *ETB11 Implementation Guide*.

The ARM Memory BIST Interface signals are listed in Table A-1.

Table A-1 Signal descriptions

Pin	Type	Description
MBDATAIN	Input	Enables control register data load.
MBDSHIFT	Input	Provides serial load of control registers.
MBRESULT[2:0]	Output	Provides runtime status: • done = bit 2 • fail = bit 1 • expire = bit 0.
MBRUN	Input	Memory testing is launched by this signal.
MBSHIFT	Input	Shift enable for the selected Dispatch Unit Register.
MTESTON	Input	Switches multiplexors to give access to the RAM blocks. Must be HIGH during MBIST mode.

Each BIST controller is serially loaded with an instruction that describes the memory to be tested and the algorithm that must be used.

Appendix B
Integration with the ARM1136 Processor

This appendix describes how to choose the MBIST RAM size for the different types of memory used with the ARM1136 processor. It contains the following sections:

2.1 Instruction Register enables field

Table B-1 shows the bits in the enable field that select the RAMs to be tested. Only one group of RAMs can be selected at a time. Selecting multiple groups produces undefined behavior.

Table B-1 Enable bit RAM selection

Enable bit	RAM group name
[0]	TLB
[1]	BTAC
[2]	ITCM Data
[3]	Cache Valid
[4]	TCM Valid
[5]	I Cache Tag
[6]	I Cache Data
[7]	Dirty
[8]	DTCM Data
[9]	D Cache Tag
[10]	D Cache Data

B.2 Choosing the RAM size

This section describes how to determine the RAM size that must be specified for each of the RAMs selected by the enables described in Table B-1 on page B-2. This is described in:

- *TLB RAM*
- *BTAC RAM*
- *Instruction TCM data RAM* on page B-4
- *Cache valid RAMs* on page B-5
- *TCM valid RAMs* on page B-5
- *Instruction cache tag RAMs* on page B-6
- *Instruction cache data RAMs* on page B-7
- *Dirty RAMs* on page B-8
- *Data TCM data RAM* on page B-9
- *Data cache tag RAMs* on page B-9
- *Data cache data RAMs* on page B-10.

B.2.1 TLB RAM

The TLB RAMs are tested as a single 64-bit wide RAM with 64 locations. This size is fixed, regardless of the cache and TCM sizes.

You must choose values for the number of x and y-Locations so that their product is 64. Some example values are shown in Table B-2.

Table B-2 Example values for x- and y-Locations, TLB RAM

x-Locations	y-Locations
64	1
32	2
16	4
8	8

B.2.2 BTAC RAM

The BTAC RAMs are tested as a single 61-bit wide RAM with 128 locations. This size is fixed, regardless of the cache and TCM sizes.

You must choose values for the number of x and y-Locations so that their product is 128. Some example values are shown in Table B-3.

Table B-3 Choosing values for x- and y-Locations, BTAC RAM

x-Locations	y-Locations
128	1
64	2
32	4
16	8

B.2.3 Instruction TCM data RAM

The ITCM data RAM is tested as a single 64-bit wide RAM. The number of locations in this RAM depends on the size of the ITCM, as shown in Table B-4.

Table B-4 ITCM size and location

ITCM size	ITCM data RAM locations
4KB	512
8KB	1024
16KB	2048
32KB	4096
64KB	8192

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations in your ITCM data RAM.

For example, if you have an 8KB ITCM the ITCM data RAM has 1024 locations. You can therefore choose values for the number of x and y-Locations to be 512 and 2 respectively.

Note

If your implementation does not have an Instruction TCM, you must not run an instruction to test this RAM.

B.2.4 Cache valid RAMs

The cache valid RAMs are each eight bits wide, and are tested simultaneously. Logic in the memory BIST controller ensures that data read from locations that are not implemented does not cause failures.

——— **Note** ———

If your instruction and data caches are different sizes, you must choose the larger of the two sizes for determining the number of locations to be specified in the BIST instruction.

———————————

The number of locations in each of the valid RAMs is shown in Table B-5.

Table B-5 Cache valid size and locations

Cache size (I or D)	Cache valid RAM locations
4KB	16
8KB	32
16KB	64
32KB	128
64KB	256

You must choose values for the number of x and y-Locations such so that their product is equal to the number of locations in the largest of the valid RAMs in your caches. For example, consider the case of an 8KB instruction cache and a 16KB data cache.

The larger of these two values is 16KB, which has a valid RAM with 64 locations. You must select the number of x and y-Locations so that their product is equal to 64. The ARM11 MBIST Controller then ensures that only implemented locations in the Instruction cache valid RAM are checked.

B.2.5 TCM valid RAMs

The TCM valid RAMs are each eight bits wide and are tested simultaneously. Logic in the ARM11 MBIST Controller ensures that data read from locations that are not implemented does not cause failures.

Note

If your instruction and data TCMs are different sizes, you must choose the larger of the two sizes for determining the number of locations to be specified in the BIST instruction.

The number of locations in each of the valid RAMs is shown in Table B-6.

Table B-6 TCM size and locations

TCM size (I or D)	TCM valid RAM locations
No TCM	0
4KB	16
8KB	32
16KB	64
32KB	128
64KB	256

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations in the largest of the valid RAMs in your TCMs. For example, consider the case of an 8KB Instruction TCM and a 4KB Data TCM.

The larger of these two values is 8KB, which has a valid RAM with 32 locations. You must select the number of x and y-Locations so that their product is equal to 32. The BIST controller then ensures that only implemented locations in the data TCM valid RAM are checked.

Note

If your implementation has no TCMs, you must not run an instruction to test this RAM.

B.2.6 Instruction cache tag RAMs

The instruction cache tag RAMs are tested as a single 44-bit wide RAM that has twice the number of locations as a single tag RAM because of the way the RAMs are concatenated during BIST. The number of locations that must be specified in the BIST instruction depends on the size of the instruction cache.

For MBIST, the number of locations that must be specified in the BIST instruction is shown in Table B-7.

Table B-7 Instruction cache size and tag RAM locations

Instruction cache size	Instruction cache tag RAM locations
4KB	64
8KB	128
16KB	256
32KB	512
64KB	1024

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations given in Table B-7 for your specific instruction cache size.

B.2.7 Instruction cache data RAMs

The instruction cache data RAMs are tested as a single 64-bit wide RAM that has four times the number of locations as a single data RAM because of the way the RAMs are concatenated during BIST. The number of locations that must be specified in the BIST instruction depends on the size of the instruction cache.

For memory BIST, the number of locations that must be specified in the BIST instruction is shown in Table B-8.

Table B-8 Instruction cache size and data RAM locations

Instruction cache size	Instruction cache data RAM locations
4KB	512
8KB	1024
16KB	2048
32KB	4096
64KB	8192

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations given in Table B-8 on page B-7 for your specific instruction cache size.

B.2.8 Dirty RAMs

The dirty RAMs are each eight bits wide, and are tested sequentially using a single instruction. Logic in the memory BIST controller ensures that data read from locations that are not implemented does not cause failures.

You must use the greater of either the data cache size or the data TCM size to find the number of locations that must be specified in the instruction. You can find this size in Table B-9.

Table B-9 Greater of data cache or data TCM size and dirty RAM locations

Greater of data cache or data TCM size	Dirty RAM locations
4KB	64
8KB	128
16KB	256
32KB	512
64KB	1024

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations given in Table B-9. For example, consider the case of an 8KB Data Cache and a 4KB Data TCM. The greater of these sizes is 8KB, and using the table this gives 128 locations. The number of x and y-Locations must be chosen so that their product is equal to 128.

The larger of these two values is 8KB, which has a valid RAM with 32 locations. You must select the number of x and y-Locations so that their product is equal to 32. The BIST controller then ensures that only implemented locations in the data TCM dirty RAM are checked.

B.2.9 Data TCM data RAM

The *Data TCM* (DTCM) data RAM is tested as a single 64-bit wide RAM. The number of locations in this RAM depends on the size of the DTCM. The number of locations that must be specified in the BIST instruction depends on the size of the data TCM. You can find the size in Table B-10.

Table B-10 DTCM size and DTCM RAM locations

DTCM size	DTCM data RAM locations
4KB	512
8KB	1024
16KB	2048
32KB	4096
64KB	8192

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations in your DTCM data RAM.

For example, if you have an 8KB DTCM the DTCM data RAM has 1024 locations. You can therefore choose your maximum x and y-Addresses to be 512 and 2 respectively.

Note

If your implementation does not have a data TCM, you must not run an instruction to test this RAM.

B.2.10 Data cache tag RAMs

The data cache tag RAMs are tested as a single 48-bit wide RAM that has twice the number of locations as a single Tag RAM because of the way the RAMs are concatenated during BIST. The number of locations that must be specified in the BIST instruction depends on the size of the data cache.

For memory BIST, the number of locations that must be specified in the BIST instruction is shown in Table B-11.

Table B-11 Data cache size and data cache tag RAM locations

Data cache size	Data cache tag RAM locations
4KB	64
8KB	128
16KB	256
32KB	512
64KB	1024

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations given in Table B-11 for your specific data cache size.

B.2.11 Data cache data RAMs

The data cache data RAMs are tested as a single 64-bit wide RAM that has four times the number of locations as a single data RAM because of the way the RAMs are concatenated during BIST. The number of locations that must be specified to in the BIST instruction depends on the size of the data cache.

For memory BIST, the number of locations that must be specified in the BIST instruction is shown in Table B-12.

Table B-12 Data cache size and data cache tag RAM locations

Data cache size	Data cache tag RAM locations
4KB	512
8KB	1024
16KB	2048
32KB	4096
64KB	8192

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations given in the Table B-12 on page B-10 for your specific data cache size.

B.3 Connection

Connection of the ARM11 MBIST Controller to the ARM1136 processor is described in the *ARM1136 Implementation Guide*.

Appendix C
Integration with the ETB11

This appendix describes the relationship between the ETB11 address width and the number of ETB11 Trace RAM locations that you can test with the ARM11 MBIST Controller. It contains the following sections:

3.1 Instruction Register enables field

Table C-1 shows the bits in the enable field that select the RAMs to be tested. Only one group of RAMs can be selected at a time. Selecting multiple groups produces Undefined behavior.

Table C-1 Enable bit RAM selection

Enable bit	RAM group name	BIST Controller used
[11]	ETB Trace RAM	ETB11

C.2 Trace RAM

You can use the ARM11 MBIST Controller to test the ETB11 Trace RAM. The ETB11 Trace RAM is a single 24 or 32-bit wide RAM. The number of locations in this RAM depends on the value of ETB_ADDR_WIDTH. Refer to the *ETB11 Implementation Guide* for more information. For MBIST, the number of locations that you must specify in the MBIST instruction is shown in Table C-2.

Table C-2 ETB_ADDR_WIDTH and ETB11 Trace RAM locations

ETB_ADDR_WIDTH	Trace RAM locations
8	256
9	512
10	1K
11	2K
12	4K
13	8K
14	16K
15	32K
16	64K
17	128K
18	256K
19	512K
20	1M
21	2M
22	4M
23	8M

Note

You cannot use ARM MBIST to test your Trace RAM if it has more than 8M locations. This is 32MB if a 32-bit wide RAM is used.

You must choose values for the number of x and y-Locations so that their product is equal to the number of locations in your Trace RAM.

Note

You must not run an instruction to test Trace RAM if your implementation does not include an ETB11.

C.3 Connection

Connection of the ARM11 MBIST Controller to the ETB11 is described in the *ETB11 Implementation Guide*.

Index

The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.