ETB11™

Revision: r0p1

Technical Reference Manual

ARM®

ARM DDI 0275D

ETB11
Technical Reference Manual

Release Information

Change history

Date	Issue	Change
December 2002	A	First release
February 2003	B	ETB revision has changed to r0p1
May 2003	C	Description of ETMv1/ETMv2 supported removed.
August 2003	D	Preface and Index updated and corrected, Resets correctly described 2.10.2, and 3.2.5 RAM Data Register corrected.

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
ETB11 Technical Reference Manual

List of Tables
ETB11 Technical Reference Manual

List of Figures
ETB11 Technical Reference Manual

Preface

This preface introduces the *ARM11 Embedded Trace Buffer* (ETB11™) *Technical Reference Manual*. It contains the following sections:

About this document

This document is the technical reference manual for the *ARM11 Embedded Trace Buffer* (ETB11) r0p1.

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 document has been written for experienced hardware and software engineers who want to design or obtain trace information from chips that use ARM cores with the ETM facility.

Using this manual

This document is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an overview of the ETB11.

Chapter 2 *Functional Description*

Read this chapter for a description of the major functional blocks, configurability, read and write timing information, clocks, and resets.

Chapter 3 *Programmer's Model*

Read this chapter for a description of the registers and programming information.

Chapter 4 *Timing Requirements*

Read this chapter for a description of the ETB11 AC timing requirements.

Appendix A *Signal Descriptions*

This appendix lists the ETB11 signals.

Appendix B *Integrating the ETB11*

This appendix describes how to integrate the ETB11 if you are not using the ETK11 Integration Kit.

Conventions

This section describes the conventions that this manual uses:

- *Typographical*
- *Timing diagrams* on page xii
- *Signal naming* on page xii
- *Numbering* on page xiii.

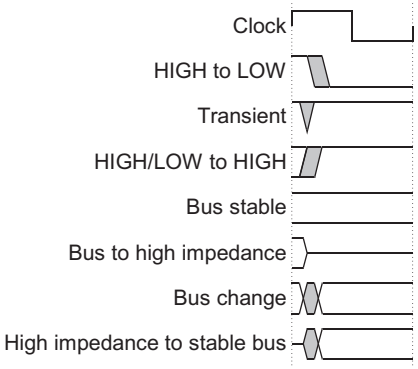
Typographical

This manual uses the following typographical conventions:

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 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 >	Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example: • `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>` • The Opcode_2 value selects which register is accessed.

Timing diagrams

This manual contains one or more timing diagrams. The figure named *Key to timing diagram conventions* explains the components used in these diagrams. When variations occur they have clear labels. You must not assume any timing information that is not explicit in the diagrams.



Key to timing diagram conventions

Signal naming

The level of an asserted signal 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 A Denotes *Advanced eXtensible Interface* (AXI) global and address channel signals.

Prefix B Denotes AXI write response channel signals.

Prefix C Denotes AXI low-power interface signals.

Prefix H Denotes *Advanced High-performance Bus* (AHB) signals.

Prefix n Denotes Active-LOW signals except in the case of AHB or *Advanced Peripheral Bus* APB reset signals. These are named **HRESETn** and **PRESETn** respectively.

Prefix P Denotes an APB signal.

Prefix R Denotes AXI read channel signals.

Prefix W Denotes AXI write channel signals.

Numbering

<size in bits>'<base><number>

This is a Verilog method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM Limited.

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.

ARM publications

This document contains information that is specific to the ETB11. Refer to the following documents for other relevant information:

- *ETB11™ Implementation Guide* (ARM DII 0067)
- *Embedded Trace Buffer (Rev 0) Technical Reference Manual* (ARM DDI 0242B)
- *Embedded Trace Macrocell Specification* (ARM IHI 0014)
- *ETM11RV™ Technical Reference Manual (ARM DDI 0233)*
- *ETM11RV Implementation Guide (ARM DII 0061)*
- *ETM11RV User Guide* (ARM DUI 0223)
- *ARM1136JF-S and ARM1136J-S Technical Reference Manual* (ARM DDI 0211)
- *ARM1136JF-S and ARM1136J-S Implementation Guide* (ARM DII 0022)
- *ARM1136JF-S and ARM1136J-S Test Chip Implementation Guide* (ARM DXI 0144)
- *ARM Architecture Reference Manual* (ARM DDI 0100)
- *ARM AMBA® Specification* (ARM IHI 0001)
- *Multi-ICE System Design Considerations* (ARM DAI 0072)
- *Multi-ICE® User Guide* (ARM DUI 0048)
- *Multi-layer AHB Overview* (ARM DVI 0045).

Feedback

ARM Limited welcomes feedback both on the ETB11 r0p1, and on the documentation.

Feedback on the ETB11

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 document

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

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

General suggestions for additions and improvements are also welcome.

Chapter 1
Introduction

This chapter introduces the *Embedded Trace Buffer* (ETB11) and its features. It contains the following sections:

1.1 About the Embedded Trace Buffer

As process speeds increase it is increasingly difficult to obtain trace information off a chip from an *Embedded Trace Macrocell* (ETM). This causes difficulties in maintaining acceptable signal quality or the signals have to be demultiplexed on to what can become a very large number of trace port pins.

The solution is to provide a buffer area on-chip where the trace information is stored, and read from the chip later, at a slower rate.

The ETB11 stores data produced by the ETM11RV. The buffered data can then be accessed by the debugging tools using a JTAG (IEEE 1149.1) interface, as shown in Figure 1-1.

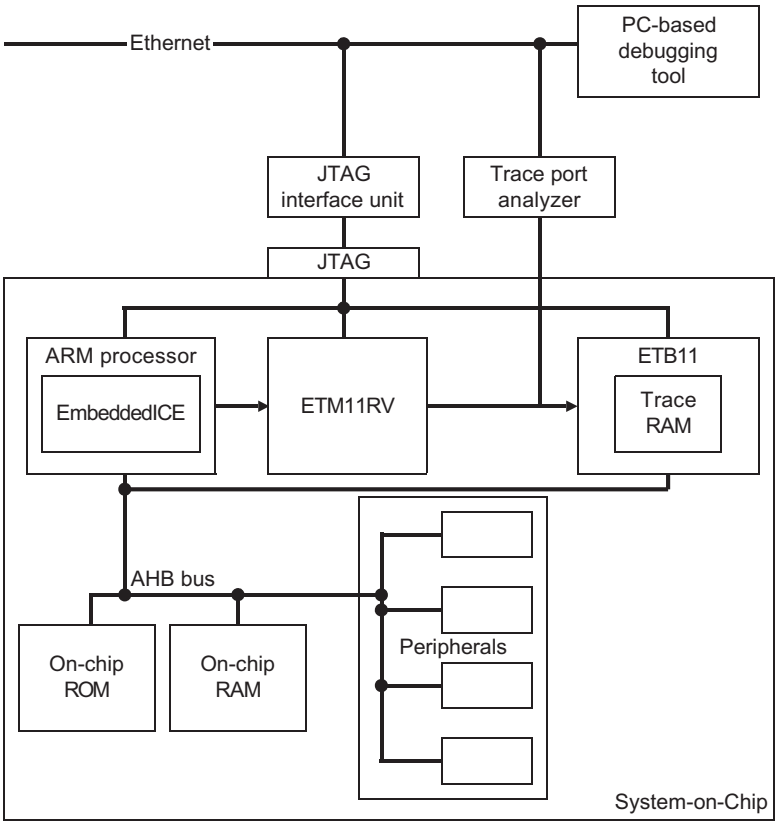


Figure 1-1 System-on-Chip debug implementation

Providing an on-chip buffer enables the trace data generated by the ETM11RV (at the system clock rate) to be read by the debugger at a reduced clock rate. This removes the requirement for high-speed pads for the trace data.

This buffered data can also be accessed through an AHB slave-based AHB interface included as part of the ETB11. This enables software running on the processor to read the trace data generated by the ETM11RV.

The major blocks shown in Figure 1-1 on page 1-2 are:

ETM11RV The ETM11RV monitors the ARM core buses and passes compressed information in real time to the ETB11 where it is stored for later retrieval. The data is then passed through the JTAG trace port to an interface unit. This is an external hardware device that passes the information from the trace port to a debugging tool, for example, a PC. The debug tool:

- retrieves data from the interface unit
- reconstructs a historical view of processor activity including data accesses
- configures the macrocell through the JTAG interface unit and port.

User-definable filters enable you to limit the amount of information captured in search of a bug, reducing upload time from the trace port analyzer.

EmbeddedICE

EmbeddedICE is a JTAG-based debugging environment for ARM microprocessors. EmbeddedICE provides the interface between the ARM source-level symbolic debugger, ARMxd, and an ARM microprocessor embedded within any ASIC. The ARMxd debugger is available for PC compatible and Sun workstation platforms.

EmbeddedICE provides:

- real-time address and data-dependent breakpoints
- single stepping
- full access and control of the ARM CPU
- access to the ASIC system.

EmbeddedICE also enables the embedded microprocessor to access the host system peripherals, for instance screen display, keyboard input and disk drive storage.

JTAG interface unit

Boundary scan is a methodology enabling complete controllability and observability of the boundary pins of a JTAG-compatible device by software control. This capability enables in-circuit testing without requiring specially designed in-circuit test equipment.

1.2 ETM versions and variants

The ETB11 is an enhanced version of the ETB that is designed to support the higher operating speeds of ETM11RV.

Although ETB11 supports older ETM protocols, it is intended for use with ETM11RV only. For this reason this document only describes details related to storing trace from ETM11RV. For details on using an ETB with other ETM products see the *Embedded Trace Buffer Technical Reference Manual*.

The history of the ETM is listed in Table 1-1.

Table 1-1 ETM major architecture versions

Name	Major architecture version
ETM7	ETMv1
ETM9	ETMv1
ETM10	ETMv2
ETM XScale	ETMv2
ETM10RV	ETMv3
ETM11RV	ETMv3

1.3 Silicon revision

This manual is for ETB11 r0p1. ETB11 r0p1 includes corrections for errata in ETB11 r0p0. Further information can be found in the ETB11 errata list.

Chapter 2
Functional Description

This chapter describes how the ETB11 operates. It contains the following sections:

2.1 Functional information

This section provides basic functional information:

- *Interfaces*
- *Global configurability*.

2.1.1 Interfaces

The on-chip ETB11 module has three primary interfaces:

- the trace port from the ETM11RV
- a five-pin IEEE 1149.1 (JTAG) interface
- an AHB slave interface to give software access to the ETB11 registers.

Additionally, the ETB11 accesses a trace RAM that must be implemented in the target technology. It is not possible to provide a single generic RAM interface block because of the large number of different RAMs that can be integrated. Therefore, the RAM interface is specified but the RAM block must be supplied by the system integrator. The RAM interface is described in *Trace RAM interface* on page 2-15.

Connection of the AHB interface is optional. If you do not require software access to the ETB registers or trace RAM, the AHB interface can be left unconnected. If this is done, all accesses to the ETB must be performed using the JTAG interface.

A block diagram of the ETB11 module is shown in Figure 2-1 on page 2-3.

2.1.2 Global configurability

The size of the trace RAM is configurable. See the ETB11 Implementation Guide for more information. Throughout this document **ETB_ADDR_WIDTH** refers to the address of the trace RAM. For example, if the trace RAM is 4KB, organized as 1024**x32-bit words, ETB_ADDR_WIDTH** is 10. ETB11 must always use 32-bit trace RAM with ETM11RV. Throughout this document, **ETB_DATA_WIDTH** is 32.

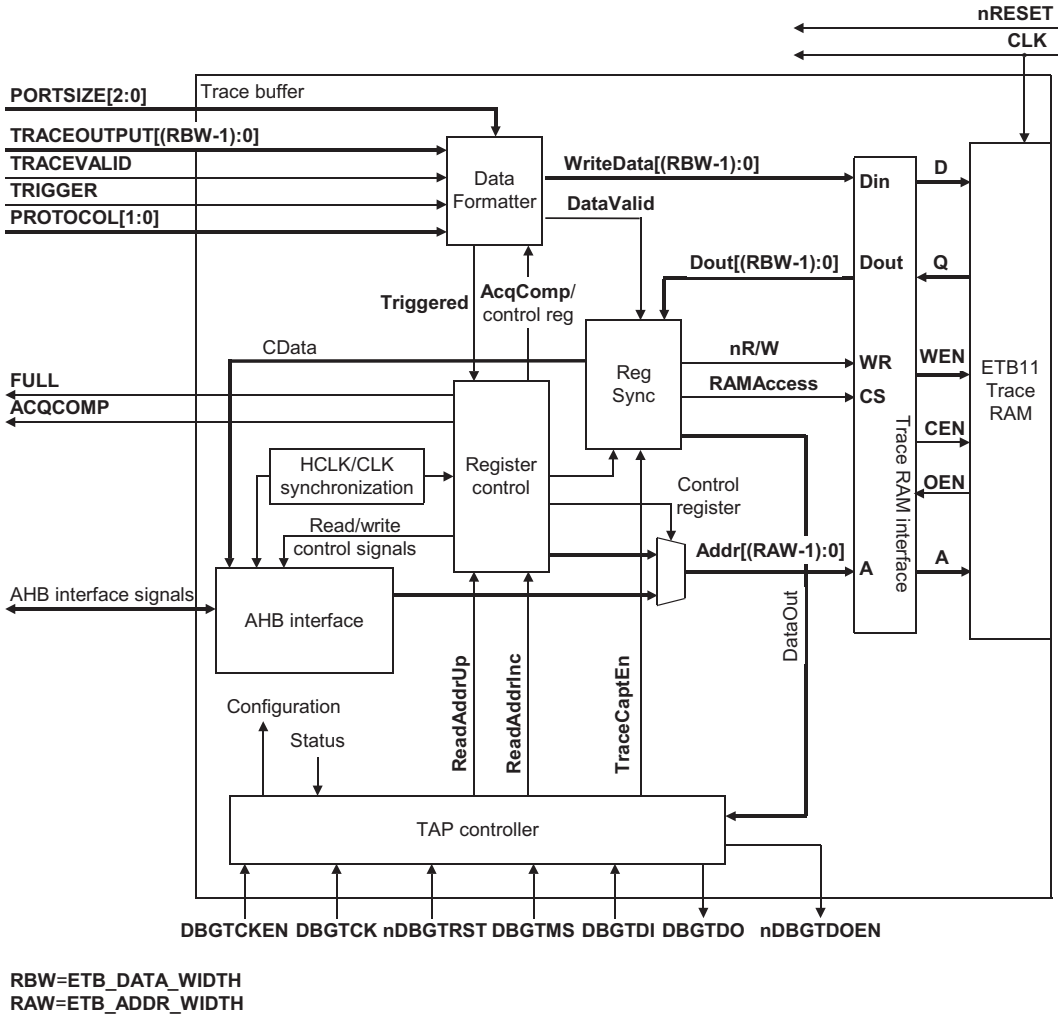


Figure 2-1 ETB11 module block diagram

2.2 Operation

The on-chip ETB11 operates as follows:

1. The ETM architecture version is supplied to the Data Formatter using the **PROTOCOL[1:0]** signal. This must be set to b10.
2. Configuration registers are set up through the TAP controller or through the AHB interface.
3. Trace capture is enabled using the control register.
4. Trace data is continuously written into the trace RAM while the ETB11 is enabled and the trigger counter value is nonzero. Once the ETM11RV indicates a trigger by asserting **TRIGGER**, the trigger counter decrements once per word of trace stored.

   When the trigger counter reaches zero the acquisition complete flag, AcqComp, is activated and trace capture stops. The value loaded into the trigger counter therefore sets the number of data words stored in the trace RAM after a trigger event.
5. The debugging tools through the TAP controller can read trace data stored in the trace RAM through the TAP controller or through the AHB interface.
   - To read data through the TAP controller you must:
     1. Disable trace capture. If trace capture is enabled when the RAM Data Register is accessed, the RAM value read is incorrect.
     2. Write the location that data is read from into the RAM Read Pointer Register.
     3. Read the RAM Data Register to return the data at the address stored in the RAM Read Pointer Register. The read address pointer increments after each RAM Data Register read and the next data value is automatically read from the RAM and stored in the RAM Data Register.

        Note

        You must precede the read access by a write to the RAM Read Pointer Register to ensure that the first RAM Read Register access returns valid data.
   - The trace data can also be read using the AHB interface. The trace RAM is aliased into the system memory space. This means that reading a value from the trace RAM requires an `LDR` type instruction from the trace RAM address space. The AHB interface can also write to the ETB11 memory when trace capture is disabled.

6. There are three internal status signals:
   - **AcqComp**
   - **Triggered**
   - **Full**.

   These can be read at any time while trace capture is in progress. The status signals are cleared when **TraceCaptEn** is cleared.

2.3 Control logic

Control logic monitors the **TraceCaptEn** signal, the status flags, and the **DataValid** signal from the Data Formatter. The logic enables a RAM write access cycle when there is valid data and trace capture is active. Trace capture is active while the **TraceCaptEn** signal is asserted and **TrgDelayCounter** is nonzero.

TraceCaptEn directly selects RAM write or read mode and the RAM address source. When **TraceCaptEn** is asserted all RAM access cycles are writes using the write pointer as the address. When **TraceCaptEn** is deasserted, RAM accesses are controlled by the AHB interface when **SoftwareCntl** (control register bit 4) is HIGH and **SWEN** is HIGH. Otherwise, all access cycles are reads using the RAM Read Pointer Register as the address. Timing diagrams showing the operation of the control logic are given in Figure 2-2 and Figure 2-3 on page 2-7.

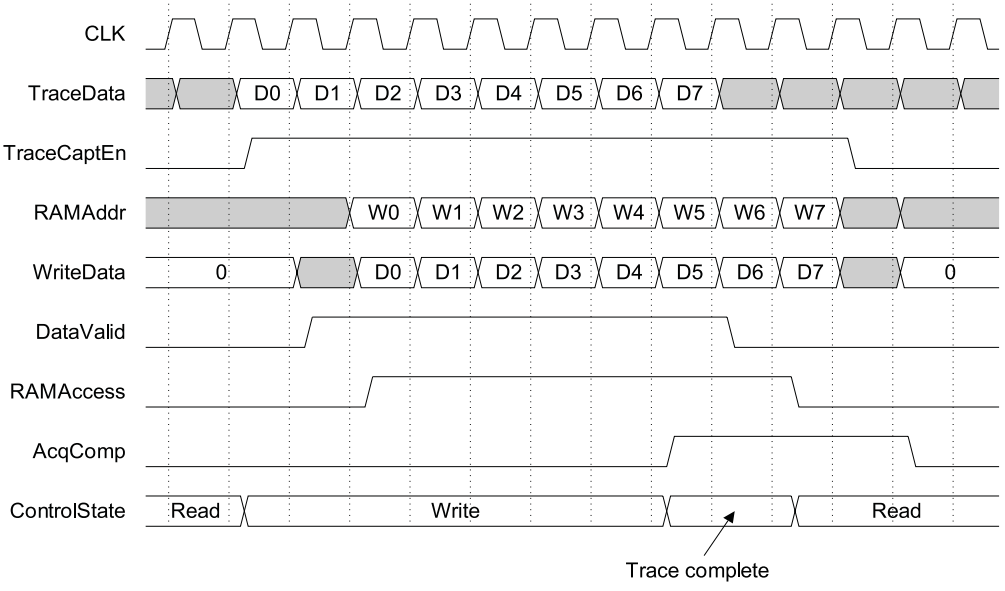


Figure 2-2 Trace capture operation

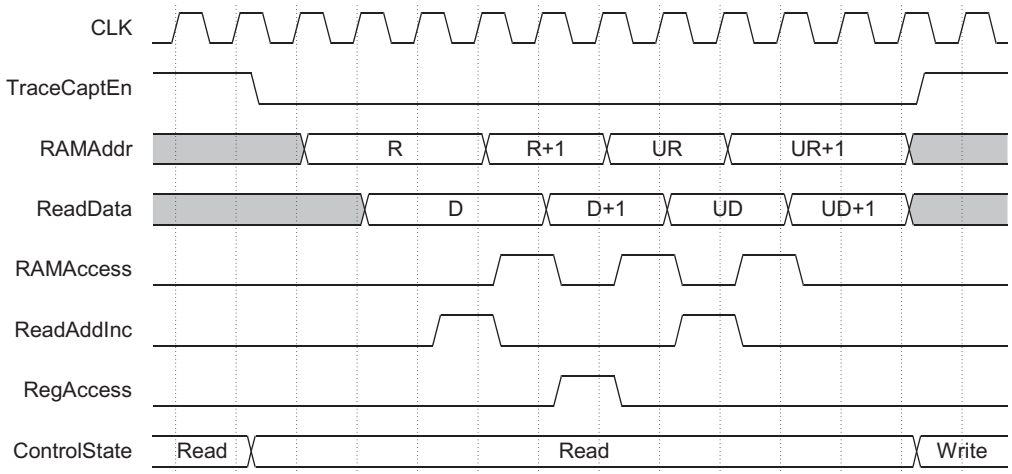


Figure 2-3 Trace read operation

2.4 Data Formatter

The Data Formatter is used to pack the trace data from ETMv1 ETMs. It is not used by ETM11RV. Trace data is written to the trace RAM one word at a time when **TRACEVALID** is asserted by the ETM11RV. You must set the port size to 32 bits and port mode to dynamic in the ETM11RV otherwise Unpredictable behavior might occur while using the ETB11. See the ETM Specification for details.

2.5 Trigger delay counter

The trigger delay counter, **TrgDelayCounter**, controls how many data words are written into the trace RAM after a trigger event. When a trigger event is detected, the **Triggered** flag is asserted. This enables the trigger delay counter which decrements every time a data word is written into the trace RAM. When **TrgDelayCounter** reaches zero the acquisition complete flag, **AcqComp**, is asserted. This prevents further writes to the trace RAM. The **AcqComp** flag is cleared when trace capture is disabled (**TraceCaptEn=**0). The state of the triggered flag can be read from the Status Register. The **Triggered** flag is cleared when trace capture is disabled.

AcqComp is output as a signal from the macrocell for possible use by ASIC logic.

2.6 Address generation

There are two RAM address pointer registers:

- the RAM Write Pointer Register is selected during trace capture
- the RAM Read Pointer Register is used as the RAM address source:
  - when trace capture is disabled
  - if software access to registers is disabled.

TraceCaptEn selects which pointer is used.

2.6.1 Write address generation

The RAM Write Pointer Register sets the trace RAM start address. It must be programmed before trace capture is enabled. The RAM Write Pointer Register increments when the **DataValid** flag is asserted by the Data Formatter. Reading the register returns the current RAM Write Pointer Register value. The RAM Write Pointer Register can be read back at any time. However if the TAP controller clock, **DBGTCK,** is asynchronous to **CLK**, the value might be indeterminate if read while trace capture is enabled. Therefore, the pointer must be accessed when **TraceCaptEn** is deasserted. The RAM Write Pointer Register is not affected by AHB writes to the RAM.

2.6.2 Read address generation

When trace capture and software access to registers are disabled, the RAM Read Pointer Register generates the RAM address. Updating the RAM Read Pointer Register automatically triggers a RAM access to ensure the RAM data output is up-to-date. Either writing to the RAM Read Pointer Register or reading the RAM Data Register updates the RAM Read Pointer Register. The RAM Read Pointer Register increments each time the RAM Data Register is read. The RAM Read Pointer Register can be accessed at any time. Reading the RAM Read Pointer Register returns its current value, the RAM read address. The RAM Read Pointer Register is not affected by AHB reads from the RAM.

2.7 BIST interface

ATPG testing can only test out the interface between the ETB11 RAM and the ETB11. It is unable to find faults in the actual RAM. A *Built-In Self Test* (BIST) interface is required that fully tests the RAM.

ATPG vectors are created using Synopsys TetraMax. These enable the shadow logic around the ETB11 RAM to be tested provided that the TetraMax has access to a model of the RAM used. Greater than 99% stuck-at fault coverage can be achieved.

A block diagram of the BIST interface is shown in Figure 2-4.

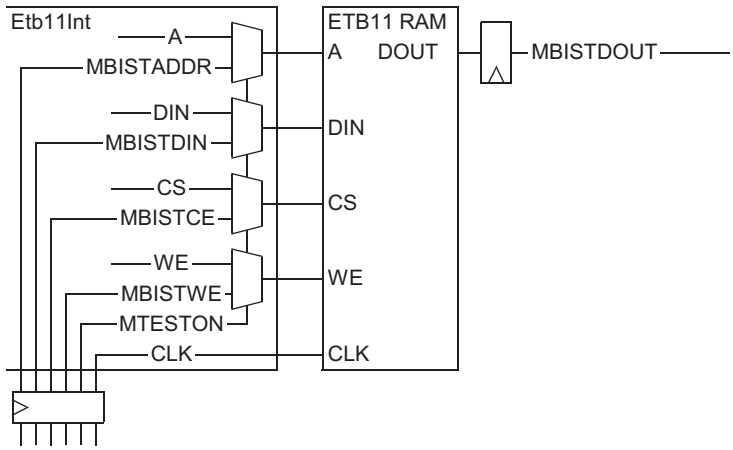


Figure 2-4 BIST interface block diagram

The **MTESTON** signal gives an external BIST controller access to the inputs and outputs of the ETB11 RAM. It is not possible for the ETB11 to operate in functional mode when the BIST is testing the RAM. When **MTESTON** is HIGH do not:

- set the TraceCaptEn bit
- write to the ETB11 RAM
- read from the ETB11 RAM.

2.8 TAP controller

All registers in the ETB11 are programmed through the TAP controller or the AHB interface. Registers are accessed through scan chain 0. The TAP controller is connected in series with other TAP controllers on the chip.

The registers described in this section are:

- *Test data registers*
- *Instruction Register* on page 2-13

2.8.1 Test data registers

There are two test data registers that can be connected between **DBGTDI** and **DBGTDO**. They are described in:

- *Bypass Register*
- *Scan chain 0* on page 2-13.

Bypass Register

This is a single bit register that can be selected as the path between **DBGTDI** and **DBGTDO** to enable the device to be bypassed during boundary-scan testing. When the BYPASS instruction is the current instruction in the instruction register, the SHIFT-DR state transfers serial data from **DBGTDI** to **DBGTDO** with a delay of one **DBGTCK** cycle. A logic 0 is loaded from the parallel input of the Bypass Register in the CAPTURE-DR state.

Scan chain 0

Scan chain 0 accesses a 40-bit register with the same structure as the ETM TAP controller shift register:

- a 32-bit data field
- a 7-bit address field
- a read/write bit.

Registers are read or written under the control of bit 39, the read/write bit, and the register access occurs when the TAP state machine passes through the Update-DR state. The registers are described in Chapter 3 *Programmer's Model*.

2.8.2 Instruction Register

The Instruction Register is four bits long.

There is no parity bit.

The fixed value loaded into the Instruction Register during the CAPTURE-IR controller state is b0001. The public instructions listed in Table 2-1 are supported.

Table 2-1 Supported public instructions

Instruction	Binary code	Description
SCAN_N	b0010	SCAN_N connects the 5-bit scan chain selection register between **DBGTDI** and **DBGTDO**.
INTEST	b1100	INTEST connects the scan register selected by the scan chain selection register, between **DBGTDI** and **DBGTDO**. Only scan chain 0 is implemented. Scan chain 0 is used to access all of the ETB11 registers.
IDCODE	b1110	The IDCODE instruction connects the device identification register (ID register) between **DBGTDI** and **DBGTDO**. The ID register is a 32-bit register. The value of the register is set by a `define TAP_ID_CODE`, in the `Etb11TapController.v` file. See *Identification Register, r0* on page 3-4 for the current ID value.
BYPASS	b1111	The BYPASS instruction connects a one-bit shift register, the BYPASS register, between **DBGTDI** and **DBGTDO**. **Note** The first bit shifted out is a zero.

2.8.3 Asynchronous clocks and testing in ETB11

ETB11 contains three clocking domains:

- **CLK**
- **DBGTCK**
- **HCLK**.

CLK and **DBGTCK** can be asynchronous or synchronous. ETM11RV does not have DBGTCK but uses **CLK** and **DBGTCKEN** to gate the flops in the JTAG block.

2.9 Trace RAM interface

This section describes the Trace RAM interface:

- *Signals*
- *Read access*
- *Write access* on page 2-16.

2.9.1 Signals

The TraceRAM interface block to the trace RAM uses the signals listed in Table 2-2.

Table 2-2 Trace RAM interface signals

Signal	Description
CLK	Clock
A	A configurable width address bus
CE	An active HIGH chip enable signal
WE	An active HIGH write enable signal
D	RAM data input bus
OE	An active HIGH output enable signal
Q	RAM data output bus

The timing requirements for the ETB11 are described in Chapter 4 *Timing Requirements*.

2.9.2 Read access

A timing diagram showing a read access from the Trace RAM to the Trace RAM interface is shown in Figure 2-5 on page 2-16.

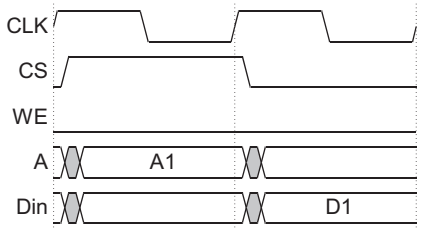


Figure 2-5 Read access from Trace RAM timing diagram

2.9.3 Write access

A timing diagram showing a write access to the Trace RAM from the Trace RAM interface is shown in Figure 2-6.

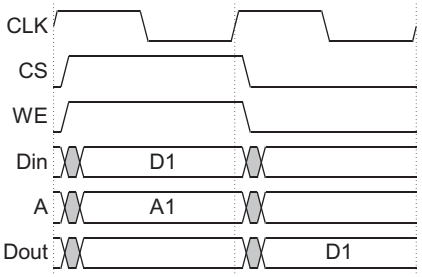


Figure 2-6 Write access to Trace RAM timing diagram

2.10 Clocks, and resets

This section describes:

- *Clocks*
- *Resets*.

2.10.1 Clocks

The ETB11 has three clock domains:

- TAP controller, clocked by **DBGTCK**
- memory-mapped peripheral, clocked by **HCLK**
- remainder of the system including ETB11 registers, clocked by **CLK**.

DBGTCK is synchronous to CLK when used with ETM11RV. Synchronization issues can therefore be ignored and are not discussed further. See *ETM versions and variants* on page 1-5 for details of other ETB products,

HCLK can be synchronous or asynchronous to CLK depending on your system design. Synchronization logic is provided for asynchronous designs. Register read and write accesses, and RAM read and write accesses, using the AHB interface are described in:

- *Read transfer* on page 2-19
- *Write transfer* on page 2-22.

2.10.2 Resets

There are the following resets:

- **nRESET** resets all of the ETB11 registers in the **CLK** domain. **nRESET** must be synchronized to **CLK** using the circuit shown in Figure 2-7.

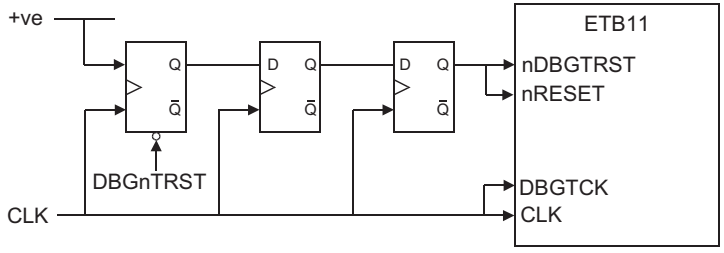


Figure 2-7 Example synchronizer

- **nDBGTRST** is the TAP controller reset signal used to reset the ETB11 TAP controller and other **DBGTCK** domain registers. **nDBGTRST** must be synchronized to CLK using the circuit shown in Figure 2-7.

- **HRESETn** is the AHB interface reset signal and is used to reset all of the registers in the AHB interface.

2.11 AHB transfers

This section describes:

- *Read transfer*
- *Write transfer* on page 2-22.

2.11.1 Read transfer

Two types of read transfer are described:

- *Asynchronous HCLK and CLK*
- *Synchronous HCLK and CLK* on page 2-22.

Asynchronous HCLK and CLK

When **HSEL** goes HIGH, this indicates that an AHB transfer involving the ETB11 AHB interface has started. On the same cycle that **HSEL** is asserted, the type of transfer and the address of the transfer are specified on **HWRITE** and **HADDR** respectively. **HReq** goes HIGH after **HSEL** goes HIGH, indicating the start of the synchronization period between the **HCLK** and **CLK** domain if the **SBYPASS** signal is LOW. **HReq** is registered twice in the **CLK** domain to form **CReq**. When **CReq** goes HIGH, the address value on **HADDRReg**, the registered version of **HADDR** that remains valid until **HReq** goes LOW, is valid. The **CS** and **CRegRead** signals that control read access of the ETB11 RAM and the ETB11 registers go HIGH for one cycle after **CReq** goes HIGH.

Data is returned from the ETB11 RAM or the ETB11 registers and registered into **CData**. **CAck** goes HIGH to indicate that the read value has been retrieved.

CAck is registered twice in the **HCLK** domain to form **HAck**. After **HAck** goes HIGH, **HRDATAMEM** gets the value of **MuxedData** (a multiplexed version of the data returned from RAM, registers, and **CData**) and **HREADYMEM** goes HIGH indicating to the AHB bus master that the data on **HRDATAMEM** is valid.

HReq then goes LOW, indicating that the AHB transfer has finished. This causes **CAck** to go LOW one cycle after **CReq** goes LOW.

Finally, **HAck** goes LOW, finishing the read cycle.

How the **CReq**, **CAck**, and **HAck** signals are produced is shown in Figure 2-8 on page 2-20.

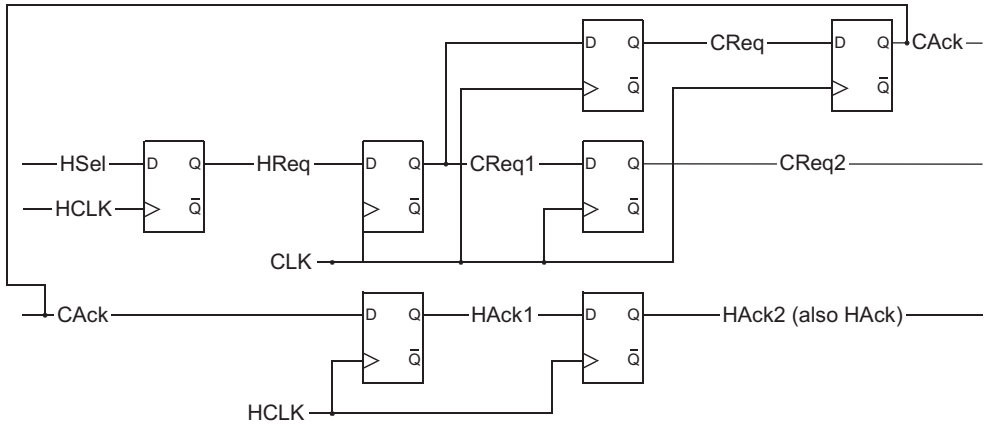


Figure 2-8 Synchronization logic between HCLK and CLK domains

A software read cycle with **HCLK** and **CLK** asynchronous is shown in Figure 2-9 on page 2-21.

In synchronous designs where **HCLK** is derived from **CLK**, **HCLKEN** is used to control the generation of **HCLK** and **CLK** is connected to the **HCLK** input of the ETB11.

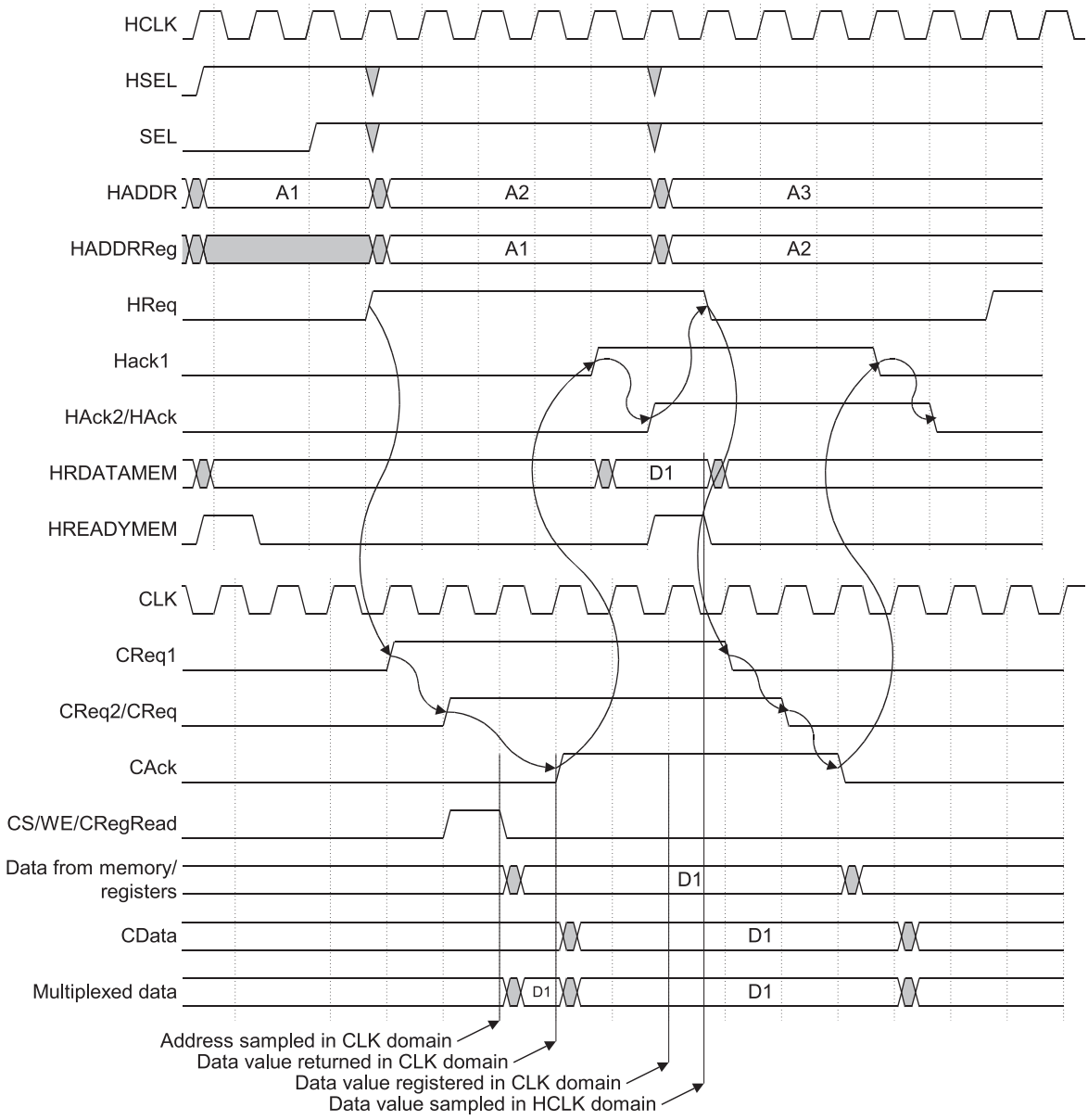


Figure 2-9 Software read cycle with asynchronous CLK and HCLK

Synchronous HCLK and CLK

When **HCLK** and **CLK** are synchronous, **HCLKEN** is used to control the generation of **HCLK**, which is derived from **CLK**. This means that the **HCLK** rising edge always corresponds to a **CLK** rising edge. Therefore, the read transfer does not require any synchronization logic (the **SBYPASS** signal is HIGH).

A software read cycle with **CLK** and **HCLK** synchronous is shown in Figure 2-9 on page 2-21. The pipelined nature of the ETB11 data means that data takes more than a single cycle to perform a read (and write) operation. Wait states are inserted until the read cycle is completed.

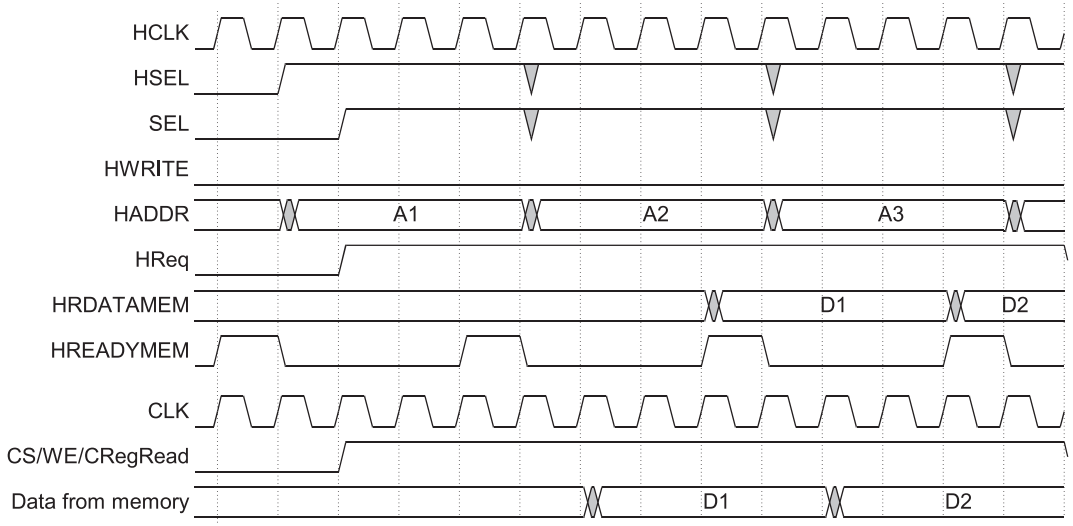


Figure 2-10 Software read cycle with synchronous CLK and HCLK

2.11.2 Write transfer

Two types of write transfer are described:

- *Asynchronous HCLK and CLK*
- *Synchronous HCLK and CLK* on page 2-24.

Asynchronous HCLK and CLK

The relationship between **HReq** and **CReq**, and **CAck** and **HAck** is the same as it is for a read transfer.

When **CReq** goes HIGH, the data is already valid on **HWriteData** (the registered version of **HWDATA**), and the address is already valid on **HADDRReg**. The **CS**,**WE**, and **RegWrite** signals that control write access of the ETB11 RAM and the ETB11 registers then go HIGH for one cycle after **CReq** goes HIGH to perform the write access. **CAck** then goes HIGH one cycle after **CReq** goes HIGH to indicate that the write data has been used in the **CLK** domain.

At the same time that **HAck** goes HIGH, **HREADYMEM** goes HIGH indicating to the AHB bus master that the data has been written to its destination.

HReq then goes LOW, indicating that the AHB transfer has finished. This, in turn, causes **CAck** to go LOW one cycle after **CReq** goes LOW.

Finally, **HAck** goes LOW, finishing the write cycle.

A software write cycle with **CLK** and **HCLK** asynchronous is shown in Figure 2-11 on page 2-24.

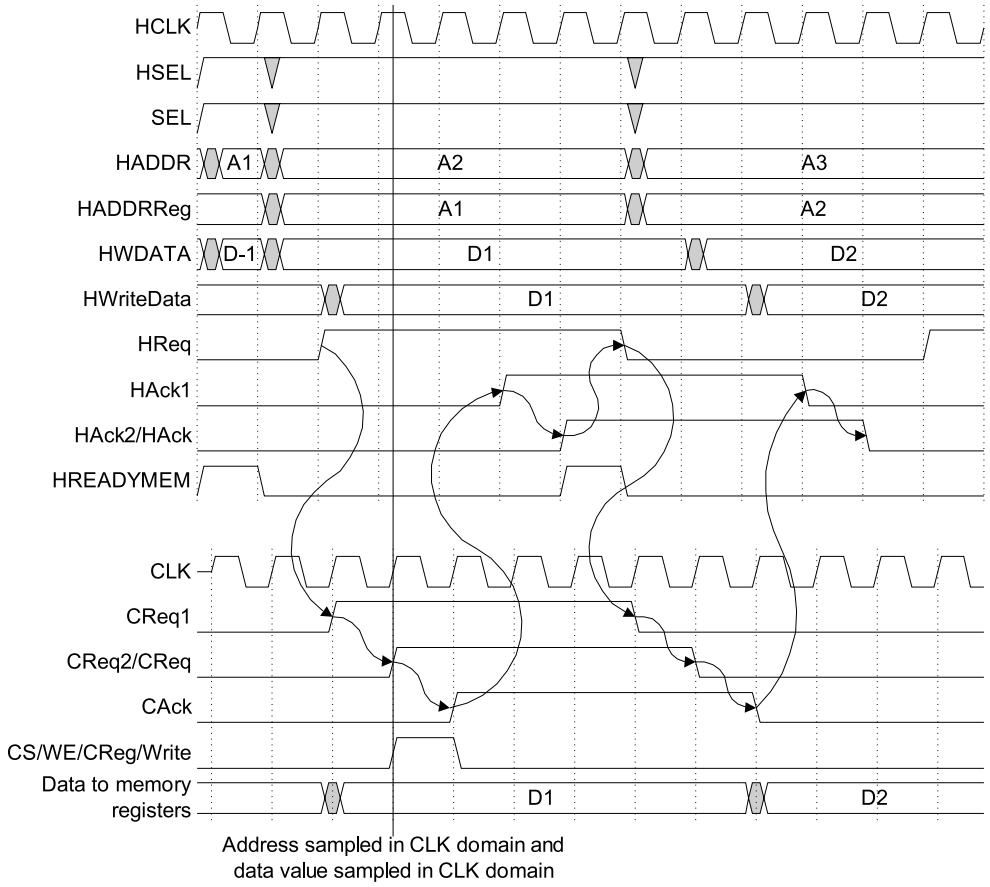


Figure 2-11 Software write cycle with asynchronous CLK and HCLK

Synchronous HCLK and CLK

Software write cycles with **CLK** and **HCLK** synchronous (the **SBYPASS** signal is HIGH) is shown in Figure 2-12 on page 2-25.

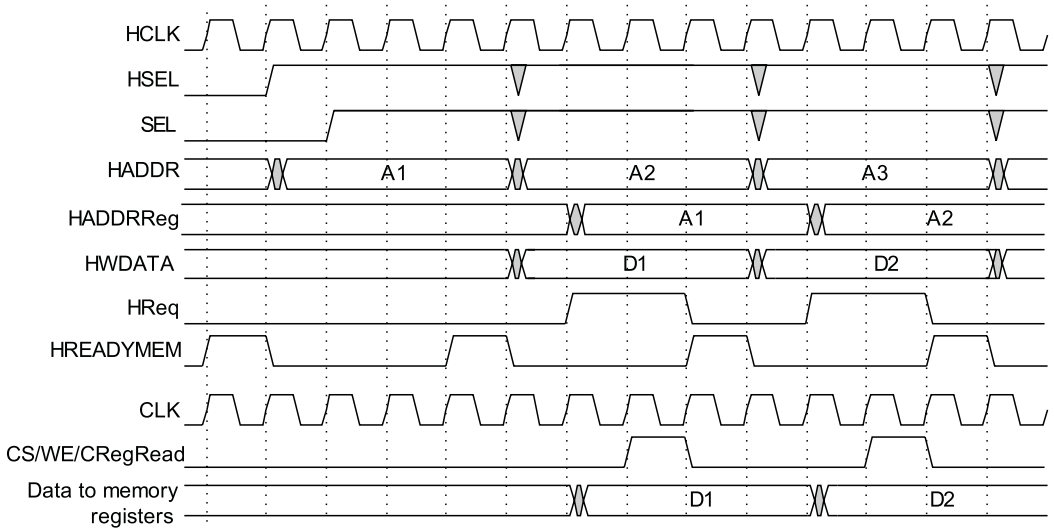


Figure 2-12 Software write cycle with synchronous CLK and HCLK

Chapter 3
Programmer's Model

This chapter describes the ETB11 registers and provides details required when programming the buffer. It contains the following sections:

3.1 About the programmer's model

This section provides general information relevant to the ETB11 programmer's model:

- *Register fields*
- *Register map*.

3.1.1 Register fields

You must not access reserved or unused address locations because this can result in unpredictable behavior.

All reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated within the relevant text.

All registers bits are reset to a logic 0 by a reset unless otherwise stated in the relevant text.

All registers support read and write accesses unless otherwise stated in the relevant text. A write updates, and a read returns, the contents of the register.

3.1.2 Register map

The ETB11 register map is shown in Table 3-1.

Table 3-1 Register map

Register number		Type	Description
Decimal	Binary		
0	b000 0000	Read-only	Identification Register
1	b000 0001	Read-only	RAM Depth Register
2	b000 0010	Read-only	RAM Width Register
3	b000 0011	Read-only	Status Register
4	b000 0100	Read-only	RAM Data Register
5	b000 0101	Read/write	RAM Read Pointer Register
6	b000 0110	Read/write	RAM Write Pointer Register

Table 3-1 Register map (continued)

Register number		Type	Description
Decimal	Binary		
7	b000 0111	Read/write	Trigger Counter Register
8	b000 1000	Read/write	Control Register
9-127	b000 1001- b111 1111	-	Reserved

3.2 Register descriptions

This section describes the ETB11 registers:

- *Identification Register, r0*
- *RAM Depth Register, r1* on page 3-5
- *RAM Width Register, r2* on page 3-5
- *Status Register, r3* on page 3-6
- *RAM Data Register, r4* on page 3-7
- *RAM Read Pointer Register, r5* on page 3-7
- *RAM Write Pointer Register, r6* on page 3-8
- *Trigger Counter Register, r7* on page 3-9
- *Control Register, r8* on page 3-10.

3.2.1 Identification Register, r0

The Identification Register enables the TAP controller to be identified by Multi-ICE, or any other run control device. It is a read-only 32-bit register. Table 3-2 describes the bits.

Table 3-2 Identification register description

Bit number	Value	Description
[31:28]	`0x2`	Version
[27:12]	`0xB900`	Part number
[11:1]	0x787	Manufacturer identity. This field always indicates ARM as a manufacturer and cannot be changed to indicate the silicon manufacturer.
[0]	1	Always 1 as defined by IEEE 1149.1

For the current implementation the ID value is `32'h2B900F0F`. It is recommended that tools check the value of bits [27:1] to detect that the ETB11 is present.

3.2.2 RAM Depth Register, r1

The RAM Depth Register is a read-only register that indicates the number of addressable entries in the RAM to the trace tools. Register bit allocations for the RAM Depth Register are listed in Table 3-3.

Table 3-3 RAM Depth Register bit allocations

Bit number	Type	Function
[31:0]	Read-only	RAM data depth This value is configurable in the RTL but must be fixed when the ETB11 is synthesized.

3.2.3 RAM Width Register, r2

This is a read-only register, that indicates the number of bits in each addressable entry in the RAM to the trace tools. This is always 32 when used with ETM11RV. Register bit allocations for the RAM Width Register are listed in Table 3-4.

Table 3-4 RAM Width Register bit allocations

Bit number	Type	Function
[31:6]	-	Reserved
[5:0]	Read-only	RAM data width b100000=32-bit data

3.2.4 Status Register, r3

The read-only Status Register contains ETB11 status flags. You can read it at any time. Register bit allocations for the Status Register are listed in Table 3-5.

Table 3-5 Status Register bit allocations

Bit number	Name	Function
[31:4]	-	Reserved.
[3]	DFEmpty	Data Formatter pipeline empty. This bit is required because when tracing is disabled there might still be some trace data in the Data Formatter pipeline. This is drained within a few cycles after trace capture is disabled (see *Control Register, r8* on page 3-10). You can ensure that all trace data has been written to the buffer by waiting for this bit to be set.
[2]	AcqComp	Acquisition complete. The acquisition complete flag indicates that the trigger counter is zero.
[1]	Triggered	Triggered. The Triggered bit is set when a trigger has been observed, from the ETM11RV.
[0]	Full	RAM full. The flag indicates when the RAM write pointer has overflowed or wrapped around.

The Status Register is cleared on the cycle that trace capture is enabled, see *Control Register, r8* on page 3-10.

The recommended procedure for use of the tools is:

1. Program the ETB11 registers.
2. Enable tracing.
3. Wait until the AcqComp bit is set.
4. Disable tracing.
5. Wait until the DFEmpty bit is set.
6. Read the trace.

3.2.5 RAM Data Register, r4

You can use the read-only RAM Data Register, while trace capture is disabled, to return the contents of the ETB11 SRAM location addressed by the RAM Read Pointer Register. Reading this register increments the RAM Read Pointer Register and triggers a RAM access cycle. Register bit allocations for the RAM Data Register are listed in Table 3-6.

Table 3-6 RAM Data Register bit allocations

Bit number	Function
[31:0]	RAM data. Returns the captured trace data.

Caution

You cannot access the RAM Data Register while trace capture is enabled.

This register is not accessible from the AHB interface because the RAM is memory-mapped. See *Software access to the ETB11 using the AHB interface* on page 3-11.

3.2.6 RAM Read Pointer Register, r5

This read/write register enables you to set and read the pointer used to read entries from the RAM. Register bit allocations for the RAM Read Pointer Register are listed in Table 3-7.

Table 3-7 RAM Read Pointer Register bit allocations

Bit number	Function
[31: ETB_ADDR_WIDTH]	Reserved
[(ETB_ADDR_WIDTH-1):0]	RAM read pointer

Note

`ETB_ADDR_WIDTH` is a constant used to define the width of the trace RAM address bus.

When the RAM Read Pointer Register is written to, the read address updated (ReadAddrUp) flag is activated. This initiates a RAM access.

When read, the RAM Read Pointer Register returns the current trace RAM read address.

You cannot write to this register if **TraceCaptEn** is HIGH.

This register is not accessible from the AHB interface because the RAM is memory-mapped. See *Software access to the ETB11 using the AHB interface* on page 3-11.

3.2.7 RAM Write Pointer Register, r6

This read/write register enables you to set and read the pointer used to write entries from the ETM11RV into the RAM. Register bit allocations for the RAM Write Pointer Register are listed in Table 3-8.

Table 3-8 RAM Write Pointer Register bit allocations

Bit number	Function
[31: ETB_ADDR_WIDTH]	Reserved
[(ETB_ADDR_WIDTH-1):0]	RAM write pointer

Note

`ETB_ADDR_WIDTH` is a constant used to define the width of the trace RAM address bus.

You cannot write to this register if **TraceCaptEn** is HIGH.

The initial value of the trace memory write address pointer is set by writing to the RAM Write Pointer Register. You must program the pointer before tracing starts. In most circumstances the initial pointer value is zero. During trace capture, the pointer increments when the DataValid flag is asserted by the Data Formatter. When the RAM Write Pointer Register value increments from its maximum value back to zero, the Full flag is set.

Reading the RAM Write Pointer Register returns the current trace RAM write address. During trace capture, write pointer changes might be asynchronous. For example, if **DBGTCK** is not related to the CPU clock, in this case the read back value is indeterminate. This register only returns the correct write pointer value when trace acquisition is stopped (when the AcqComp status bit is set or trace capture is disabled).

3.2.8 Trigger Counter Register, r7

This read/write register disables writes to the trace RAM after a defined number of words have been stored, following the trigger event. The counter is used as follows:

Trace after The counter is set to a large value (slightly less than the number of entries in the RAM).

Trace before The counter is set to a small value.

Trace about The counter is set to half the number of entries in the ETB11 RAM.

The register bit allocations for the Trigger Counter Register are listed in Table 3-9.

Table 3-9 Trigger Counter Register bit allocations

Bit number	Function
[31: ETB_ADDR_WIDTH]	Reserved.
[ETB_ADDR_WIDTH-1:0]	Trigger count. The number of datawords written into the trace RAM following the trigger event is given by the equation: Count = Trigger Counter Register value + 1 If this is set to 0, the trigger is ignored.

Note

`ETB_ADDR_WIDTH` is a constant used to define the width of the trace RAM address bus.

When written, the value of the Trigger Counter Register is set. You must update this register before enabling trace capture, failure to do so can result in unexpected trace behavior.

Reading the Trigger Counter Register samples the value of the trigger counter. During trace capture, the value of the counter can change at any time. Therefore, if a read is performed asynchronously the returned value might be unreliable.

You cannot write to this register if **TraceCaptEn** is HIGH.

3.2.9 Control Register, r8

The Control Register is used to enable/disable the trace capture using bit 0. The Control Register bit allocations are listed in Table 3-10.

Table 3-10 Control Register bit allocations

Bit number	Name	Type	Function
[31:3]	-	-	Reserved
[2]	SoftwareCntl	Read/write (JTAG only)	Controls software and hardware register access: 1 = Software register access 0 = JTAG register access
[1]	Demux	Read/write	Demultiplexed memory support: 1 = Demultiplexed support enabled 0 = Demultiplexed support disabled
[0]	TraceCaptEn	Read/write	Trace capture enable: 1 = Trace capture is enabled 0 = Trace capture is disabled

Control register bit 0 drives the **TraceCaptEn** signal.

When **TraceCaptEn** is set the ETB11 SRAM is in write mode. If you attempt to read the RAM Data Register read while **TraceCaptEn** is set, then the contents of the SRAM are altered, resulting in the corruption of any stored trace data.

The ETB11 starts up from reset with the SoftwareCntl bit enabled. The value of this bit can only be changed through the TAP controller. It is cleared when the `INTEST` instruction is selected by the TAP controller and is set by writing a 1 as normal. While this bit is clear, all accesses to the register by the AHB interface are ignored.

3.3 Software access to the ETB11 using the AHB interface

The AHB interface:

- is a slave-based interface
- resides at a user-defined block of system memory, for example `0x13800000`
- enables software access to the ETB11 registers and RAM
- has read and write access to the ETB11 registers.

ETB memory is aliased into the interface memory space, so that software can read out the trace information stored in the memory and additionally write to the ETB11 memory.

Registers contained in the ETB11 that require software access are listed in Table 3-11.

Table 3-11 Registers that require software access

Register number	Description	Location
000 0000	Identification Register	Register base address (for example, `0x13800000`)
000 0001	RAM Depth Register	Register base address + `0x4` (for example, `0x13800004`)
000 0010	RAM Width Register	Register base address + `0x8` (for example, `0x13800008`)
000 0011	Status Register	Register base address + `0xC` (for example, `0x1380000C`)
000 0110	RAM Write Pointer Register	Register base address + `0x18` (for example, `0x13800018`)
000 0111	Trigger Counter Register	Register base address + `0x1C` (for example, `0x1380001C`)
000 1000	Control Register	Register base address + `0x20` (for example, `0x13800020`)
-	Aliased trace RAM	RAM base address (for example, `0x13900000`)

The base addresses of the ETB11 registers and the RAM are defined by the AHB decoder.

Software access to registers is only enabled when bit 2 of the control register (SoftwareCntl) is set to 1. This is the default and is set to 1 on reset.

Additionally, software access to ETB11 RAM is only enabled when bit 0 of the control register is set to 0 (ETB11 is enabled).

The interface contains an input signal called **SWEN**. When this signal is LOW, the interface is disabled. The AHB interface is enabled if **SWEN** is HIGH.

The AHB interface is enabled when the SoftwareCntl bit of the control register and the **SWEN** signal are ANDed.

The ETB11 registers and RAM accesses are controlled by separate read/write ports and each has their own separate **HSEL** input. This enables the ETB11 RAM to share the address space with main memory.

3.3.1 Restrictions on use of the AHB interface

The AHB interface can be used for two purposes:

- to read trace data captured by the ETB11 from software
- as system memory when tracing is not required.

When using the AHB interface to read trace data, the following are not permitted:

- byte or halfword accesses when the memory does not support them
- unaligned accesses
- transfers larger than a word, for example 64-bit transfers.

Only aligned, single-word accesses are permitted.

Register access supports aligned, single-word accesses only.

Use of the AHB interface as system memory requires careful system design to ensure that the memory is not required for system use when tracing is required. If you want to use the AHB interface for this purpose, you must ensure the following:

- the memory supports byte writes (see the *Embedded Trace Buffer Implementation Guide*)
- if connected to a 64-bit AHB bus, 64 bit operations are supported, see *Connecting the ETB11 in a 64-bit AHB system* on page B-4).

This is to ensure that all ARM load/store instructions are supported.

Chapter 4
Timing Requirements

The timing requirements for the ETB11 interfaces are defined in this chapter. It contains the following sections:

4.1 AHB interface

The timings for the AHB interface signals are shown in Figure 4-1.

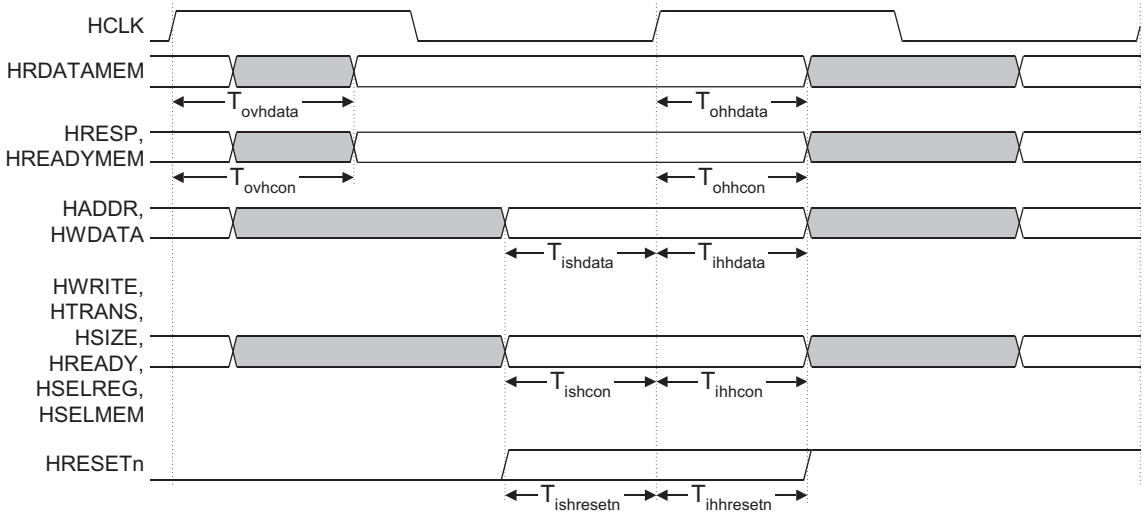


Figure 4-1 AHB interface signals

The timing requirements for the AHB interface are listed in Table 4-1. All figures are expressed as percentages of the **HCLK** period at maximum operating frequency.

Note

A 0% figure in Table 4-1 indicates the hold time to clock edge plus the maximum clock skew for internal clock buffering.

Table 4-1 AHB interface timing requirements

Parameter	Description	Max	Min
$T_{ovhdata}$	Rising **HCLK** to **HRDATAMEM** valid	40%	-
$T_{ohhdata}$	**HRDATAMEM** hold time from **HCLK** rising	-	>0%
T_{ovhcon}	Rising **HCLK** to AHB control outputs valid	40%	-
T_{ohhcon}	AHB control outputs hold time from **HCLK** rising	-	>0%
$T_{ishdata}$	AHB data inputs setup to rising **HCLK**	-	30%

Table 4-1 AHB interface timing requirements (continued)

Parameter	Description	Max	Min
$T_{ihhdata}$	AHB data inputs hold from rising **HCLK**	-	0%
T_{ishcon}	AHB control inputs setup to rising **HCLK**	-	30%
T_{ihhcon}	AHB control inputs hold from rising **HCLK**	-	0%
$T_{ishresetn}$	**HRESETn** input setup to rising **HCLK**	-	30%
$T_{ihhresetn}$	**HRESETn** input hold from rising **HCLK**	-	0%

4.2 CLK domain

The timings for the **CLK** domain signals are shown in Figure 4-2.

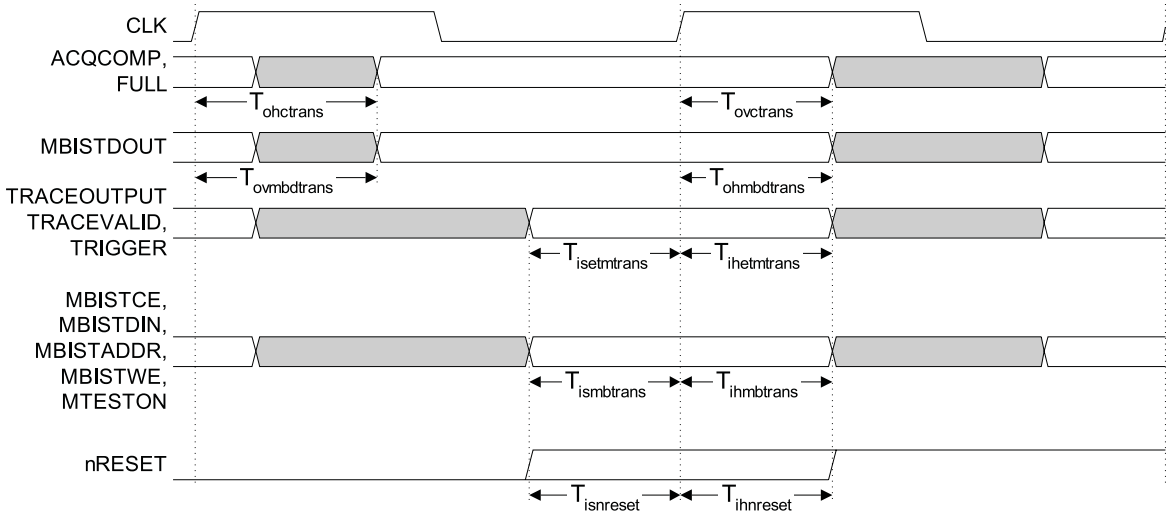


Figure 4-2 CLK domain signals

The timing requirements for the **CLK** domain signals are listed in Table 4-2. All figures are expressed as percentages of the **CLK** period at maximum operating frequency.

Note

A 0% figure in Table 4-2 indicates the hold time to clock edge plus the maximum clock skew for internal clock buffering.

Table 4-2 CLK domain timing requirements

Parameter	Description	Max	Min
$T_{ovctrans}$	Rising **CLK** to **CLK** domain outputs valid	40%	-
$T_{ohctrans}$	**CLK** domain outputs hold time from **CLK** rising		>0%
$T_{ovmbdtrans}$	Rising **CLK** to **MBISTDOUT** output valid	60%	-
$T_{ohmbdtrans}$	**MBISTDOUT** output hold time from **CLK** rising	-	>0%

Table 4-2 CLK domain timing requirements (continued)

Parameter	Description	Max	Min
$T_{ismbtrans}$	**MBIST** inputs setup to rising **CLK**	-	40%
$T_{ihmbtrans}$	**MBIST** inputs hold from rising **CLK**	-	0%
$T_{isetmtrans}$	ETM interface inputs setup to rising **CLK**	-	40%
$T_{ihetmtrans}$	ETM interface inputs hold from rising **CLK**	-	0%
$T_{isnreset}$	**nRESET** input setup to rising **CLK**	-	40%
$T_{ihnreset}$	**nRESET** input hold from rising **CLK**	-	0%

4.3 IEEE1149.1 interface

The IEEE1149.1 interface signals are shown in Figure 4-3.

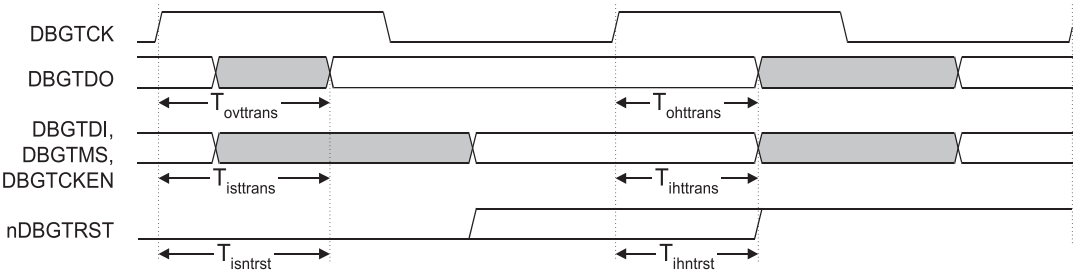


Figure 4-3 IEEE1149.1 interface signals

The timing requirements for the IEEE1149.1 interface trace data signals are listed in Table 4-3. All figures are expressed as percentages of the **DBGTCK** period at maximum operating frequency.

Note

A 0% figure in Table 4-3 indicates the hold time to clock edge plus the maximum clock skew for internal clock buffering.

Table 4-3 IEEE1149.1 interface timing requirements

Parameter	Description	Max	Min
$T_{ovttrans}$	Rising **DBGTCK** to **DBGTDO** output valid	40%	-
$T_{ohttrans}$	**DBGTDO** output hold time from **DBGTCK** rising	-	>0%
$T_{isttrans}$	JTAG inputs setup to rising **DBGTCK**	-	40%
$T_{ihttrans}$	JTAG inputs hold from rising **DBGTCK**	-	0%
$T_{isntrst}$	**nDBGTRST** input setup to rising **DBGTCK**	-	40%
$T_{ihntrst}$	**nDBGTRST** input hold from rising **DBGTCK**	-	0%

Appendix A
Signal Descriptions

This appendix describes the ETB11 input and output signals. It contains the following sections:

A.1 Signal properties and requirements

To ensure ease of integration of the ETB11 into embedded applications, and to simplify synthesis flow, the following design techniques have been used:

- a single rising edge clock times all activity
- all signals and buses are unidirectional
- all inputs are required to be synchronous to the relevant clock (**CLK**, **DBGTCK**, or **HCLK**).

These techniques simplify the definition of the top-level ETB11 signals because all outputs change from the rising edge and all inputs are sampled with the rising edge of the clock. In addition, all signals are either input or output only. Bidirectional signals are not used.

Note

You must use external logic to synchronize asynchronous signals (for example, interrupt sources) before applying them to the ETB11.

A.2 Signal descriptions

Table A-1 lists the ETB11 input and output signals.

Table A-1 Signal descriptions

Signal Name	Clock domain	Type	Description
ACQCOMP[a]	**CLK**	Output	When HIGH indicates that trace acquisition is complete.
CLK	-	Input	This clock times all operations in the Trace Buffer.
DBGTCK	-	Input	Test clock.
DBGTCKEN	**DBGTCK**	Input	Test clock enable. Enable term for **DBGTCK** domain.
DBGTDI	**DBGTCK**	Input	Test data input.
DBGTDO	**DBGTCK**	Output	Test data output.
DBGTMS	**DBGTCK**	Input	Test mode select.
FULL[a]	**CLK**	Output	When HIGH indicates that the ETB11 RAM has overflowed.
HADDR[31:0]	**HCLK**	Input	The 32-bit AHB system address bus.
HCLK	-	Input	AHB system bus clock.
HCLKEN		Input	Enable term for **HCLK** domain.
HRDATAMEM [31:0]	**HCLK**	Output	The 32-bit AHB read data bus.
HREADY	**HCLK**	Input	When HIGH indicates that a transfer has finished on the AHB bus.
HREADYMEM	**HCLK**	Output	When LOW indicates that the ETB11 is carrying out an AHB transfer.
HRESETn	**HCLK**	Input	AHB system bus reset.

Table A-1 Signal descriptions (continued)

Signal Name	Clock domain	Type	Description
HRESPMEM[1:0]	**HCLK**	Output	Memory-mapped peripheral AHB transfer response. Provides additional information on the transfer status: 00 = OKAY 01 = ERROR 10 = RETRY 11 = SPLIT. ETB11 does not support splits and retries. ETB11 outputs an OK or ERROR response only/
HSELMEM	**HCLK**	Input	Indicates that the ETB11 RAM has been selected for an AHB transfer.
HSELREG	**HCLK**	Input	Indicates that the ETB11 registers have been selected for an AHB transfer.
HSIZE[2:0]	**HCLK**	Input	Indicates the size of the AHB transfer.
HTRANS[1:0]	**HCLK**	Input	Indicates the type of AHB transfer: 00 = IDLE 01 = BUSY 10 = NONSEQ 11 = SEQ.
HWDATA[31:0]	**HCLK**	Input	The 32-bit AHB write data bus.
HWRITE	**HCLK**	Input	When HIGH indicates an AHB write transfer. When LOW indicates an AHB read transfer.
MBISTADDR[ETB_ADDR_WIDTH-1:0]	**CLK**	Input	Address Bus for external BIST controller (active when **MTESTON** is HIGH).
MBISTCE	**CLK**	Input	Active HIGH chip select for external BIST controller (active when MTESTON is HIGH).
MBISTDIN[ETB_DATA_WIDTH-1:0]	**CLK**	Input	Write data bus for external BIST controller (active when **MTESTON** is HIGH).
MBISTDOUT[ETB_DATA_WIDTH-1:0]	**CLK**	Output	Read data bus for external BIST controller (active when **MTESTON** is HIGH).

Table A-1 Signal descriptions (continued)

Signal Name	Clock domain	Type	Description
MBISTWE	**CLK**	Input	Active HIGH write enable for external BIST controller (active when **MTESTON** is HIGH).
MTESTON	**CLK**	Input	Enable signal for external BIST controller.
nDBGTRST	**DBGTCK**	Input	Active LOW test reset.
nDBGTDOEN	DBGTCK	Output	Enable for TDO. When LOW, this signal denotes that serial data is being driven out on the **DBGTDO** output. **nDBGTDOEN** is normally used as an output enable for a **DBGTDO** pin in a packaged part.
nRESET	**CLK**	Input	Active LOW ETB11 reset.
PORTSIZE[2:0]	**CLK**	Input	Indicates currently selected port size in use on the **TRACEPKT[15:0]** bus.
PROTOCOL[1:0]	**CLK**	Input	Indicates the currently selected ETM protocol.
SBYPASS	**CLK**	Input	Indicates that **HCLK** and **CLK** are synchronous, so the synchronizing logic can be bypassed.
SCANMODE	**CLK**	Input	Selects scan mode.
SE	CLK	Input	Scan enable.
SWEN	-	Input	When LOW disables software access to the ETB11.
TRACEOUTPUT [ETB_DATA_WIDTH-1:0]	**CLK**	Input	Trace information from the ETM11RV.
TRACEVALID	**CLK**	Input	Indicates that the current trace information on **TRACEOUTPUT** is valid.
TRIGGER	**CLK**	Input	Indicates that an ETM11RV trigger has occurred.

a. Can be left unconnected during normal operation.

Appendix B
Integrating the ETB11

This section describes how to integrate the ETB11 if you are not using the ETK11 Integration Kit. It contains the following sections:

B.1 ASIC connections

Table B-1 lists some ETB11 signals and how to use and connect them.

Table B-1 ETB11 connection guide

Signal	Connection information
SWEN	If the AHB interface is to be used to access the ETB11 registers and the ETB11 RAM, then this must be tied HIGH. Otherwise it must be tied LOW.
ACQCOMP	This is a status signal from the ETB11 that can be used to control on-chip logic. For example, **ACQCOMP** can be used to generate an interrupt request to the ARM processor in the system to indicate that the ETB11 is finished collecting trace information.
FULL	This is a status signal from the ETB11 that can be used to control on-chip logic. For example, **FULL** can be used to generate an interrupt request to the ARM processor in the system to indicate that the ETB11 RAM is full.
SBYPASS	If **HCLK** and **CLK** are synchronous then this signal must be tied HIGH so that the synchronization logic between the **HCLK** and **CLK** domain is bypassed. Otherwise this must be tied LOW.
HCLK	If **HCLK** and **CLK** are synchronous then this must be tied to **CLK**.
HCLKEN	If **HCLK** and **CLK** are asynchronous then this must be tied HIGH.
HSELMEM	This is an AHB select signal to indicate that an access to the ETB11 registers is being initiated. This must be generated in the ASIC AHB decode and has a separate memory map to the ETB11 RAM.
HSELREG	This is an AHB select signal to indicate that an access to the ETB11 RAM is being initiated. This must be generated in the ASIC AHB decoder and has a separate memory map to the ETB11 registers.

B.2 Connecting to ETM11RV

Use the connection scheme listed in Table B-2 to connect the ETB11 to a generic trace port interface device, such as an ETM11RV.

Table B-2 ETB11 to generic trace port interface connections

ETB11 signal	Connection to ETM11RV
TRACEOUTPUT[31:0]	**TRACEDATA[31:0]**
TRACEVALID	**TRACEVALID**
TRIGGER	**TRIGGER**
PROTOCOL[1:0]	b10
PORTSIZE[2:0]	**PORTSIZE[2:0]**
DBGTCK	CLK
DBGTCKEN	DBGTCKEN

B.3 Connecting the ETB11 in a 64-bit AHB system

The ETB11 AHB interfaces are 32-bits wide, and cannot therefore be directly connected to a 64-bit AHB bus, such as that used in the ARM10 processor systems. A simple bridge can be constructed that replicates the data on both halves of the bus, as described in the *AMBA Specification*. The Verilog code shown in Example B-1 demonstrates how this can be done:

Example B-1 Connecting the ETB11 in a 64-bit system

```
<code>
// Logic to multiplex the 64-bit AHB bus to a 32-bit bus for 32-bit devices
always @(posedge HCLK)
begin
    if(!HRESETn)
        HWDATAMEMSelect <= 1'b0;
    else if(HREADY)
        HWDATAbusSelect <= HADDR[2];
end

assign HWDATAMEM32 = HWDATAMEMSelect ? HWDATAMEM[63:32] : HWDATAMEM[31:0];

// Recreate the 64-bit bus from the 32-bit Trace Buffer HRDATA bus
assign HRDATAMEM = {HRDATAMEM32,HRDATAMEM32};
</code>
```

If the code shown in Example B-1 is used then load/store multiple instructions that access the ETB11 have unpredictable results because these use both halves of the 64-bit bus at the same time. These accesses do not cause an AHB ERROR response, which normally cause a Data Abort, so the error is not seen by the system. To remain compatible with these systems, load/store multiple instructions are not permitted when accessing the ETB11 to retrieve trace information (see *Restrictions on use of the AHB interface* on page 3-12).

This scheme can be used only when the trace RAM is to be used exclusively for tracing. If the trace RAM is to be used as system memory then a full downsizer must be used that bridges between 64-bit and 32-bit AHB buses, and convert a single 64-bit transfer into two 32-bit transfers. This is provided as part of the *AMBA Design Kit* (ADK).

Glossary

This glossary describes some of the terms used in this manual. Where terms can have several meanings, the meaning presented here is intended.

Advanced Microcontroller Bus Architecture (AMBA)
AMBA is the ARM open standard for multi-master on-chip buses, capable of running with multiple masters and slaves. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules. AHB conforms to this standard.

See also Advanced High-performance Bus and AHB-Lite.

AHB *See* Advanced High-performance Bus.

AHB-Lite AHB-Lite is a subset of the full AHB specification. It is intended for use in designs where only a single AHB master is used. This can be a simple single AHB master system or a multi-layer AHB system where there is only one AHB master on a layer.

AMBA *See* Advanced Microcontroller Bus Architecture.

APB *See* Advanced Peripheral Bus.

Byte An eight-bit data item.

Clock gating Gating a clock signal for a macrocell with a control signal, and using the modified clock that results to control the operating state of the macrocell.

Data Abort An indication from a memory system to a core that it must halt execution of an attempted illegal memory access. A Data Abort is attempting to access invalid data memory.

Debugger A debugging system that includes a program, used to detect, locate, and correct software faults, together with custom hardware that supports software debugging.

An application that monitors and controls the operation of a second application. Usually used to find errors in the application program flow

Embedded Trace Macrocell (ETM)
A hardware macrocell which, when connected to a processor core, outputs instruction and data trace information on a trace port.

ETM *See* Embedded Trace Macrocell

Halfword A 16-bit data item.

JTAG *See* Joint Test Action Group

Joint Test Action Group (JTAG)
The name of the organization that developed standard IEEE 1149.1. This standard defines a boundary-scan architecture used for in-circuit testing of integrated circuit devices. It is commonly known by the initials JTAG.

Macrocell A complex logic block with a defined interface and behavior. A typical VLSI system comprises several macrocells (such as an ETM9 and a memory block) plus application-specific logic.

Reserved A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are set as specified. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as zero and will be read as zero.

TAP *See* Test Access Port

Test Access Port (TAP)
The collection of four mandatory and one optional terminals that form the input and output and control interface to a JTAG boundary-scan architecture. The mandatory terminals are **DBGTDI**, **DBGTDO**, **DBGTMS**, and **DBGTCK**. The optional terminal is **nDBGTRST**.

TPA *See* Trace Port Analyzer.

Trace driver A target that controls a piece of trace hardware. That is, the trigger macrocell, trace macrocell and trace capture tool.

Trace hardware A term for a device that contains an ETM.

Trace port analyzer (TPA)
The trace port analyzer is an external hardware device that stores the information from the trace port, for example a logic analyzer or a low-cost collection unit. The debug tools retrieve data from the analyzer, reconstruct an historical view of the processor's activity including data accesses, as well as configuring the macrocell using the JTAG port. Powerful user-definable filters enable you to limit the amount of information captured in search of a bug, reducing upload time from the trace port analyzer.

Unpredictable For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.

Word A 32-bit data item. Words are normally word-aligned in ARM systems.

Index

J

N

P

R

S

T

W

