AHB Example AMBA SYstem

Technical Reference Manual

ARM®

Copyright © 1999 ARM Limited. All rights reserved.
DDI0170A

INTEL EXHIBIT 1017 Page 1
IPR2023-00787

AHB Example AMBA SYstem
Technical Reference Manual

Copyright © 1999 ARM Limited. All rights reserved.

Release Information

Change history

Date	Issue	Change
August 1999	A	First release

Proprietary Notice

ARM, the ARM Powered logo, Thumb and StrongARM are registered trademarks of ARM Limited.

The ARM logo, AMBA, PrimeCell, Angel, ARMulator, EmbeddedICE, ModelGen, Multi-ICE, ARM7TDMI, ARM7TDMI-S, ARM9TDMI, TDMI and STRONG are trademarks of ARM Limited.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM Limited in good faith. However, all warranties implied or expressed, including but not limited to implied warranties or merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

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`

INTEL EXHIBIT 1017 Page 2
IPR2023-00787

Contents
AHB Example AMBA SYstem Technical Reference Manual

Preface
About this document vi
Further reading viii
Feedback ix

Chapter 1 Introduction
1.1 Overview of EASY 1-2

Chapter 2 The EASY Microcontroller
2.1 Functional overview 2-2
2.2 The AMBA system components 2-3
2.3 Reference peripherals 2-5
2.4 Example components 2-8
2.5 System test methodology 2-9

Chapter 3 ARM7TDMI AHB Wrapper
3.1 About the ARM7TDMI AHB wrapper 3-2
3.2 Signal interface 3-3
3.3 ARM7TDMI AHB signal descriptions 3-4
3.4 Overview of the ARM7TDMI wrapper 3-7

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 3
IPR2023-00787

3.5 Connections to ARM7TDMI core 3-9
3.6 Default signal configurations 3-13
3.7 Description of the ARM7TDMI wrapper blocks 3-14

Chapter 4 AHB Modules
4.1 APB bridge 4-2
4.2 Arbiter 4-14
4.3 Decoder 4-25
4.4 Default slave 4-29
4.5 Master to slave multiplexor 4-32
4.6 Slave to master multiplexor 4-36
4.7 Reset controller 4-40
4.8 Retry slave 4-46
4.9 Static memory interface 4-53
4.10 Test interface controller 4-64

Chapter 5 APB Modules
5.1 Interrupt controller 5-2
5.2 Remap and pause controller 5-12
5.3 Timers 5-20
5.4 Peripheral to bridge multiplexor 5-35

Chapter 6 Behavioral Modules
6.1 External RAM 6-2
6.2 External ROM 6-5
6.3 Internal RAM 6-8
6.4 Test interface driver 6-12
6.5 Tube 6-24

Chapter 7 Designer's Guide
7.1 Adding bus masters 7-2
7.2 Adding AHB slaves 7-3
7.3 Adding APB peripherals 7-4

 Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 4
IPR2023-00787

Preface

This preface introduces the AHB *Example AMBA SYstem* (EASY) and its reference documentation. It contains the following sections:

- *About this document* on page vi
- *Further reading* on page viii
- *Feedback* on page ix.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 5
IPR2023-00787

About this document

This document is a comprehensive manual for the behavioral HDL model of the *Example AMBA SYstem* (EASY). It gives detailed information about:

- the function of the whole system
- each module in the system
- how to design a new system module.

This document refers to the *Advanced High-performance Bus* (AHB). For information on the *Advanced System Bus* (ASB) refer to the *ASB Example AMBA SYstem Technical Reference Manual*.

Intended audience

This document has been written for experienced hardware and software engineers who wish to incorporate a fully functional AMBA system into their hardware and software design.

Organization

This document is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an overview of the AHB Example AMBA SYstem.

Chapter 2 ***The EASY Microcontroller***

Read this chapter for a description of the modules of the AHB EASY microcontroller.

Chapter 3 ***ARM7TDMI AHB Wrapper***

Read this chapter for a description of the ARM7TDMI AHB wrapper module.

Chapter 4 ***AHB Modules***

Read this chapter for details of the AHB modules that are used in the AHB Example AMBA SYstem.

Chapter 5 ***APB Modules***

Read this chapter for details of the APB modules that are used in the AHB Example AMBA SYstem.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 6
IPR2023-00787

Chapter 6 *Behavioral Modules*

Read this chapter for details of how to use the behavioral modules, including memory modules and the external AMBA Test Interface Driver module (the *TICBOX*). This chapter contains a description of the *TICTalk* command language.

Chapter 7 *Designer's Guide*

Read this chapter for details of how to add new bus master, slave and peripheral modules to the AHB EASY microcontroller.

Typographical conventions

The following typographical conventions are used in this document:

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

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 7
IPR2023-00787

Chapter 1
Introduction

This chapter introduces the *AHB Example AMBA SYstem* (EASY). It contains the following section:

- *Overview of EASY* on page 1-2.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 8
IPR2023-00787

1.1 Overview of EASY

The EASY microcontroller comprises the building blocks needed to create an example system based on the low-power, generic design methodology of the *Advanced Microcontroller Bus Architecture* (AMBA).

The EASY microcontroller:

- enables custom devices to be developed in very short design cycles
- allows the resulting subcomponents to be easily reused in future designs.

Note

This document refers to the *Advanced High-performance Bus* (AHB). For information on the *Advanced System Bus* (ASB) refer to the *ASB Example AMBA SYstem Technical Reference Manual*.

1.1.1 EASY system blocks

The example design provides all the system modules needed to manage an AMBA system:

- reset controller
- arbiter
- decoder.

These system modules control various aspects of the *Advanced High Performance Bus* (AHB).

1.1.2 EASY components

The example design comprises:

- Two buses:
  - the AHB
  - the *Advanced Peripheral Bus* (APB).
- The ARM processor AHB wrapper, to allow execution of ARM code in an AHB system.
- The *Test Interface Controller* (TIC), to allow external control of the AHB during system test.
- A minimum set of basic microcontroller peripherals. These are supported, and are implemented as low-power designs on the APB. They include:
  - an interrupt controller

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 9
IPR2023-00787

— a remap and pause controller

— a 16-bit timer module.

- The example *Static Memory Interface* (SMI). This demonstrates the minimum requirements for an *External Bus Interface* (EBI).
- A 1KB block of internal memory.

The EASY system consists of a microcontroller with some external memory as shown in Figure 1-1.

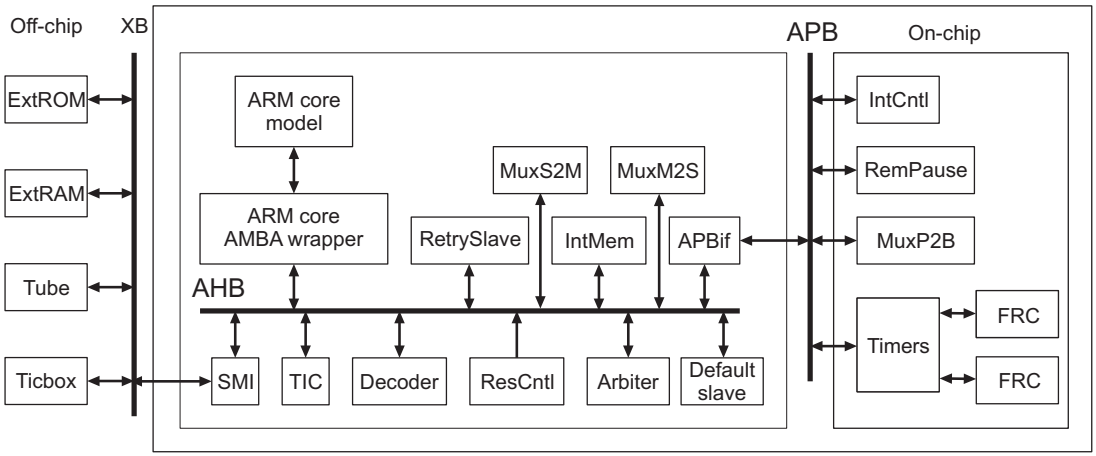

Figure 1-1 EASY system diagram

The descriptions in this manual refer to an AHB-based EASY system. For details of ASB-based EASY system design refer to the *Example AMBA SYstem Technical Reference Manual*.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 10
IPR2023-00787

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 11
IPR2023-00787

Chapter 2
The EASY Microcontroller

This chapter describes the microcontroller which is the main unit of the EASY system. It contains the following sections:

- *Functional overview* on page 2-2
- *The AMBA system components* on page 2-3
- *Reference peripherals* on page 2-5
- *Example components* on page 2-8
- *System test methodology* on page 2-9.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 12
IPR2023-00787

2.1 Functional overview

The modules of the EASY microcontroller are grouped in five classes:

AMBA system components

Used to control the general operation of the system.

Peripherals

Low-power peripherals, which are connected to the peripheral bus.

Example components

Demonstration modules that are only simulation models.

System test methodology

Modules used for testing the system.

Processor core

The ARM processor core that is built into the EASY microcontroller.

With the exception of the processor core the above modules are fully described in this chapter. For details of the processor core refer to Chapter 3 *ARM7TDMI AHB Wrapper*.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 13
IPR2023-00787

2.2 The AMBA system components

The *Advanced Microcontroller Bus Architecture* (AMBA) system comprises:

- *Reset controller*
- *Arbiter*
- *Decoder*
- *AHB to APB bridge* on page 2-4.

2.2.1 Reset controller

The reset controller consists of a state machine which generates the **HRESETn** signal. This signal indicates the current reset state of the AMBA bus and is used by all the other elements in the EASY microcontroller, primarily for power-on initialization.

Note

All other reset modes, such as standby or warm reset, must be implemented separately.

2.2.2 Arbiter

The arbiter provides arbitration between bus masters competing for access to the AHB. Although there are only two bus masters in the EASY microcontroller, the ARM and the TIC, the arbiter has provision for up to four masters. To extend the number of masters, refer to Chapter 7 *Designer's Guide*. The arbitration is currently assigned with a simple priority system, with the TIC as the highest priority, and the processor as the lowest (also the reset default). The arbitration scheme is not defined in the *AMBA Specification* and can be dependent on implementation.

2.2.3 Decoder

The decoder consists of a simple address decoding logic, which is used to select the system bus slaves based on the address of the current transfer. This module controls the configurable memory map for the system.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 14
IPR2023-00787

Chapter 4
AHB Modules

This chapter describes the data sheets for the modules that are connected to the *Advanced High Performance Bus* (AHB). It contains the following sections:

- *APB bridge* on page 4-2
- *Arbiter* on page 4-14
- *Decoder* on page 4-25
- *Default slave* on page 4-29
- *Master to slave multiplexor* on page 4-32
- *Slave to master multiplexor* on page 4-36
- *Reset controller* on page 4-40
- *Retry slave* on page 4-46
- *Static memory interface* on page 4-53
- *Test interface controller* on page 4-64.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 15
IPR2023-00787

4.1 APB bridge

The AHB to APB bridge is an AHB slave, providing an interface between the high-speed AHB and the low-power APB. Read and write transfers on the AHB are converted into equivalent transfers on the APB. As the APB is not pipelined, then wait states are added during transfers to and from the APB when the AHB is required to wait for the APB. Figure 4-1 shows the block diagram of the APB bridge module.

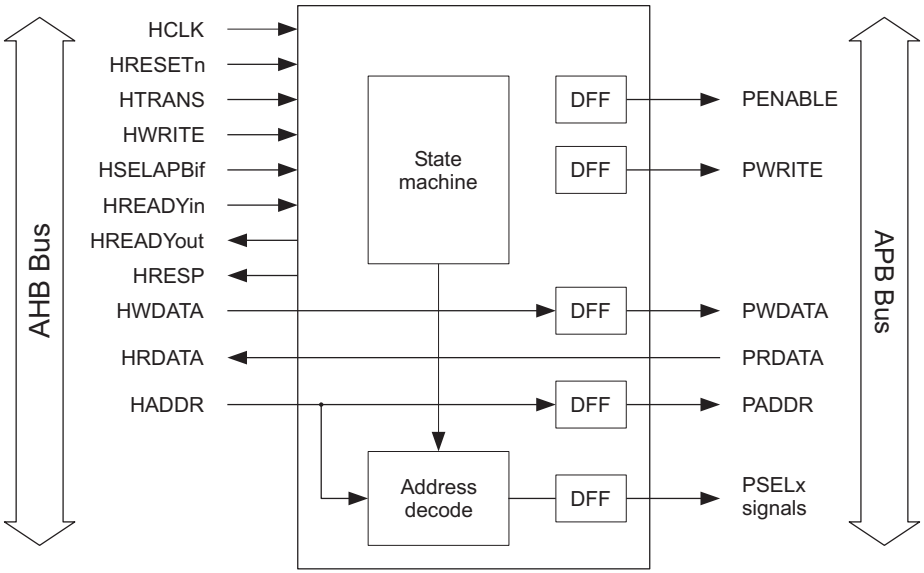

Figure 4-1 Block diagram of bridge module

The main sections of this module are:

- AHB slave bus interface
- APB transfer state machine, which is independent of the device memory map
- APB output signal generation.

To add new APB peripherals, or alter the system memory map, only the address decode sections need to be modified.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 16
IPR2023-00787

4.1.1 Signal descriptions

The APB bridge module signals are described in Table 4-1.

Table 4-1 Signal descriptions for bridge module

Signal	Type	Direction	Description
HCLK	Bus clock	Input	This clock times all bus transfers.
HRESETn	Reset	Input	The bus reset signal is active LOW, and is used to reset the system and the bus.
HADDR[31:0]	Address bus	Input	The 32-bit system address bus.
HTRANS[1:0]	Transfer type	Input	This indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE	Transfer direction	Input	When HIGH this signal indicates a write transfer, and when LOW, a read transfer.
HWDATA[31:0]	Write data bus	Input	The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HSELAPBif	Slave select	Input	Each APB slave has its own slave select signal, and this signal indicates that the current transfer is intended for the selected slave. This signal is a combinatorial decode of the address bus.
HRDATA[31:0]	Read data bus	Output	The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HREADYin **HREADYout**	Transfer done	Input/output	When HIGH the **HREADY** signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer.
HRESP[1:0]	Transfer response	Output	The transfer response provides additional information on the status of a transfer. This module will always generate the OKAY response.
PRDATA[31:0]	Peripheral read data bus	Input	The peripheral read data bus is driven by the selected peripheral bus slave during read cycles (when **PWRITE** is LOW).

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 17
IPR2023-00787

4.7 Reset controller

The reset controller is used to generate the system reset signal from an external reset input as shown in Figure 4-23.

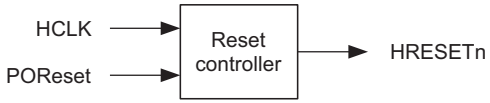

Figure 4-23 Reset controller module interface diagram

This module is based around a state machine, which is used to detect the external reset being asserted, and is used to generate the system reset output.

4.7.1 Signal descriptions

Table 4-7 shows the signal descriptions for the reset controller.

Table 4-7 Reset controller signal descriptions

Signal	Type	Direction	Description
HCLK	Bus clock	Input	This clock times all bus transfers.
POReset	Power-on reset	Input	Power-on reset input. This active LOW signal causes a cold reset when LOW. May be asserted asynchronously to **HCLK**.
HRESETn	Reset	Output	The bus reset signal is active LOW, and is used to reset the system and the bus.

The source of the **POReset** signal is implementation-dependent.

4.7.2 Function and operation of module

HRESETn is asserted LOW, and is used to indicate a reset condition where all bus and system states should be initialized. This signal is suitable as an asynchronous clear into state machine flip-flops, and for resetting any peripheral registers that require initialization.

During reset, the arbiter grants the bus to the default reset bus master, and the decoder selects the default slave.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 18
IPR2023-00787

Assertion (the falling edge) of **HRESETn** is asynchronous to **HCLK**. De-assertion (the rising edge) of **HRESETn** is synchronous to the rising edge of **HCLK**. **HRESETn** is only asserted during a power-on reset condition, caused by the assertion of the **POReset** signal. The **POReset** input is an asynchronous input, so a synchronizing register is required to eliminate propagation of metastable values. Figure 4-24 shows the operation of the **HRESETn** signal with respect to an example **POReset** input signal and the system clock.

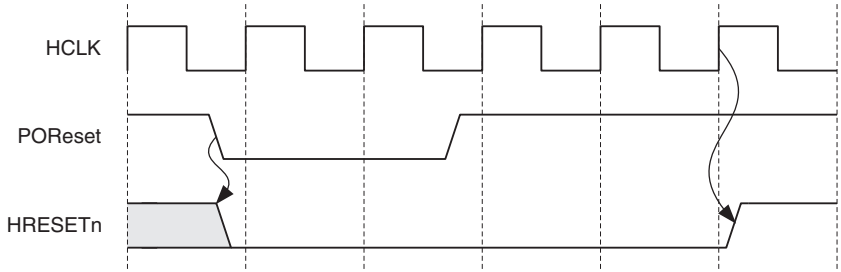

Figure 4-24 Reset signal timing

The reset controller contains a state machine running from the rising edge of **HCLK**. The **HRESETn** signal directly reflects a single bit of the current state, minimizing the combinational logic applied to the reset output.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 19
IPR2023-00787

Figure 4-25 shows the state machine for the reset controller.

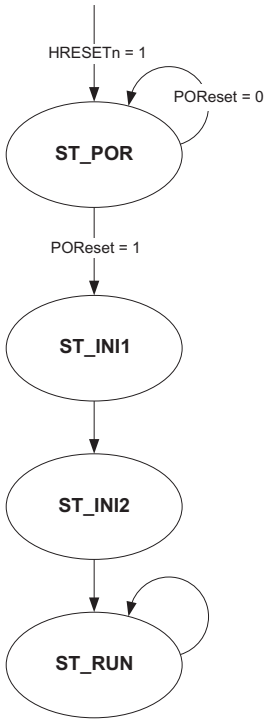

Figure 4-25 State machine for reset controller

The four states are described in:

- *ST_POR* on page 4-43
- *ST_INI1* on page 4-43
- *ST_INI2* on page 4-43
- *ST_RUN* on page 4-43.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 20
IPR2023-00787

ST_POR

During this state, the system is initialized when the reset line is asserted. This state should be preserved by a power on reset cell or controller, until the system bus clock is running and stable, and the system power supply has reached its correct operating voltage (within its allowed limits).

The ST_POR state is entered from:

- reset, when the external reset input is first asserted LOW
- ST_POR when the external reset input is still asserted and the system clock is running.

The next state is:

- ST_INI1 when the external reset input is de-asserted
- ST_POR when the external reset input is still asserted and the system clock is running.

If there is a clock valid signal in the system, this should be used to prevent the ST_POR state from being exited until the clock is valid.

ST_INI1

This state is used to hold the **HRESETn** output LOW for an extra cycle after the external reset is de-asserted.

This state is always entered from ST_POR on the first rising edge of the clock that the external reset is HIGH.

The next state is always ST_INI2.

ST_INI2

This state is used in the same manner as ST_INI1.

This state is always entered from ST_INI1.

The next state is always ST_RUN.

ST_RUN

This state is used during normal system operation when the **HRESETn** output is set HIGH.

This state is held until the external reset is re-asserted.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 21
IPR2023-00787

The default reset controller implementation asserts **HRESETn** for two cycles after the external reset is de-asserted, but this may be altered by adding extra ST_INI states to the state machine, so that it takes more cycles to reach the final ST_RUN state.

4.7.3 System description

The following paragraphs give a description of how the HDL code for the reset controller is set out. A simple system block diagram, with information about the main parts of the HDL code, is followed by details of the registers, inputs, and outputs used in the module. This part should be read together with the HDL code.

Figure 4-26 shows the reset controller module block diagram.

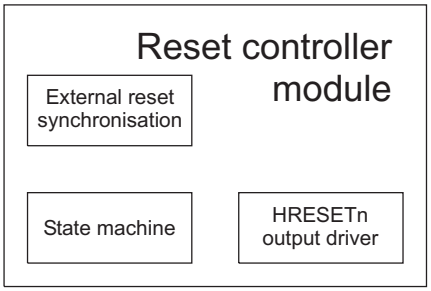

Figure 4-26 Reset controller module block diagram

The reset controller is comprised of a register used to synchronize the external reset input, and a state machine used to control the generation of the system reset output.

All registers used in the system are clocked from the rising edge of the system clock **HCLK**.

Figure 4-27 shows the reset controller HDL file.

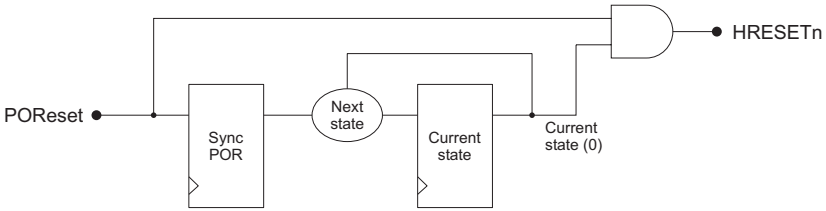

Figure 4-27 Reset controller module system diagram

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 22
IPR2023-00787

The main sections in this module are explained in the following paragraphs:

- *Asynchronous reset input synchronization*
- *Reset state machine*
- *Reset output generation*.

Asynchronous reset input synchronization

The asynchronous external reset is first passed through a rising-edge-triggered register. This is to avoid metastability, due to the arrival time of the input relative to the system clock when used in the state machine.

Reset state machine

The state machine shown in Figure 4-25 on page 4-42 is used to control the generation of the system reset output, based on the status of the synchronized external reset input and the system clock.

The number of cycles the module holds **HRESETn** asserted after the de-assertion of the external reset may be changed by altering the number of initialization states between the first and last states.

Reset output generation

The reset output is generated directly from bit 0 of the state machine registers, gated with the external reset input. This allows asynchronous assertion of the reset output when the external reset input is set LOW and the system clock is not running, but ensures that de-assertion is synchronous to the rising edge of the clock.

Copyright © 1999 ARM Limited. All rights reserved.

INTEL EXHIBIT 1017 Page 23
IPR2023-00787