PrimeCell™ Static Memory Controller (PL092)

Revision: r1p3

Technical Reference Manual

ARM®

ARM DDI 0203F

PrimeCell Static Memory Controller (PL092)
Technical Reference Manual

Release Information

Change history

Date	Issue	Change
April 2001	A	First release
June 2001	B	Signal changes to F1-3, Page 2-58, A-7, A-8
July 2002	C	Incorporation of errata
January 2003	D	Incorporation of errata
June 2003	E	Additional signal incorporated, revision r1p3
December 2003	F	Editorial update, register diagrams added, errata sheet incorporated, new Hardware Preface included, index updated

Proprietary Notice

Confidentiality Status

Product Status

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

Web Address

http://www.arm.com

Contents
PrimeCell Static Memory Controller (PL092) Technical Reference Manual

List of Tables
PrimeCell Static Memory Controller (PL092) Technical Reference Manual

List of Figures
PrimeCell Static Memory Controller (PL092) Technical Reference Manual

Preface

This preface introduces the *ARM PrimeCell Static Memory Controller (PL092) Revision r1p3 Technical Reference Manual*. It contains the following sections:

About this document

This is the *Technical Reference Manual* (TRM) for the ARM PrimeCell *Static Memory Controller* (SMC).

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 manual is written for implementation engineers and architects, and provides a description of an optimal PrimeCell SMC architecture. The PrimeCell SMC provides an interface between the *Advanced High-performance Bus* (AHB) system bus and external (off-chip) memory devices.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the PrimeCell SMC and its features.

Chapter 2 *Functional Overview*

Read this chapter for an overview of the major functional blocks and the operation of the PrimeCell SMC.

Chapter 3 *Programmer's Model*

Read this chapter for a description of the registers and for details of system initialization.

Chapter 4 *Programmer's Model for Test*

Read this chapter for a description of the additional logic for functional verification and production testing.

Appendix A *Signal Descriptions*

Read this appendix for a description of the PrimeCell SMC signals.

Glossary Read the Glossary for definitions of terms used in this manual.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams* on page xiv
- *Signals* on page xiv
- *Numbering* on page xv.

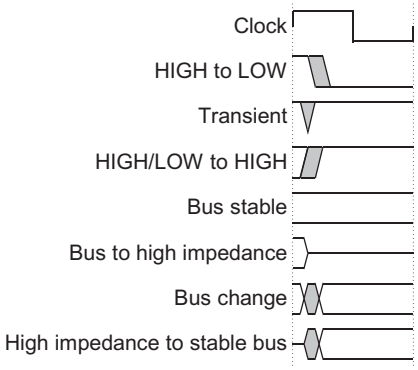
Typographical

The typographical conventions are:

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

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



Key to timing diagram conventions

Signals

The signal conventions are:

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

Prefix R Denotes AXI read channel signals.

Prefix W Denotes AXI write channel signals.

Numbering

The numbering convention is:

<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, and by third parties.

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

ARM publications

This manual contains information that is specific to the ARM PrimeCell SMC. Refer to the following document for other relevant information:

- *AMBA Specification* (ARM IHI 0011).

Feedback

ARM Limited welcomes feedback on the ARM PrimeCell SMC and its documentation.

Feedback on the ARM PrimeCell SMC

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 the manual, send email to `errata@arm.com` giving:

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

ARM Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the SMC. It contains the following sections:

1.1 About the SMC

The PrimeCell *Static Memory Controller* (SMC) is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM Limited.

The SMC is an AMBA slave module, and connects to the *Advanced High-performance Bus* (AHB). It is a reusable soft-IP block that has been developed with the prime aim of reducing time-to-market for *Application-Specific Integrated Circuit* (ASIC) development. It contains an AHB *Test Interface Controller* (TIC) AMBA master block that can be used to test the system using externally applied TIC vectors.

A carefully chosen set of design rules has been followed to enable faster integration in most ASIC design flows using standard synthesis and test tools. The implementation can easily be customized to suit specific customer requirements. The AMBA TIC is included to reduce the design time required to construct an SMC-based system that enables testing through TIC vectors.

For further information on AMBA see the *AMBA Specification*.

1.1.1 Features of the PrimeCell SMC

The PrimeCell SMC macro block offers the following features:

- soft macrocell available in both VHDL and Verilog
- fully scan-insertable design
- functional verification using ARM BusTalk functional test environment
- compatibility with AMBA AHB on-chip bus systems.

The PrimeCell SMC supports:

- static memory-mapped devices including RAM, ROM, flash, and burst ROM
- asynchronous page mode read operation in nonclocked memory subsystems
- asynchronous burst mode read access from burst mode ROM and flash devices
- 8, 16, and 32-bit wide external memory data paths
- little-endian and big-endian memory architectures
- AHB burst transfers
- independent configuration for up to eight memory banks, each up to 64MB
- programmable wait states (up to 31)
- programmable bus turnaround cycles (up to 15)
- programmable output enable and write enable delays (up to 15)
- write enable and byte lane select outputs for use with 32, 16, or 8-bit SRAM devices
- independent byte lane control for each memory bank

- external asynchronous wait control
- configurable size at reset for boot memory bank using external control pins
- system testing using externally applied TIC vectors through built-in TIC AMBA master block
- access port enables a future device access to external bus.

1.1.2 Programmable parameters

The following key parameters are programmable for each memory bank:

- external memory width, 8, 16, or 32-bit
- burst mode operation
- write protection
- chip select polarity
- external wait control enable
- external wait polarity
- write WAIT states for static RAM devices
- read WAIT states for static RAM and ROM devices
- initial and subsequent burst read WAIT state for burst ROM devices
- read byte lane enable control
- bus turn-around (idle) cycles
- output enable and write enable output delays.

1.2 Example of a typical system

Figure 1-1 shows a diagram of a typical AHB-based microcontroller system, with an optional additional asynchronous memory controller.

The PrimeCell SMC has two AHB connections, the slave interface for the SMC and the master interface for the TIC.

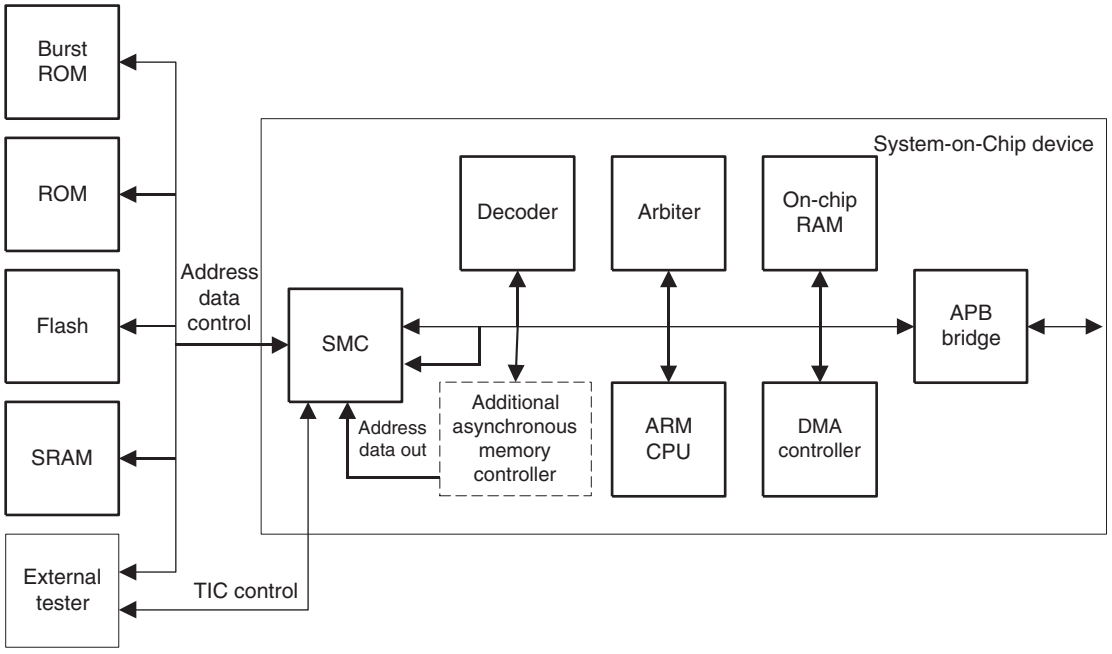


Figure 1-1 Typical AMBA AHB-based microcontroller system

1.3 Input and output connections

Figure 1-2 shows the input and output connections for the PrimeCell SMC.

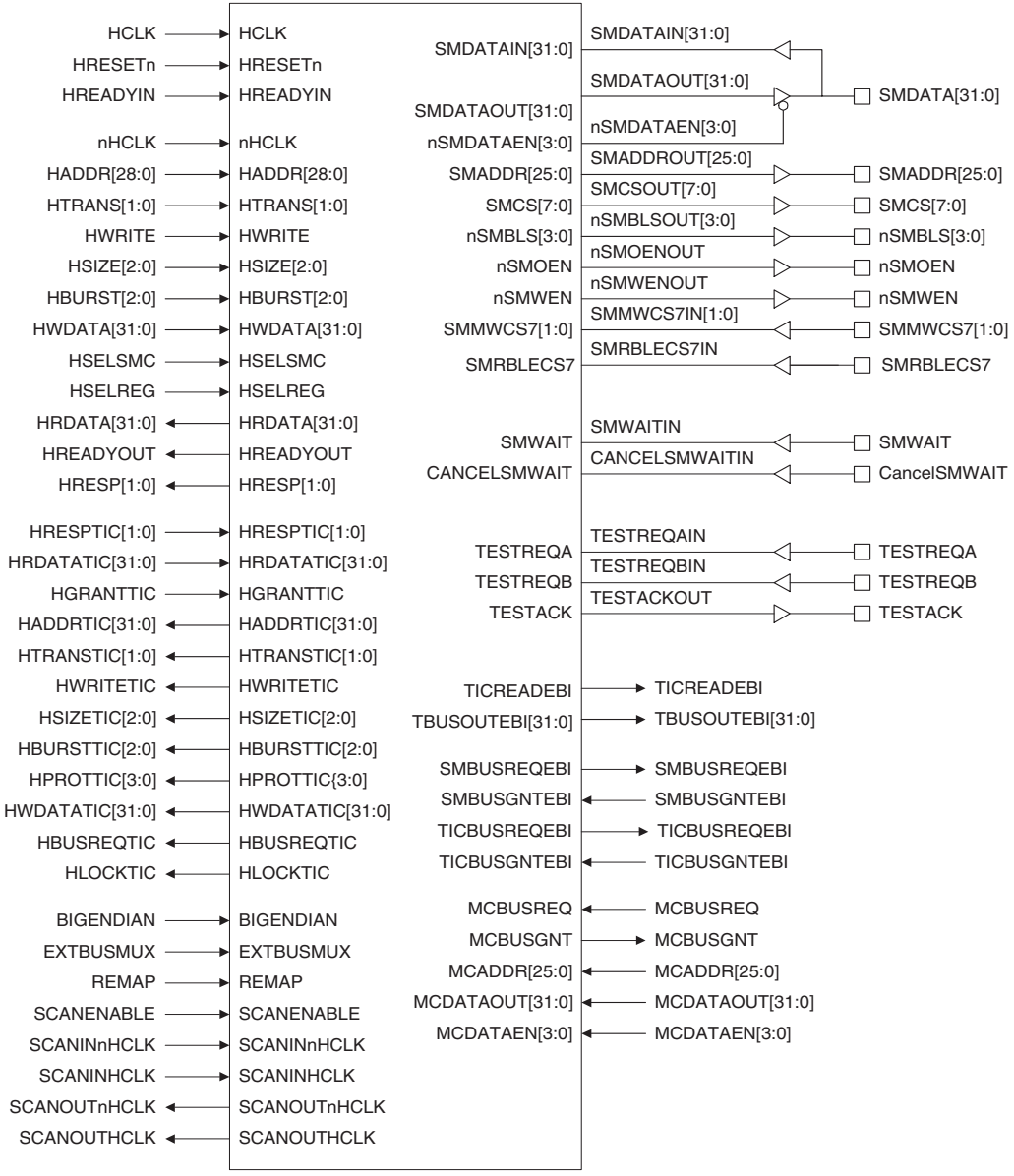


Figure 1-2 PrimeCell SMC input and output connections

1.4 Synchronous AHB memory controller

The PrimeCell SMC supports the connection of a synchronous AHB memory controller, such as an SDRAM controller. A typical system is shown in Figure 1-3.

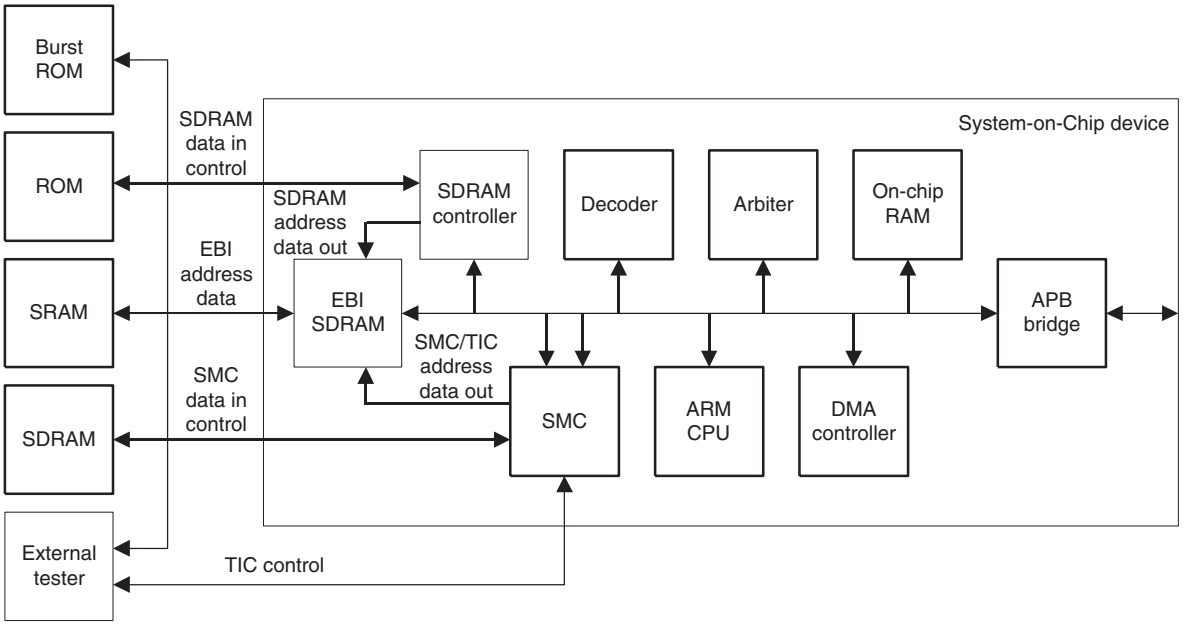


Figure 1-3 AMBA AHB-based microcontroller system with SMC and SDRAM controller

1.5 Additional asynchronous memory controller

Figure 1-4 shows the signal connections of an additional asynchronous memory controller to the PrimeCell SMC.

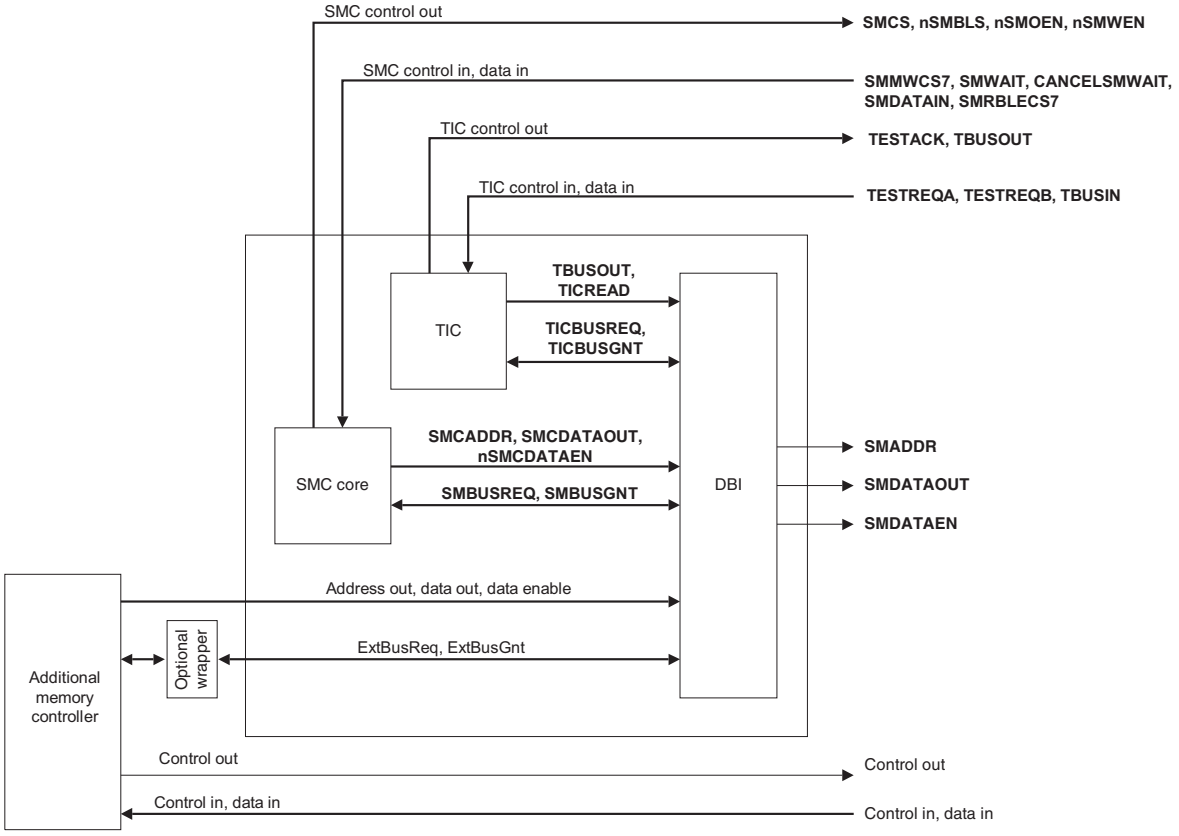


Figure 1-4 Signal connections for additional asynchronous memory controller

Depending on the bus request and grant protocol supported by the additional memory controller, a simple wrapper might be required to enable it to operate with the protocol implemented in the PrimeCell SMC.

1.6 Product revisions

This section describes differences in functionality between product revisions of the PrimeCell SSMC:

r0p0-r0p1 Contains the following differences in functionality:

- Two consecutive writes of differing HSIZE now give the correct bytelane
- Improved functionality for burst write, then half-word write, then burst read operations
- Corrected field width definition in register bwPL093_WSTBRD
- Improved functionality for write after write to a write-protected area.

1.6.1 Section 2.8.1 SMWAIT assertion timing

In the wait enabled or external wait control mode, when the SMC is waiting for the **SMWAIT** assertion, it also starts counting down according to the values programmed in the wait state count field WST1 or WST2, that are used for read and write transfers respectively. You can use this feature to ensure that adequate time is available to the SMC to detect **SMWAIT** as there might be a delay before the external device asserts **SMWAIT**. If **SMWAIT** is not asserted during this time, the transfer is assumed to be zero wait.

Note

When you use the **SMWAIT** input to time memory transfers, the WST1 and WST2 timing registers are used to program the external wait assertion delay. You must set these registers to a minimum of `0x03` instead of the default of `0x00` for standard memory transfers:

- One cycle is required for the minimum chip select to external wait assertion delay
- Two cycles are required for the double synchronization of the **SMWAIT** input before use.

Chapter 2
Functional Overview

This chapter describes the ARM PrimeCell Static Memory Controller (PL092) operation. It contains the following sections:

2.1 ARM PrimeCell SMC overview

The SMC is an AMBA AHB slave module that provides an interface between an AMBA AHB system bus and external (off-chip) memory devices.

The PrimeCell SMC provides support for up to eight independently configurable memory banks simultaneously. Each memory bank is capable of supporting:

- SRAM
- ROM
- flash EPROM
- burst ROM memory.

You can configure each memory bank to use either 8, 16, or 32-bit external memory data paths. The PrimeCell SMC can be configured to support either little-endian or big-endian operation.

The PrimeCell SMC memory banks can be configured to support:

- nonburst read and write accesses to high-speed CMOS Static RAM, for example Samsung KM681002A and K6R1016C1C, and Intel 28F800C3
- nonburst write accesses, nonburst read accesses, and asynchronous page mode read accesses to fast-boot block flash memory, for example Intel 28F800F3 and 28F128J3A
- asynchronous page mode read accesses to ROM, for example Samsung K3P5V(U)2000D-SC and K3P6C2000B-SC.

Support is provided for connecting an additional AHB asynchronous memory controller to external memory, to pass the address, data and data enable lines to the external bus. A simple request and grant protocol is used to control the current driver of the external bus. Use of a synchronous memory controller is also supported, but this requires the use of a suitable external bus multiplexor. See *Using the SMC with an external bus multiplexor or SDRAM controller* on page 2-57 for more details.

Figure 2-1 on page 2-3 shows a block diagram of the PrimeCell SMC.

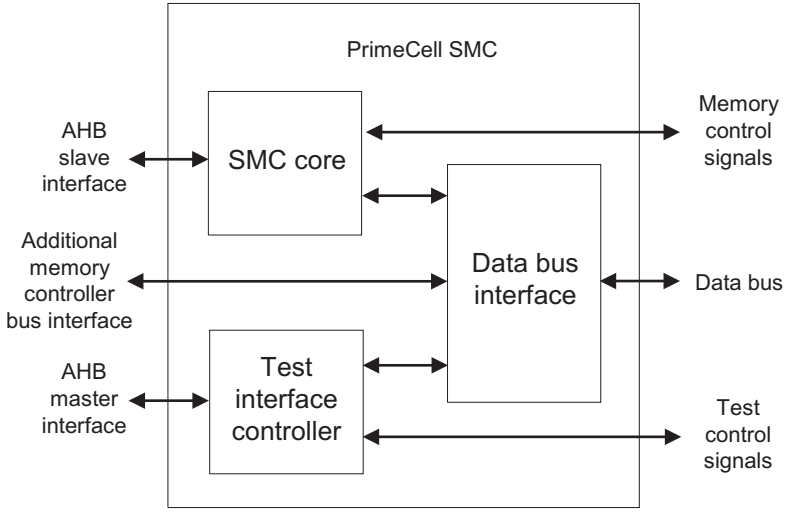


Figure 2-1 PrimeCell SMC block diagram

The three main sub-blocks in the PrimeCell SMC design are:

SMC core Performs read and write accesses to external memory through the AMBA AHB slave interface.

Test interface controller

The TIC is used during testing to read external test vectors and apply them to the system through the AMBA AHB master interface.

Data bus interface

The data bus interface selects either the memory controller, the TIC, or the additional memory controller as the current user of the external data bus.

2.2 SMC core

The SMC core performs read and write accesses to external memory through the AMBA AHB slave interface. Figure 2-2 shows a block diagram of the SMC core.

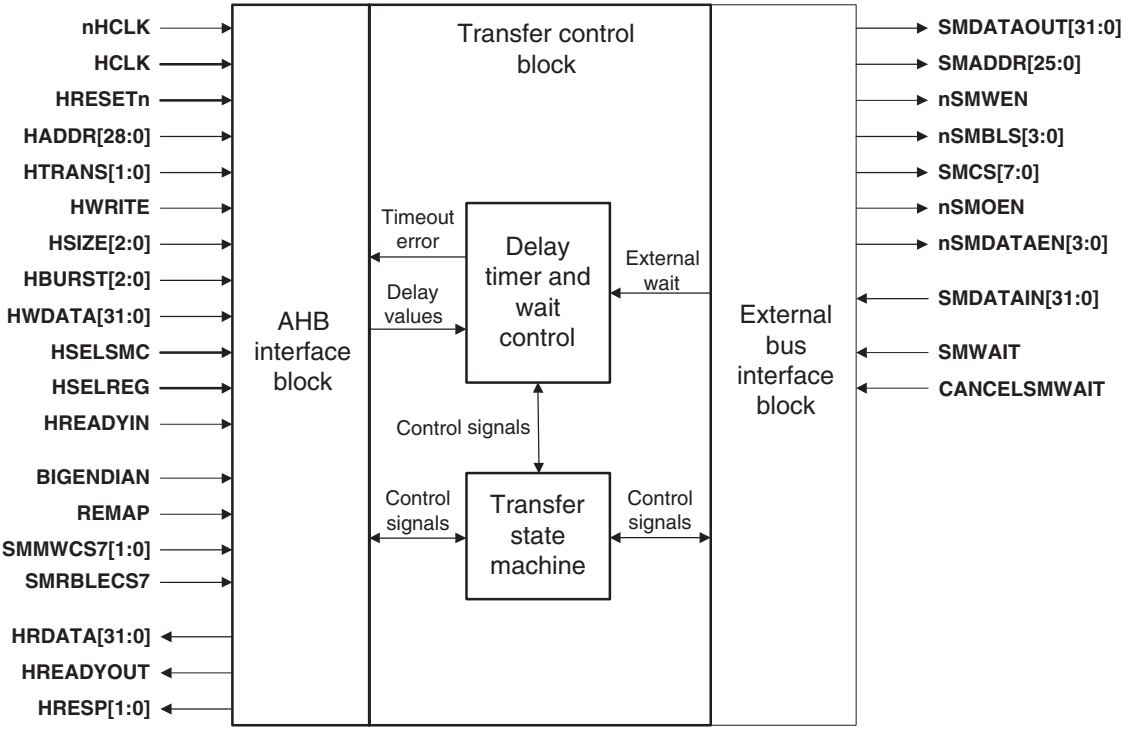


Figure 2-2 SMC core block diagram

The three main blocks of the SMC core are:

- *AMBA AHB interface*
- *Transfer control* on page 2-5
- *External bus interface* on page 2-5.

2.2.1 AMBA AHB interface

The AMBA AHB interface block provides the interface to the AMBA AHB bus. It contains all of the Control (SMBCRx) and Status (SMBSRx) registers for the external memory banks, and the PrimeCell and Peripheral Identification Registers.

The AHB control signals are sampled and passed to the transfer control block, and the control signals received are used to generate the AHB response and read data outputs.

The read/write SMBCRx register values are accessed through the AHB, and their values are passed to the rest of the peripheral. The read-only SMBSRx register values are generated from control signals within the transfer control block.

2.2.2 Transfer control

The transfer control block contains two main sub-blocks:

Transfer state machine

This block is used to control the transfer currently being performed.

Delay timer and wait control

This block is used to control the operation of reads, writes and external waits when a delay is required, and to control the externally waited transfers.

2.2.3 External bus interface

The external bus interface block provides the interface to the external bus. It contains the logic to generate the external control signals, and the 32-bit read/write buffer used to store the data value during transfers. It is used to:

- drive the chip selects for the memory banks
- drive the write enables for the memory banks
- drive the output enable for the memory banks
- drive the external databus enable
- generate read enables for read data from the memory to AMBA AHB interface
- control the data sequencing for memory read and writes for different sizes of AHB transfer and memory width.

2.3 Memory bank selection

Eight independently configurable memory banks are supported, with a separate chip select output for each bank. The chip select lines **SMCS[7:0]** for all banks are configurable to be either active HIGH or active LOW (default). Table 2-1 shows how memory bank selection is controlled by the AMBA AHB address lines **HADDR[28:26]**. All **SMCS** lines are shown as active HIGH.

Table 2-1 Static memory bank select coding

HADDR [28:26]	SMCS[7:0]	Memory bank
000	00000001	Bank 0
001	00000010	Bank 1
010	00000100	Bank 2
011	00001000	Bank 3
100	00010000	Bank 4
101	00100000	Bank 5
110	01000000	Bank 6
111	10000000	Bank 7

The base addresses of the external memory banks and the PrimeCell SMC memory bank registers are defined in the AMBA AHB address decoder. This generates the AHB slave select signals **HSELSMC** and **HSELREG**. If the default base address of the external memory banks begins at `0x00000000`, the memory banks occupy address space up to `0x1FFFFFFF` (8x64MB).

Table 2-2 shows the address mapping of **HADDR[31:0]** for external memory banks and for memory bank Configuration Registers.

Table 2-2 HADDR[31:0] address mapping

HADDR[31:0]	[31:29] [a]	[28:26]	[25:12]	[11:2]	[1:0]
External memory banks	Base address for PrimeCell SMC memory	Chip select address space for eight memory banks	64MB memory banks address space		
Memory bank Configuration Registers	Base address for PrimeCell AHB SMC registers	Unused	Unused	Memory bank register select	Unused

a. **HADDR[31:29]** are not inputs to the SMC. Decoding is performed in the AMBA address decoder.

2.4 Memory bank configuration

This section describes memory bank configuration. It contains the following information:

- *Access sequencing and memory width*
- *Wait state generation*
- *Write protection* on page 2-8.

2.4.1 Access sequencing and memory width

The data width of each external memory bank must be configured by programming the appropriate SMBCRx register. When the external memory bus is narrower than the transfer initiated from the current AMBA bus master, the internal bus transfer takes several external bus transfers to complete. For example, if Bank 0 is configured as 8-bit wide memory and a 32-bit read is initiated, the AMBA AHB bus stalls while the PrimeCell SMC reads four consecutive bytes from the memory. During these accesses the data path is controlled (in the external memory data path logic) to demultiplex the four bytes into one 32-bit word on the AMBA AHB bus.

The access sequencing supports both little-endian and big-endian operation as defined by the dedicated **BIGENDIAN** input signal to the PrimeCell SMC.

2.4.2 Wait state generation

Each bank of the PrimeCell SMC must be configured for external transfer wait states in read and write accesses. This is achieved by programming the appropriate fields of the SMBIDCYRx, SMBWST1Rx, and SMBWST2Rx registers.

The number of cycles in which an AMBA transfer completes is controlled by three other factors:

- access width
- external memory width
- external wait input.

Each bank of the PrimeCell AHB SMC has a programmable enable for the external wait (WaitEn), and a programmable polarity setting (WaitPol), enabling full configuration of the external wait for each bank.

You can program the WST1 wait state field to select up to 31 wait states for read memory accesses to SRAM and ROM, or the initial burst read access to burst ROM.

You can program the WST2 wait state field to select up to 31 wait states for write access to SRAM or burst mode reads from burst ROM devices. For example, the configuration for an access to a burst ROM with a 120ns initial access time followed by a 60ns burst access time, using a 100MHz system clock would be 12 wait states for the first access and 6 for the subsequent accesses.

2.4.3 Write protection

You can configure each memory bank for write protection. Normally SRAM is unprotected and ROM devices must be write protected, but you can set the WP field in the SMCBCRx register to write protect SRAM in addition to ROM devices.

If a write access is made to a write protected memory bank, an error is indicated by the **HRESP[1:0]** signals and the WriteProtErr bit of the SMBSRx register is asserted. If a write access is made to a memory bank containing ROM devices and the bank is not write protected, there is no error indication returned.

2.5 Static memory read control

The static memory read controls are described in the following sections:

- *Output enable programmable delay*
- *ROM, SRAM, and flash*
- *Burst ROM* on page 2-16
- *Burst flash* on page 2-18.

2.5.1 Output enable programmable delay

The delay between the assertion of the chip select and the output enable is programmable from 0-15 cycles using the WSTOEN bits of the SMBWSTOENRx registers. This delay is used to reduce the power consumption for memories that are not able to provide valid output data immediately after the chip select is asserted.

If the output of the device is enabled before the final read data value is ready, the device drives out two different values, one unknown value, followed by the valid read data. This consumes more power than just driving out the final read data value.

The output enable is always deasserted at the same time as the chip select, at the end of the transfer.

Note

The WSTOEN programmed value must be less than or equal to the WST1 value and WSTWEN must be less than or equal to WST2. In the External Wait enabled mode, the timing of the transfer (controlled by **SMWAIT**) is not known, so **SMOEN** is asserted along with **SMCS**.

2.5.2 ROM, SRAM, and flash

PrimeCell SMC uses the same read timing control for ROM, SRAM, and flash devices:

- Each read starts with the assertion of the appropriate memory bank chip select signals **SMCS[x]** and memory address **SMADDR[25:0]**.
- The read access time is determined by the number of wait states programmed for the WST1 field of the SMBWST1Rx register.
- The IDCY field in the SMBIDCYRx register determines the number of bus turnaround wait states added between external read and write transfers.

Figure 2-3 shows the timing diagram for an external memory read transfer with the minimum zero wait states (WST1 = 0), and the minimum zero output enable delay states (WSTOEN=0). A minimum of two AHB wait states are inserted during all single read transfers.

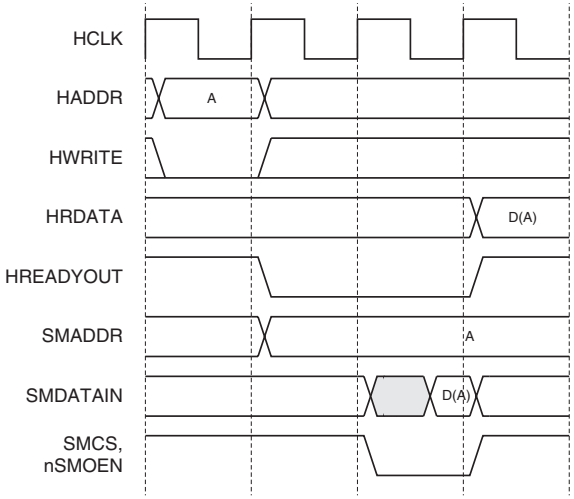


Figure 2-3 External memory zero wait state read

Figure 2-4 on page 2-11 shows the timing diagram for an external memory read transfer with two wait states (WST1 = 2), and the minimum zero output enable delay states (WSTOEN=0). Four AHB wait states are inserted during the transfer, two for the standard read, and an additional two because of the programmed wait states added.

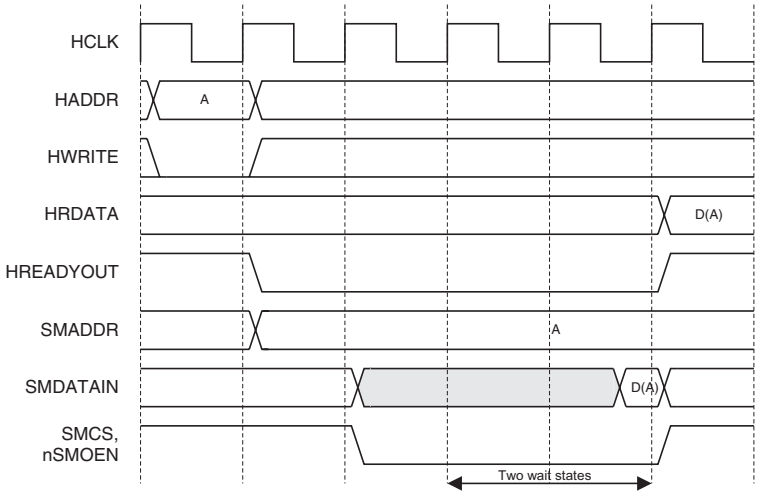


Figure 2-4 External memory two wait state read

Figure 2-5 on page 2-12 shows the timing diagram for an external memory read transfer with two output enable delay states (WSTOEN = 2) and two wait states (WST1 = 2). Four AHB wait states are inserted during the transfer, two for the standard read, and an additional two because of the output enable delay states added.

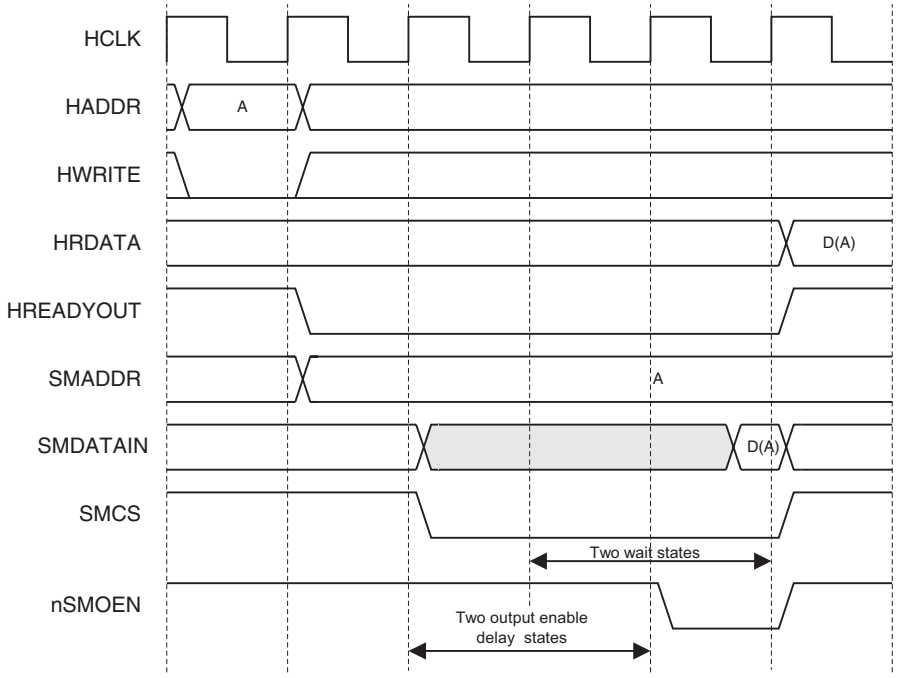


Figure 2-5 External memory two-output enable delay and two wait state read

Figure 2-6 on page 2-13 shows the timing diagram for an external memory read transfer with the minimum zero wait states where the SMC does not have control of the bus and must request for it. In this example nothing else is requesting the bus, so the SMC is granted straight away, showing the minimum timing when the bus is requested.

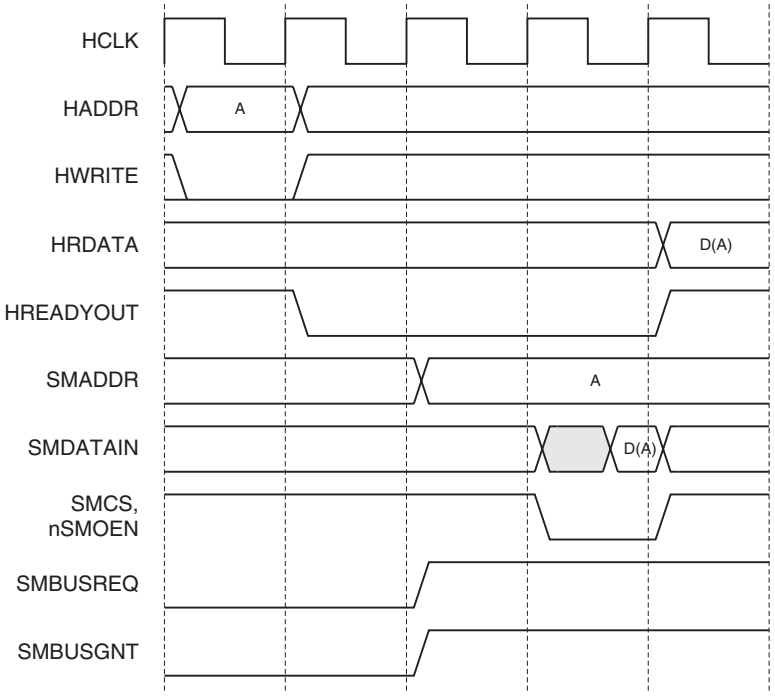


Figure 2-6 External memory zero wait state read, bus not granted

Figure 2-7 on page 2-14 shows the timing diagram for external memory read transfers with zero wait states (WST1 = 0). These might be nonsequential transfers, or sequential transfers of unspecified burst length. All transfers are treated as separate reads, so have the minimum of two AHB wait states added.

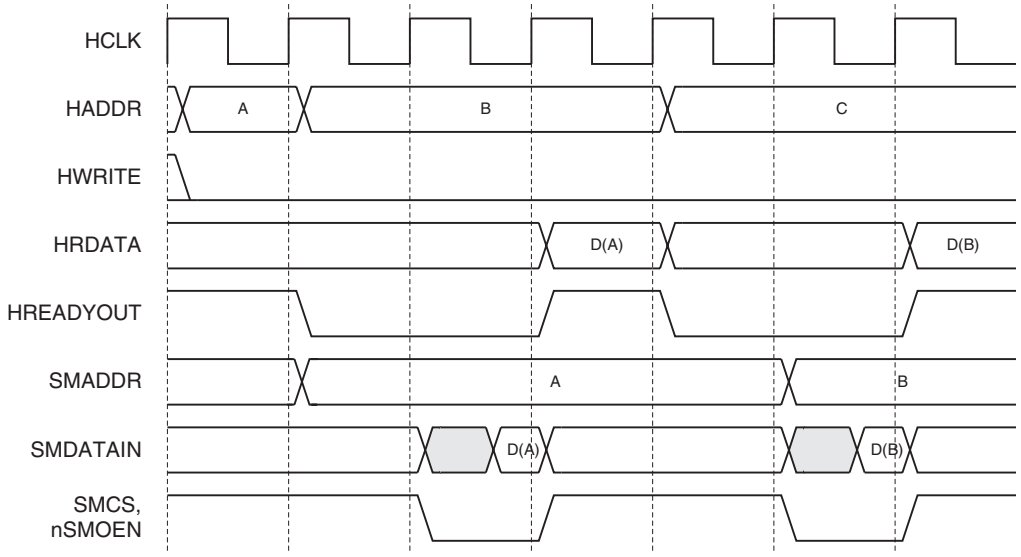


Figure 2-7 External memory three zero wait state read

Figure 2-8 on page 2-15 shows the timing diagram for a burst of zero wait state reads with the length specified. Because the length of the burst is known, it is possible to hold the chip select asserted during the whole burst, and generate the external transfers before the current AHB transfer has completed. Therefore, the first read has two AHB wait states added, and the three following sequential reads have zero AHB wait states added because of the automatic generation of the external transfers.

Note

During a fixed length burst, the chip select is held asserted during the transfer because it knows when the burst will end. However, if the AHB transfer width is smaller than the external memory width, after reading from memory and returning the data for the first AHB read, it can supply a number of subsequent AHB reads from the buffer, so **nSMOEN** is deasserted because it is carrying out a buffer read instead of a memory read. **nSMOEN** is then asserted again when it must do a memory read to fill the buffer. The address does not change until the next transfer, so it can hold the output enable asserted all of the time to match the chip select.

If the memory and AHB transfer sizes are the same, or if it is an undefined length INCR transfer, then the **nSMOEN** output must be driven in the same way as the chip select. It must be asserted for the duration of the burst.

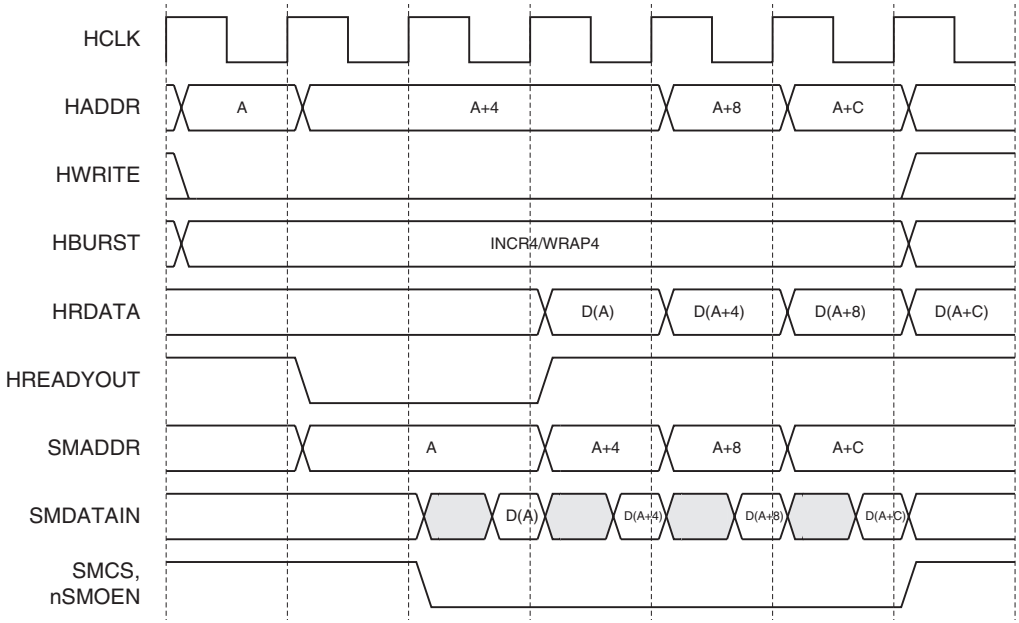


Figure 2-8 External memory zero wait fixed-length read

Figure 2-9 on page 2-16 shows the timing diagram for a burst of two wait state reads with the length specified. The WST1 value is used for all transfers in the burst. The first read has four AHB wait states inserted. All sequential transfers have two AHB wait states.

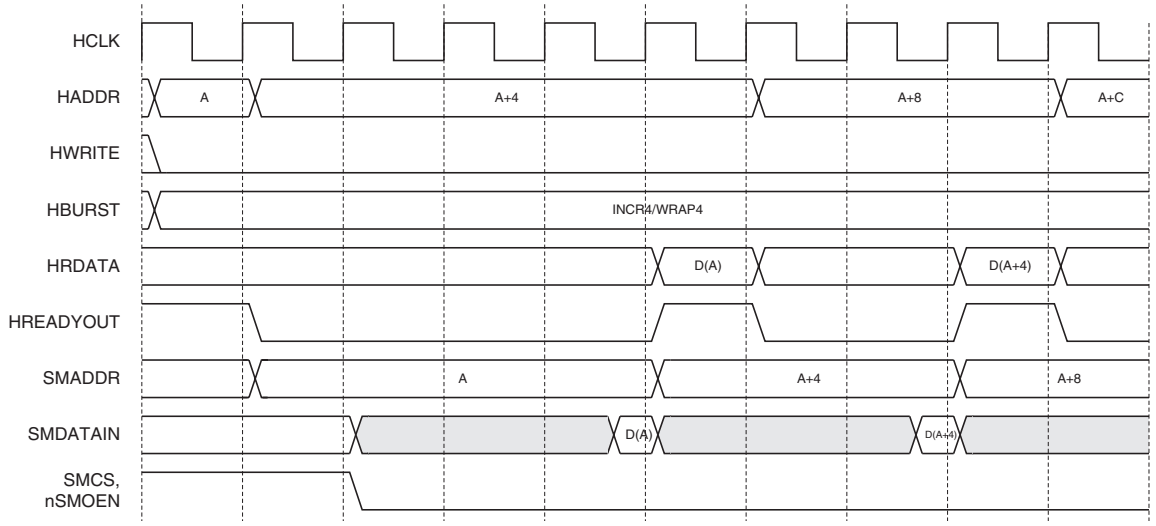


Figure 2-9 External memory two wait states fixed-length burst read

2.5.3 Burst ROM

The SMC supports sequential access burst reads to a maximum of four consecutive locations in 8, 16 or 32-bit memories. This supports burst mode ROM devices and increases bandwidth by using a reduced (configurable) access time for the sequential reads (WST2) following the first read (WST1). The chip select and output enable lines are held during the burst, and only the address changes between subsequent accesses. At the end of the burst the chip select and output enable lines are deasserted together.

Note

Bursts cannot cross quad boundaries. The boundaries are:

- **SMADDR[1:0]** = 11 for 8-bit transfers
- **SMADDR[2:1]** = 11 for 16-bit transfers
- **SMADDR[3:2]** = 11 for 32-bit transfers.

They are split up so that the first transfer after the boundary uses the slow read (WST1) timing. For example, a four-byte transfer starting at address **SMADDR[1:0]** = 01 performs a slow read from address 01, two fast reads from 10 and 11, and then a final slow read from address 00 to finish the burst.

Figure 2-10 shows the timing diagram for an external memory burst read transfer with two initial wait states, and one sequential wait state. The first read has four AHB wait states inserted, and all additional sequential transfers have only one AHB wait state. This gives increased performance over the equivalent nonburst ROM timing shown in Figure 2-9 on page 2-16.

Note

External burst transfers are always split up into bursts of maximum four transfers, with the first read using the slow timing and the subsequent three reads using the fast timing, because of the four transfer burst limit of burst ROM devices.

This four transfer limit is only applied to the external transfers. An AHB burst with size greater than the external memory is split up into bursts of four external transfers, taking into account any quad boundary connections.

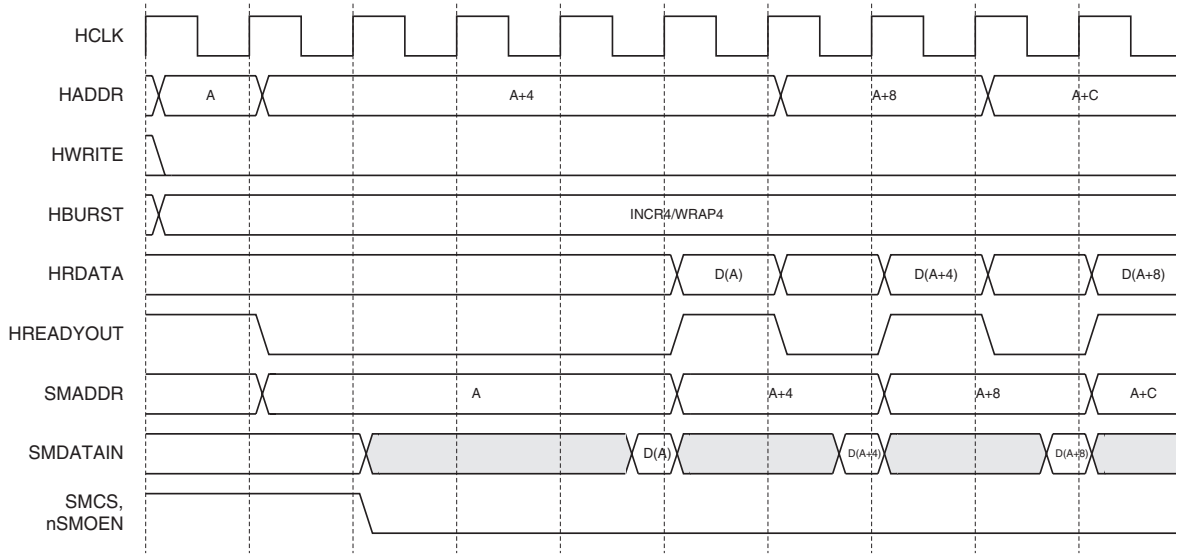


Figure 2-10 External burst ROM WST1 = 2, WST2 = 1, fixed length burst read

Figure 2-11 on page 2-18 shows the timing diagram for a 32-bit read from an 8-bit burst ROM device, causing four burst reads to be performed. A total of five AHB wait states are added during this transfer, two for the first external read, and then one for each of the subsequent reads.

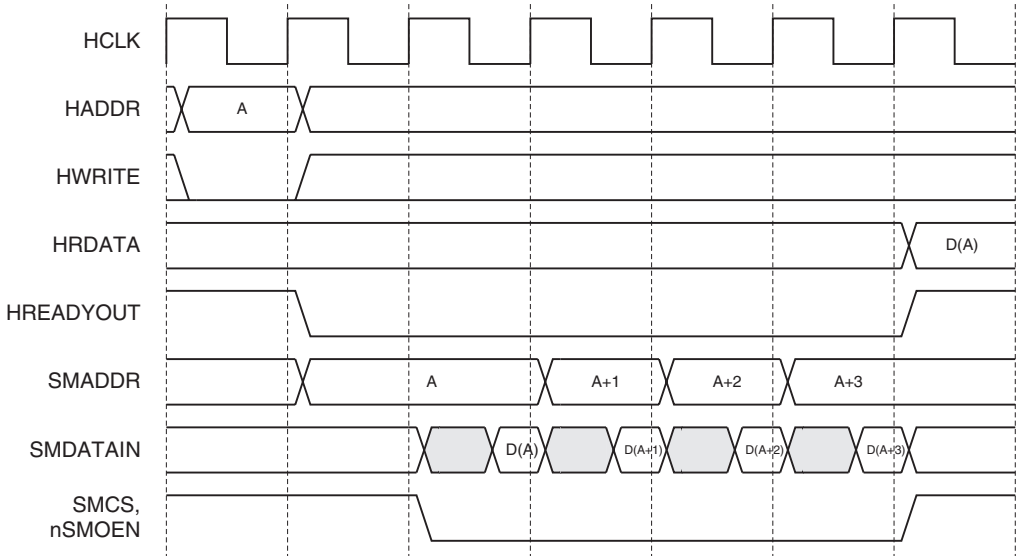


Figure 2-11 External memory 32-bit burst read from 8-bit memory

2.5.4 Burst flash

The SMC supports sequential access burst reads from burst flash devices, of the same types as for burst ROM. Because of the sharing of WST2 between write transfers and burst read transfers, it is only possible to have one setting at a time for burst flash, either the write delay or the burst read delay. This means that for a write transfer, WST2 must be programmed with the write delay value, and for a burst read transfer, WRT2 must be programmed with the burst access delay value.

2.6 Static memory write control

Write timing is described in the following sections:

- *Write enable programmable delay*
- *SRAM*
- *Flash memory* on page 2-24.

2.6.1 Write enable programmable delay

You can program the delay between the assertion of the chip select and the write enable from 0-15 cycles using the WSTWEN bits of the SMBWSTWENRx Registers. This delay is used to reduce the power consumption for memories. The write enable is asserted on the second falling edge of **HCLK** after the assertion of the chip select for one wait state (a one-cycle delay over the zero wait state timing), and so on for wait state values up to 15. The write enable is always deasserted on the falling edge of **HCLK** before the chip select, at the end of the transfer. **nSMBLS** has the same timing as **nSMWEN** for writes to 8-bit devices that use the byte lane selects instead of the write enables.

Note

The WSTOEN programmed value must be less than or equal to the WST1 value and WSTWEN must be less than or equal to WST2. In the External Wait timing mode, the timing of the transfer (controlled by **SMWAIT**) is not known, so **nSMWEN** is asserted on the falling edge of **HCLK** (rising edge of **nHCLK**) after the assertion of the chip select.

2.6.2 SRAM

Write timing for SRAM starts with assertion of the appropriate memory bank chip selects **SMCS[x]** and address signals **SMADDR[25:0]**. The write access time is determined by the number of wait states programmed for the WST2 field of the SMBWST2Rx Register. The IDCY field in the SMBIDCYRx Register determines the number of bus turnaround wait states added between external read and write transfers.

Figure 2-12 on page 2-20 shows the timing diagram for a single external memory write transfer with minimum zero wait states (WST2 = 0), and the minimum zero write enable delay states (WSTWEN=0). No AHB wait states are added.

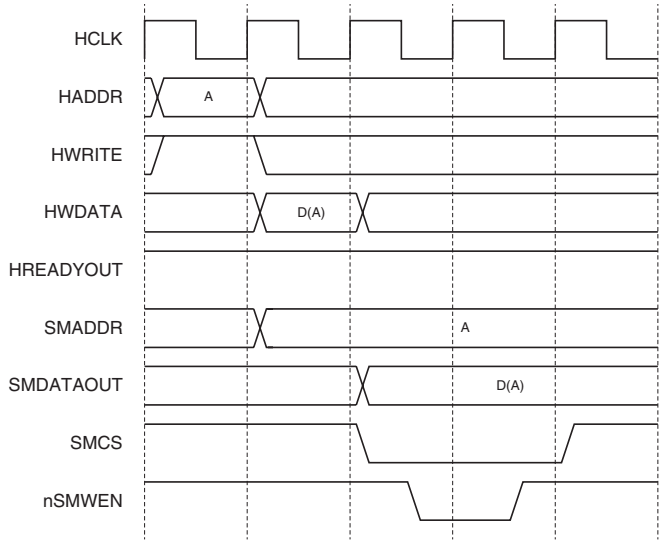


Figure 2-12 External memory zero wait state write

Figure 2-13 shows the timing diagram for a single external memory write transfer with two wait states (WST2 = 2), and the minimum zero write enable delay states (WSTWEN=0).

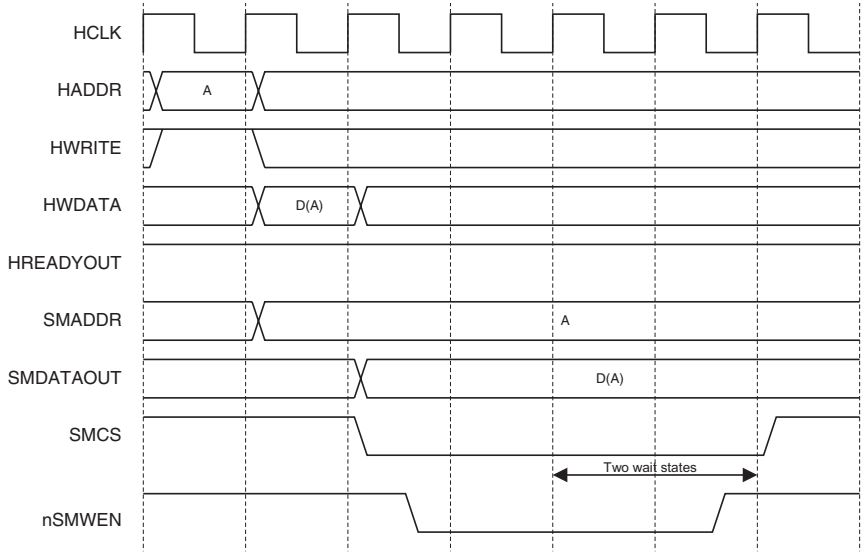


Figure 2-13 External memory two wait state write

Figure 2-14 shows the timing diagram for a single external memory write transfer with two write enable delay states (WSTWEN = 2) and two wait states (WST2 = 2). No AHB wait states are added.

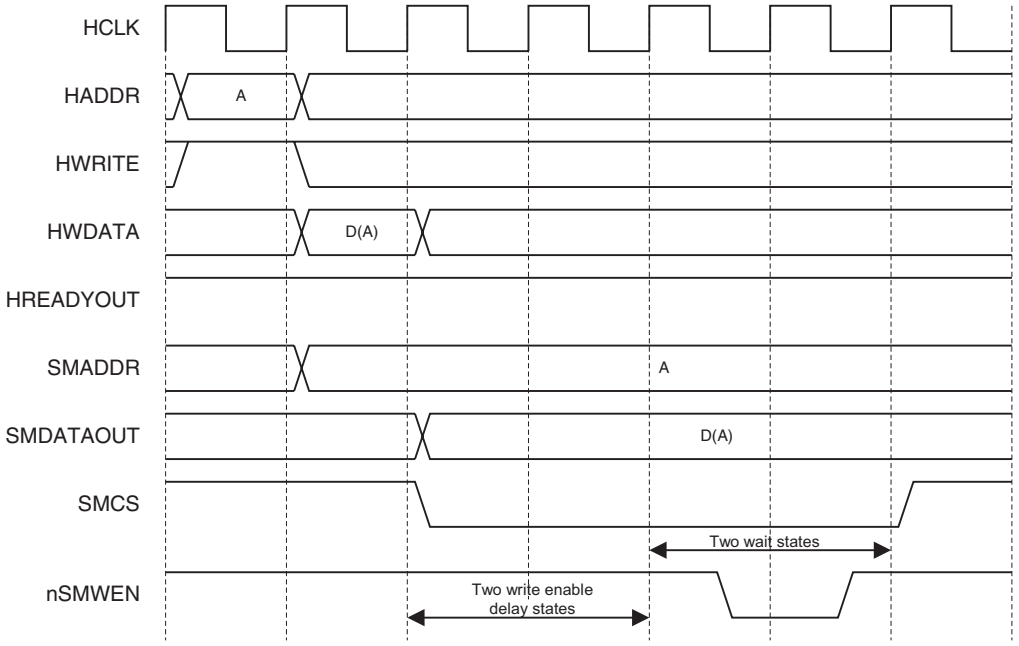


Figure 2-14 External memory two write enable delay and two wait state write

Figure 2-15 on page 2-22 shows the timing diagram for a single external memory write transfer with minimum zero wait states where the SMC does not have control of the bus and must request for it. In this example, nothing else is requesting the bus, so the SMC is granted straight away, showing the minimum timing when the bus is requested.

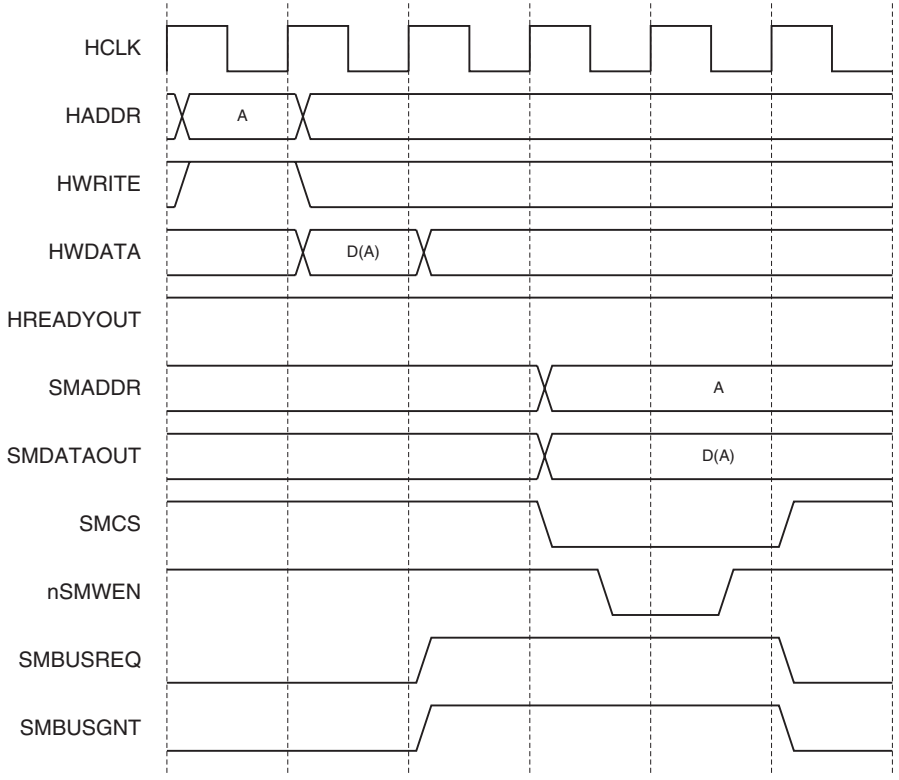


Figure 2-15 External memory zero wait state write, bus not granted

Figure 2-16 on page 2-23 shows the timing diagram for a single external memory write transfer with minimum zero wait states where the SMC does not have control of the bus and must request for it using an external synchronous bus multiplexor when a device such as an SDRAM controller is used. In this example nothing else is requesting the bus, so the SMC is granted straight away, showing the minimum timing when the bus is requested. See *Using the SMC with an external bus multiplexor or SDRAM controller* on page 2-57 for more details on the settings required for use with an SDRAM controller.

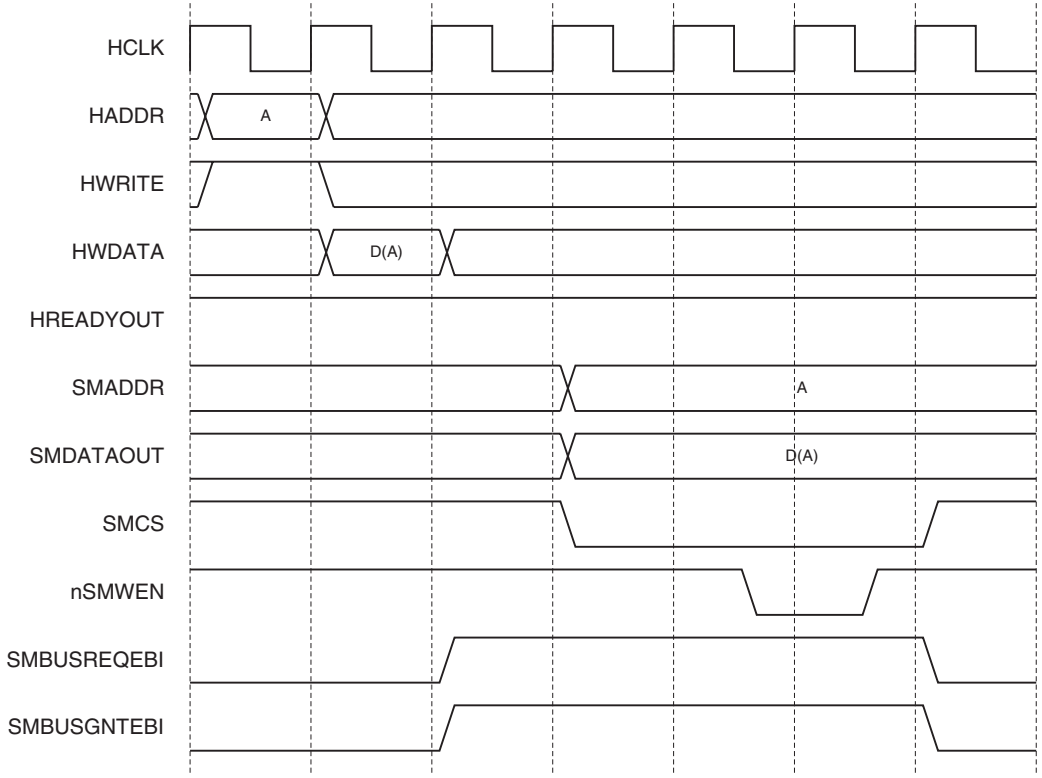


Figure 2-16 External memory zero wait state write, bus not granted, external synchronous bus multiplexor

Figure 2-17 on page 2-24 shows the timing diagram for two external memory write transfers with zero wait states (WST2 = 0). AHB wait states are added until the completion of the second write transfer. This is the timing of any sequence of write transfers:

- nonsequential to nonsequential
- nonsequential to sequential with any value of **HBURST**.

The maximum speed of write transfers is controlled by the external timing of the write enable relative to the chip select. All external writes must take a minimum of two cycles to complete:

- the write enable asserted cycle
- the write enable deasserted cycle.

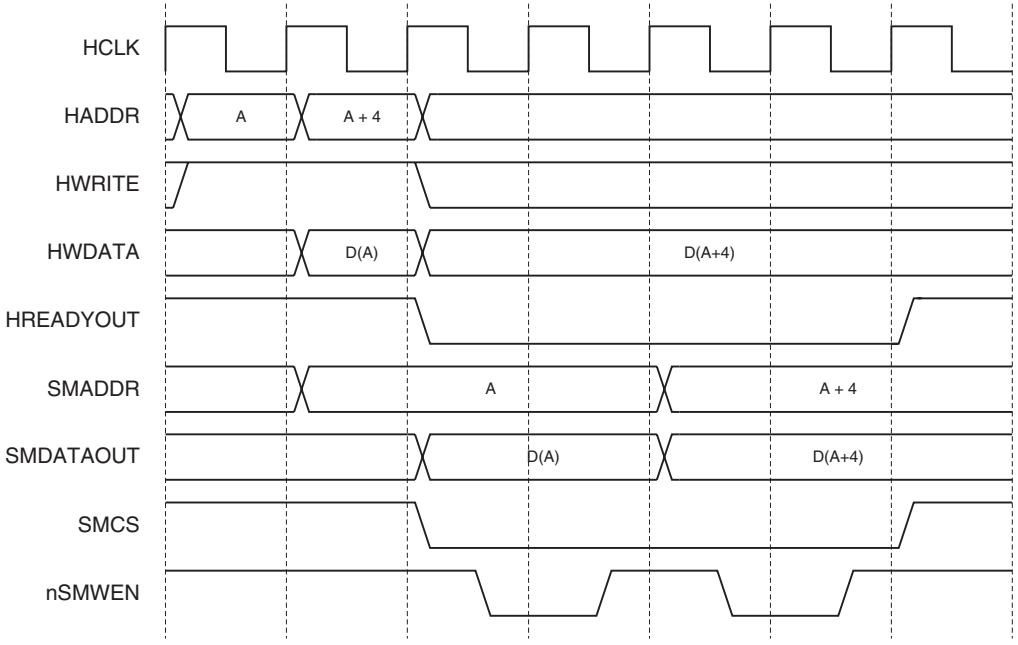


Figure 2-17 External memory two zero wait writes

2.6.3 Flash memory

Write timing for flash memory devices is the same as for SRAM devices. For more information, see *SRAM* on page 2-19.

2.7 Bus turnaround

The SMC can be configured for each memory bank to use external bus turnaround cycles between read and write memory accesses. The IDCY field can be programmed for up to 15 bus turnaround wait states. This is to avoid bus contention on the external memory data bus. Bus turnaround cycles are generated between external bus transfers as follows:

- read-to-read, to different memory banks
- read-to-write, to the same memory bank
- read-to-write, to different memory banks.

Figure 2-18 shows the timing diagram for a zero wait read followed by a zero wait write with default turnaround between the transfers of two cycles because of the timing of the AHB transfers. Standard AHB wait states are added to the transfers, two for the read, and zero for the write.

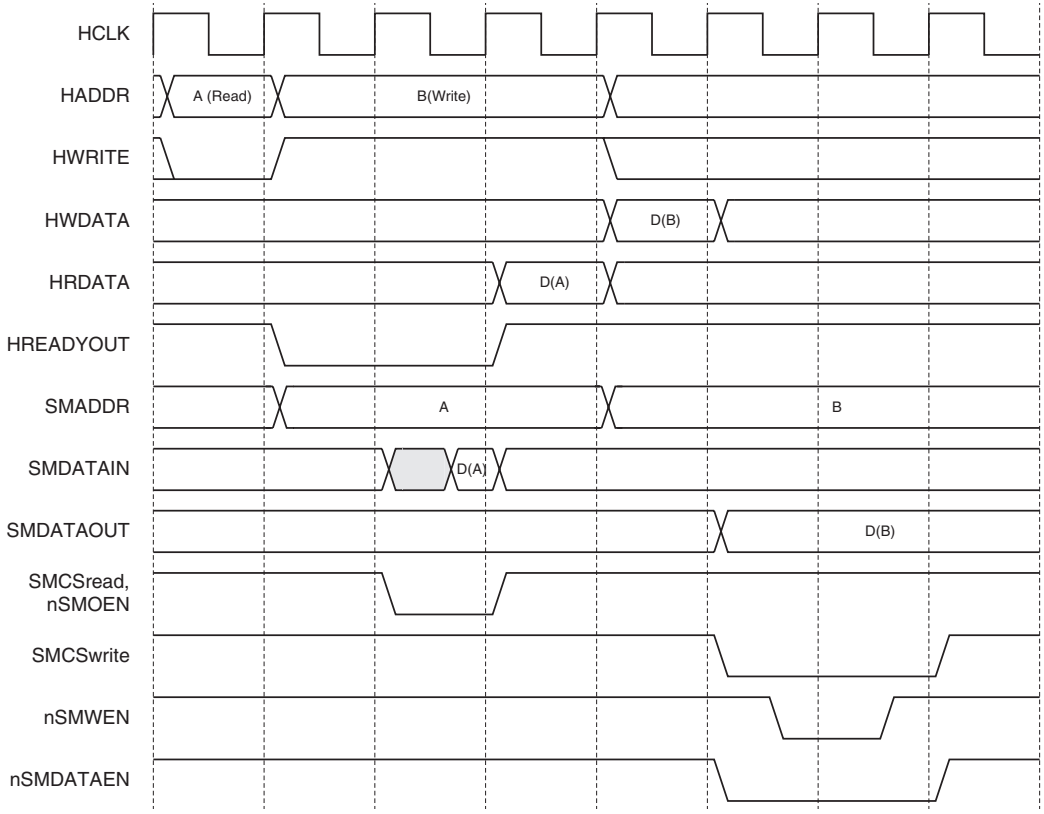


Figure 2-18 Read followed by write (both zero wait) with no turnaround

Figure 2-19 shows the timing diagram for a zero wait write followed by a zero wait read with default turnaround between the transfers of one cycle. No AHB wait states are added to the write transfer, but four are added to the read, two to enable the write to complete before the read is started, and then the standard two for the read transfer.

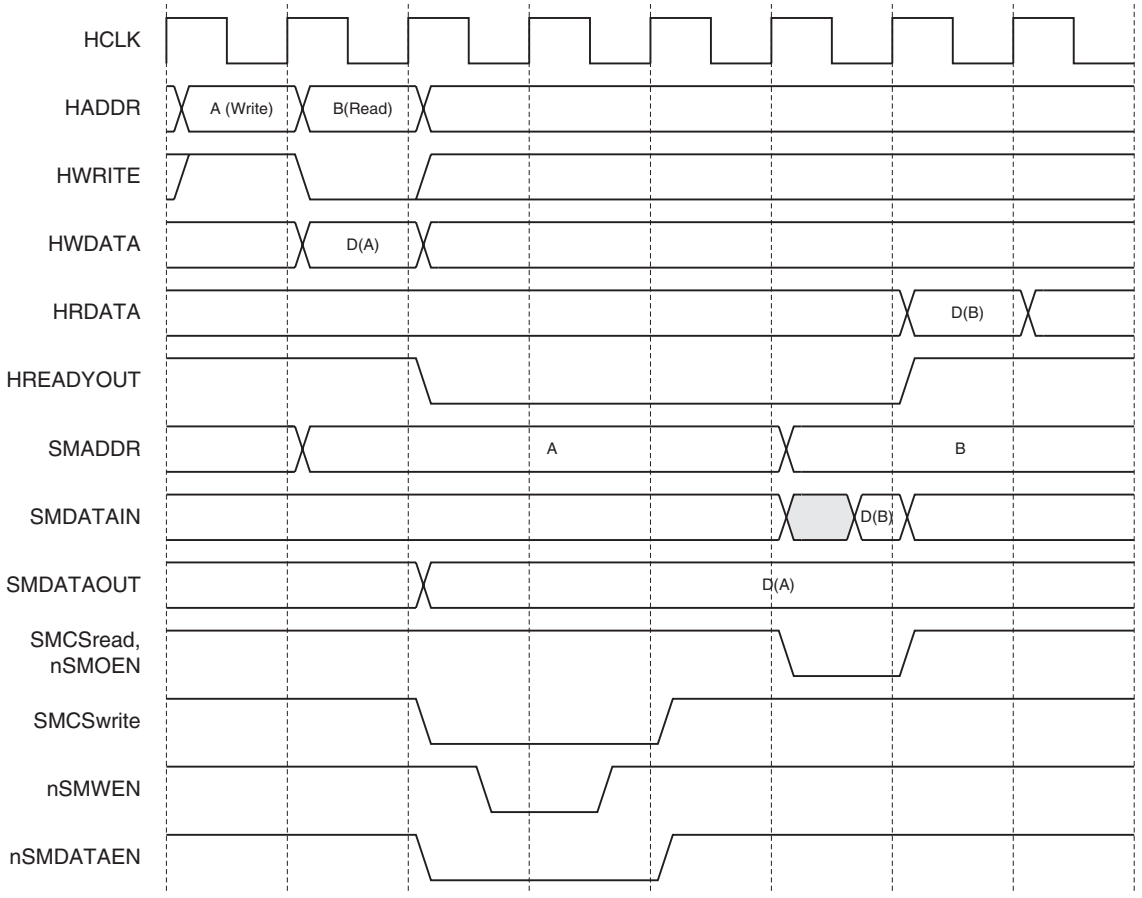


Figure 2-19 Write followed by read (both zero wait) with no turnaround

Figure 2-20 on page 2-27 shows the timing diagram for zero wait read followed by two zero wait writes with two turnaround cycles added. The standard minimum of two AHB wait states are added to the read transfer, and none are added to the first write (as for any read-write transfer sequence). Two AHB wait states are added to the second write because of insertion of the two turnaround cycles that are only generated after the first write transfer is detected.

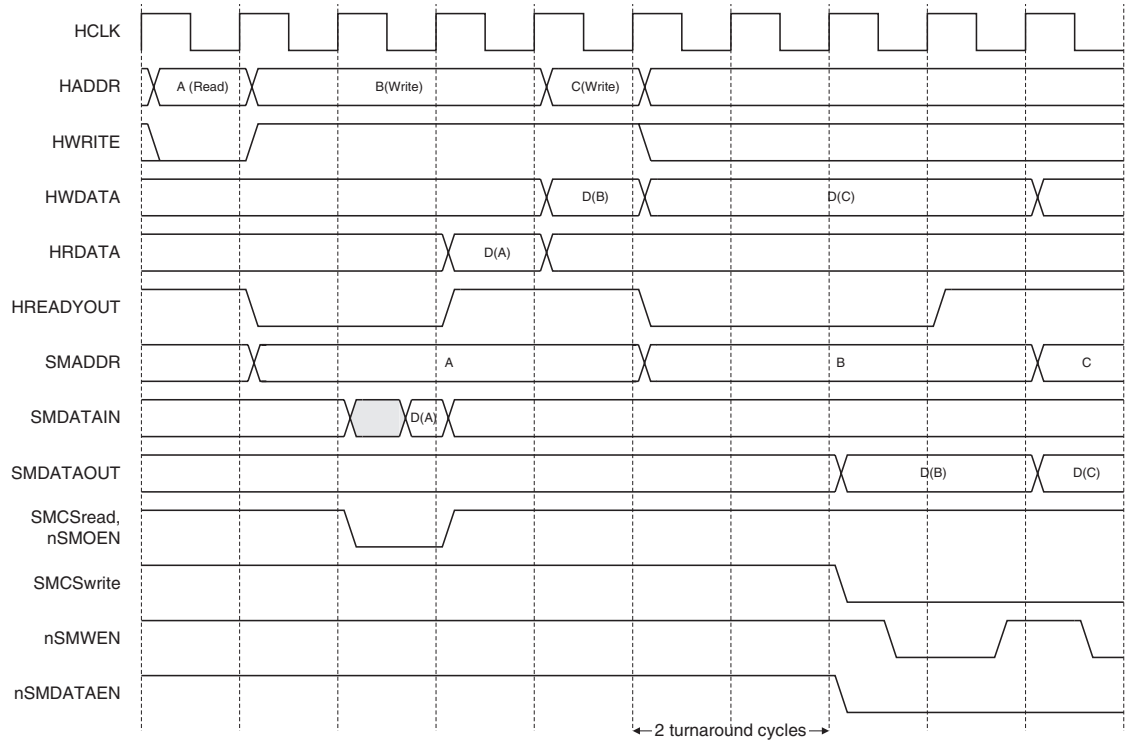


Figure 2-20 Read followed by two writes (all zero wait state) with two turnaround cycles

2.8 External wait control

The SMC supports the extension of the access cycle by an external device like a memory controller by using the **SMWAIT** input pin of the PrimeCell SMC. For this, the WaitEn bit of the SMBCRx register must be programmed appropriately. The polarity of the external **SMWAIT** input is programmed through the WaitPol field of the SMBCRx Register. If a problem occurs the active HIGH **CANCELSMWAIT** input enables externally waited transfers to be terminated early.

Note

Because the external wait control inputs (**SMWAIT** and **CANCELSMWAIT**) are asynchronous inputs, they are synchronized before use. This gives all operations using external waits a two cycle delay because of the synchronization time. Only single transfers might be performed using the external wait control. Burst transfers controlled by the external wait are not supported, and might result in unpredictable behavior.

When external wait mode is enabled, the SMC checks for assertion of the **SMWAIT** input, and waits the current transfer while **SMWAIT** stays asserted. The transaction completes when the **SMWAIT** line is deasserted (taking into account the two-cycle synchronization delay).

If the external wait control mode is disabled, then the SMC ignores the **SMWAIT** input and the access time is generated normally according to the values programmed in the SMBWST1Rx and SMBWST2Rx Registers.

2.8.1 SMWAIT assertion timing

When the SMC is waiting for the **SMWAIT** assertion in wait enabled or external wait control mode, it also starts counting down according to the values programmed in the wait state count field WST1 or WST2, used for read and write transfers respectively. You can use this feature to ensure that adequate time is available to the SMC to detect **SMWAIT** because there might be a delay before the external device asserts **SMWAIT**. If **SMWAIT** is not asserted during this time, the transfer is assumed to be zero wait.

Note

When you use the **SMWAIT** input to time memory transfers, the SMBWST1Rx and SMBWST2Rx Registers are used to program the external wait assertion delay. You must set these registers to a minimum of `0x03` instead of the default of `0x00` for standard memory transfers. This requires:

- one cycle for the minimum chip select to external wait assertion delay
- two cycles for the double synchronization of the **SMWAIT** input before use.

2.8.2 SMWAIT deassertion timing

A waited transfer only ends when the **SMWAIT** input has been deasserted. The **CANCELSMWAIT** input is provided to enable the system to recover if the external device waits the transfer for longer than expected. A timer or watchdog must be used to control the assertion of the active HIGH **CANCELSMWAIT** input, and it must be asserted for a minimum of one cycle. If a waited transfer is terminated before it has completed successfully, then an AHB error response is generated, and the WaitToutErr flag in the SMBSRx Register is asserted. Because an external wait stops any external transfers being performed on the external bus, the SMC generates an error response when a transfer is requested to any external location and the external wait input is still asserted from a previous transfer. This continues until the waited transfer is complete (**SMWAIT** deasserted), and then operation continues as normal.

The WaitStatus bit of the SMBSRx Register can be used to check the current status of **SMWAIT** after a terminated transfer.

2.8.3 SMWAIT timing diagrams

Figure 2-21 shows the timing diagram for an externally waited read transfer, taking two cycles for the wait to be asserted, and two cycles for the wait to be deasserted. The synchronization of the asynchronous **SMWAIT** input adds a further two clock cycles onto the timing of the transfer.

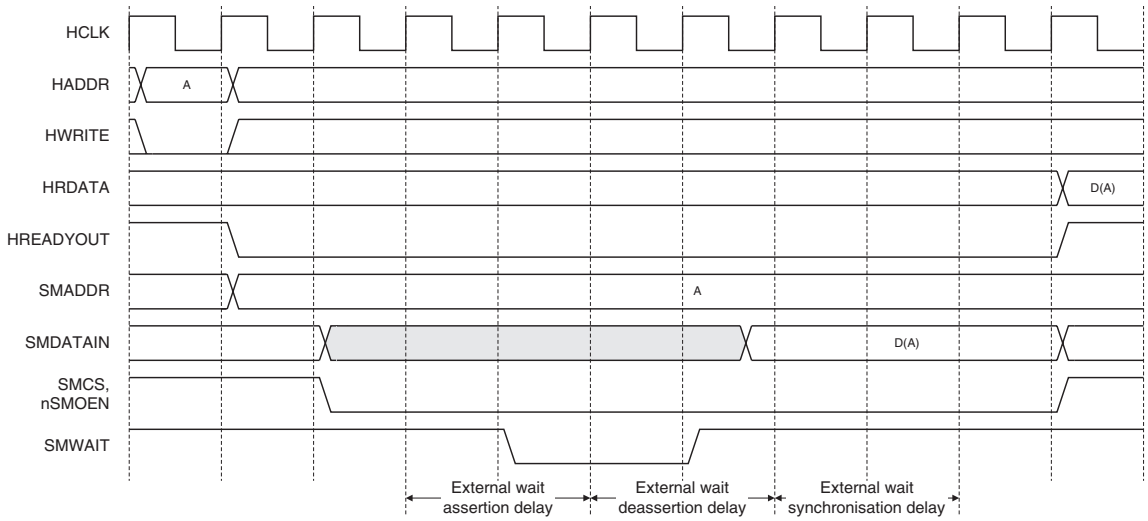


Figure 2-21 External wait timed read transfer

Figure 2-22 shows the timing diagram for an externally waited write transfer, taking two cycles for the wait to be asserted, and two cycles for the wait to be deasserted. An additional cycle is required at the end of the transfer over a read transfer to enable the deassertion of the write enable before the chip select. An externally waited transfer is also waited on the AHB, unlike a standard write transfer, which enables an error response because of a timeout to be generated correctly.

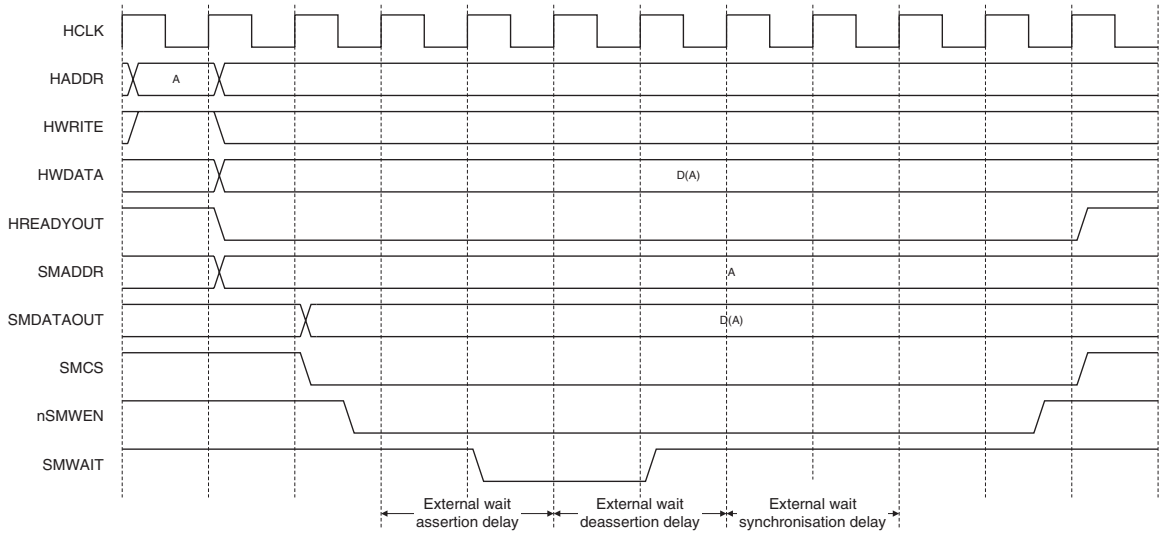


Figure 2-22 External wait timed write transfer

Figure 2-23 on page 2-31 shows the timing diagram for an aborted externally waited transfer.

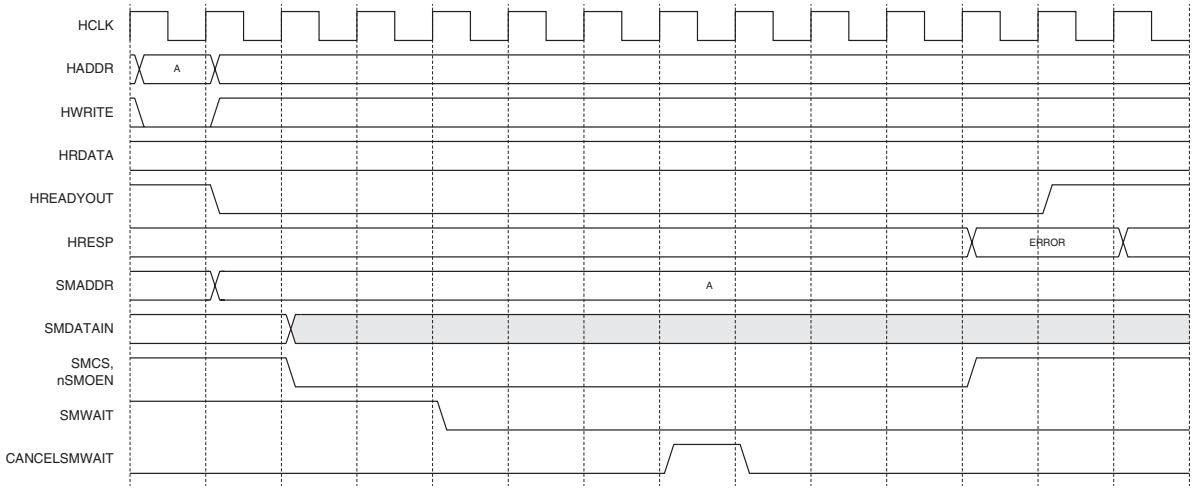


Figure 2-23 External wait timed read transfer with external abort

2.9 Byte lane control

The SMC generates byte lane control signals **nSMBLS[3:0]** according to:

- little or big-endian operation
- AMBA transfer width (indicated by **HSIZE[2:0]**)
- external memory bank data bus width, defined within each SMBCRx Register
- external memory bank type, being either byte, halfword or word
- the decoded **HADDR[1:0]** value for write accesses only.

Word transfers are the largest size transfers supported by the SMC. Any access attempted with a size greater than a word generates the error response.

Each memory bank can either be 8, 16, or 32 bits wide. The memory configuration for a particular memory bank determines how the **nSMWEN** and **nSMBLS** signals are connected to provide byte, halfword, and word access. For read accesses, it is necessary to control the **nSMBLS** signals by driving them either all HIGH, or all LOW.

This control is achieved by programming the RBLE bit in each SMBCRx Register. The following two sections explain why different connections in respect of **nSMWEN** and **nSMBLS[3:0]** are required for different memory configurations.

2.9.1 Memory banks constructed from 8-bit or non byte-partitioned memory devices

For memory banks constructed from 8-bit or non byte-partitioned memory devices, it is important that the RBLE bit is cleared to zero within the respective memory bank SMBCRx Register. This forces all **nSMBLS[3:0]** lines HIGH during a read access to that particular bank, because the byte lane selects are connected to the device write enables.

Figure 2-24 on page 2-33 shows 8-bit memory devices being used to configure memory banks that are 8, 16 and 32 bits wide. In each of these configurations, the **nSMBLS[3:0]** signals are connected to the write enable (**nWE**) inputs of each 8-bit memory.

Note

The **nSMWEN** signal from the PrimeCell SMC is not used in this configuration.

For write transfers, the relevant **nSMBLS[3:0]** byte lane signals are asserted LOW, and steer the data to the addressed bytes.

For read transfers, all **nSMBLS[3:0]** lines are deasserted HIGH, which enables the external bus to be defined for at least the width of the accessed memory.

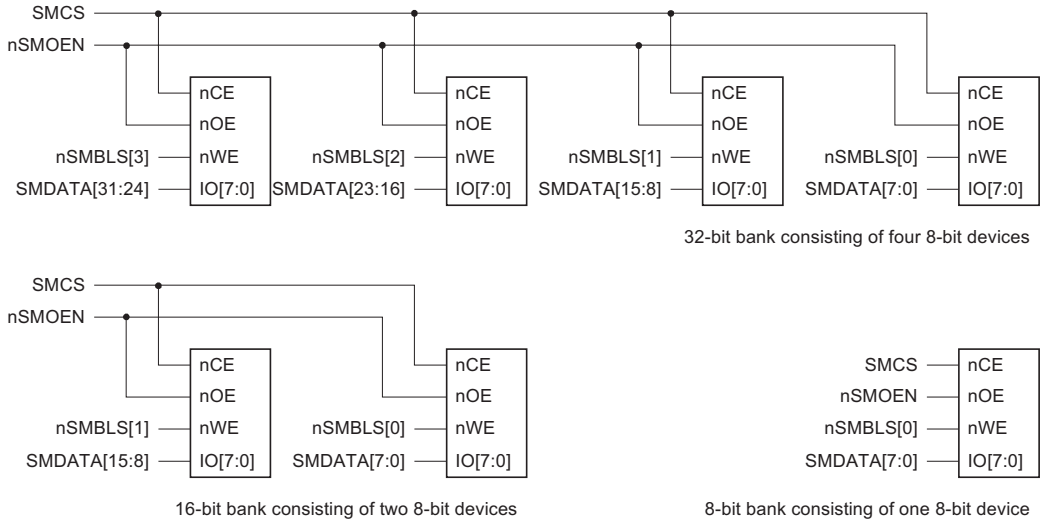


Figure 2-24 Memory banks constructed from 8-bit memory

2.9.2 Memory banks constructed from 16 or 32-bit memory devices

For memory banks constructed from 16 or 32-bit memory devices, it is important that the RBLE bit is set to one within the respective memory bank SMBCRx Register. This asserts all **nSMBLS[3:0]** lines LOW during a read access to that particular bank, because during a read all bytes of the devices must be selected to avoid undriven byte lanes on the read data value.

For 16 and 32-bit wide memory devices, byte select signals exist and these must be appropriately controlled, as shown in Figure 2-25 on page 2-34 and Figure 2-26 on page 2-34.

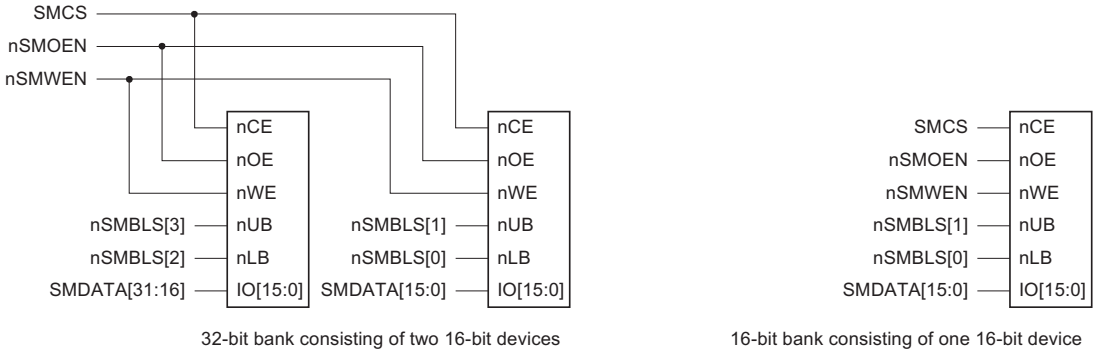


Figure 2-25 Memory banks constructed from 16-bit memory

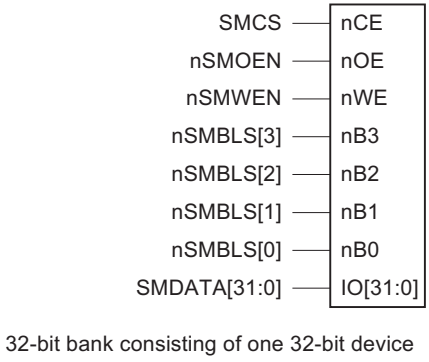


Figure 2-26 Memory banks constructed from 32-bit memory

Figure 2-27 on page 2-35 shows connections for a typical memory system with different data-width memory devices.

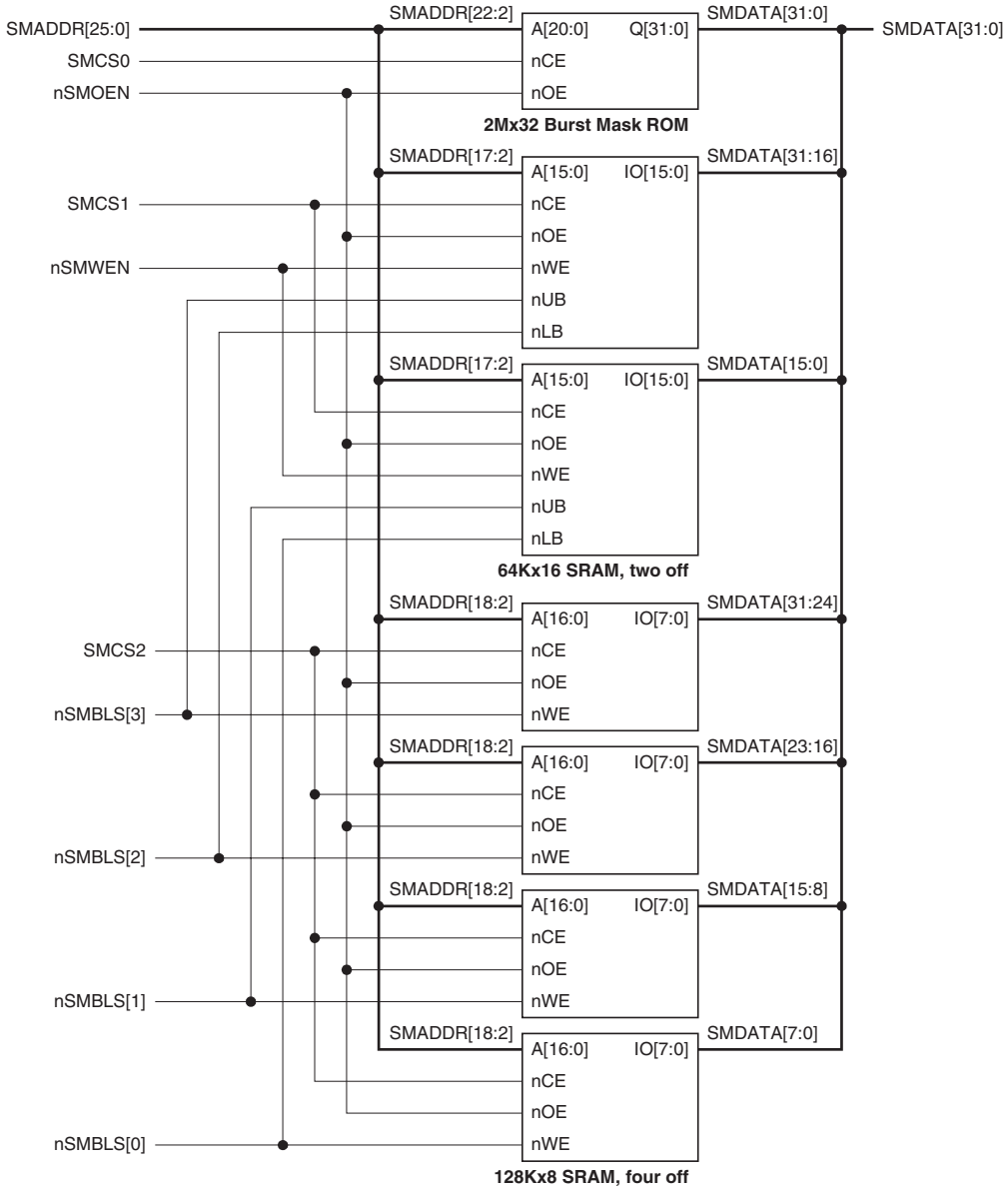


Figure 2-27 Typical memory connection diagram

2.9.3 Elimination of floating bytes on the external interface

The PrimeCell SMC uses the programmed external memory width of each bank and the endianness to determine which remaining bytes it requires to drive to ensure that the external bus is never floating.

The input/output pads cells for the external write data bus lines are controlled by **nSMDATAEN[3:0]** according to Table 2-3. Data is driven out on **SMDATAOUT** when **nSMDATAEN** is asserted LOW.

Table 2-3 SMDATAOUT controlled by nSMDATAEN

nSMDATAEN	SMDATAOUT
0	7:0
1	15:8
2	23:16
3	31:24

2.9.4 Byte lane control and data bus steering for little and big-endian configurations

Table 2-4 to Table 2-15 on page 2-42 show the relationship of signals **HSIZE[2:0], HADDR[1:0]**, **SMADDR[1:0]**, and **nSMBLS[3:0],** and mapping of data between the AHB system data bus and external memory data bus.

Table 2-4 Little-endian read, 8-bit external bus

Access: Read, little-endian, 8-bit external bus				External data mapping onto system data bus			
Internal transfer width	**HSIZE[1:0]**	**HADDR[1:0]**	**SMADDR[1:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	11	[7:0]	-	-	-
(4 transfers)	10	xx	10	-	[7:0]	-	-
	10	xx	01	-	-	[7:0]	-
	10	xx	00	-	-	-	[7:0]
Halfword	01	1x	11	[7:0]	-	-	-
(2 transfers)	01	1x	10	-	[7:0]	-	-
Halfword	01	0x	01	-	-	[7:0]	-
(2 transfers)	01	0x	00	-	-	-	[7:0]

Table 2-4 Little-endian read, 8-bit external bus (continued)

Access: Read, little-endian, 8-bit external bus				External data mapping onto system data bus			
Internal transfer width	**HSIZE[1:0]**	**HADDR[1:0]**	**SMADDR[1:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Byte	00	11	11	[7:0]	-	-	-
Byte	00	10	10	-	[7:0]	-	-
Byte	00	01	01	-	-	[7:0]	-
Byte	00	00	00	-	-	-	[7:0]

Table 2-5 Little-endian read, 16-bit external bus

Access: Read, little-endian, 16-bit external bus				External data mapping onto system data bus			
Internal transfer width	**HSIZE[1:0]**	**HADDR[1:0]**	**SMADDR[1:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	1x	[15:8]	[7:0]	-	-
(2 transfers)	10	xx	0x	-	-	[15:8]	[7:0]
Halfword	01	1x	1x	[15:8]	[7:0]	-	-
Halfword	01	0x	0x	-	-	[15:8]	[7:0]
Byte	00	11	1x	[15:8]	-	-	-
Byte	00	10	1x	-	[7:0]	-	-
Byte	00	01	0x	-	-	[15:8]	-
Byte	00	00	0x	-	-	-	[7:0]

Table 2-6 Little-endian read, 32-bit external bus

Access: Read, little-endian, 32-bit external bus				External data bus mapping on to system data bus			
Internal transfer width	**HSIZE[1:0]**	**HADDR[1:0]**	**SMADDR[1:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	xx	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	01	1x	xx	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	01	0x	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	11	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	10	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	01	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	00	xx	[31:24]	[23:16]	[15:8]	[7:0]

Table 2-7 Little-endian write, 8-bit external bus

Access: Write, little-endian, 8-bit external bus					System data mapping on to external data bus			
Internal transfer width	**HSIZE [1:0]**	**HADDR [1:0]**	**SMADDR [1:0]**	**nSMBLS [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	11	1110	-	-	-	[31:24]
(4 transfers)	10	xx	10	1110	-	-	-	[23:16]
	10	xx	01	1110	-	-	-	[15:8]
	10	xx	00	1110	-	-	-	[7:0]
Halfword	01	1x	11	1110	-	-	-	[31:24]
(2 transfers)	01	1x	10	1110	-	-	-	[23:16]
Halfword	01	0x	01	1110	-	-	-	[15:8]
(2 transfers)	01	0x	00	1110				[7:0]
Byte	00	11	11	1110	-	-	-	[31:24]
Byte	00	10	10	1110	-	-	-	[23:16]
Byte	00	01	01	1110	-	-	-	[15:8]
Byte	00	00	00	1110	-	-	-	[7:0]

Table 2-8 Little-endian write, 16-bit external bus

Access: Write, little-endian, 16-bit external bus					System data mapping on to external data bus			
Internal transfer width	**HSIZE [1:0]**	**HADDR [1:0]**	**SMADDR [1:0]**	**nSMBLS [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	1x	1100	-	-	[31:24]	[23:16]
(2 transfers)	10	xx	0x	1100	-	-	[15:8]	[7:0]
Halfword	01	1x	1x	1100	-	-	[31:24]	[23:16]
Halfword	01	0x	0x	1100	-	-	[15:8]	[7:0]
Byte	00	11	1x	1101	-	-	31:24	-
Byte	00	10	1x	1110	-	-	-	[23:16]
Byte	00	01	0x	1101	-	-	[15:8]	-
Byte	00	00	0x	1110	-	-	-	[7:0]

Table 2-9 Little-endian write, 32-bit external bus

Access: write, little-endian, 32-bit external bus					System data mapping on to external data bus			
Internal transfer width	**HSIZE [1:0]**	**HADDR [1:0]**	**SMADDR [1:0]**	**nSMBLS [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	xx	0000	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	01	1x	xx	0011	[31:24]	[23:16]	-	-
Halfword	01	0x	xx	1100	-	-	[15:8]	[7:0]
Byte	00	11	xx	0111	[31:24]	-	-	-
Byte	00	10	xx	1011	-	[23:16]	-	-
Byte	00	01	xx	1101	-	-	[15:8]	-
Byte	00	00	xx	1110	-	-	-	[7:0]

Table 2-10 Big-endian read, 8-bit external bus

Access: Read, big-endian, 8-bit external bus				External data mapping on to system data bus			
Internal transfer width	**HSIZE[1:0]**	**HADDR[1:0]**	**SMADDR[1:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	11	-	-	-	[7:0]
(4 transfers)	10	xx	10	-	-	[7:0]	-
	10	xx	01	-	[7:0]	-	-
	10	xx	00	[7:0]	-	-	-
Halfword	01	1x	11	-	-	-	[7:0]
(2 transfers)	01	1x	10	-	-	[7:0]	-
Halfword	01	0x	01	-	[7:0]	-	-
(2 transfers)	01	0x	00	[7:0]	-	-	-
Byte	00	11	11	-	-	-	[7:0]
Byte	00	10	10	-	-	[7:0]	-
Byte	00	01	01	-	[7:0]	-	-
Byte	00	00	00	[7:0]	-	-	-

Table 2-11 Big-endian read, 16-bit external bus

Access: Read, big-endian, 16-bit external bus				External data mapping on to system data bus			
Internal transfer width	**HSIZE[1:0]**	**HADDR[1:0]**	**SMADDR[1:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	1x	-	-	[15:8]	[7:0]
(2 transfers)	10	xx	0x	[15:8]	[7:0]	-	-
Halfword	01	1x	1x	-	-	[15:8]	[7:0]
Halfword	01	0x	0x	[15:8]	[7:0]	-	-
Byte	00	11	1x	-	-	-	[7:0]
Byte	00	10	1x	-	-	[15:8]	-
Byte	00	01	0x	-	[7:0]	-	-
Byte	00	00	0x	[15:8]	-	-	-

Table 2-12 Big-endian read, 32-bit external bus

Access: Read, Big-endian, 32-bit external bus				External data mapping on to system data bus			
Internal transfer width	**HSIZE[1:0]**	**HADDR[1:0]**	**SMADDR[1:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	xx	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	01	1x	xx	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	01	0x	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	11	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	10	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	01	xx	[31:24]	[23:16]	[15:8]	[7:0]
Byte	00	00	xx	[31:24]	[23:16]	[15:8]	[7:0]

Table 2-13 Big-endian write, 8-bit external bus

Access: Write, big-endian, 8-bit external bus					System data mapping on to external data bus			
Internal transfer width	**HSIZE [1:0]**	**HADDR [1:0]**	**SMADDR [1:0]**	**nSMBLS [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	11	1110	-	-	-	[7:0]
(4 transfers)	10	xx	10	1110	-	-	-	[15:8]
	10	xx	01	1110	-	-	-	[23:16]
	10	xx	00	1110	-	-	-	[31:24]
Halfword	01	1x	11	1110	-	-	-	[7:0]
(2 transfers)	01	1x	10	1110	-	-	-	[15:8]
Halfword	01	0x	01	1110	-	-	-	[23:16]
(2 transfers)	01	0x	00	1110	-	-	-	[31:24]
Byte	00	11	11	1110	-	-	-	[7:0]
Byte	00	10	10	1110	-	-	-	15:8
Byte	00	01	01	1110	-	-	-	23:16
Byte	00	00	00	1110	-	-	-	31:24

Table 2-14 Big-endian write, 16-bit external bus

Access: Write, big-endian, 16-bit external bus					System data mapping on to external data bus			
Internal transfer width	**HSIZE [1:0]**	**HADDR [1:0]**	**SMADDR [1:0]**	**nSMBLS [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	1x	1100	-	-	[15:8]	[7:0]
(2 transfers)	10	xx	0x	1100	-	-	[31:24]	[23:16]
Halfword	01	1x	1x	1100	-	-	[15:8]	[7:0]
Halfword	01	0x	0x	1100	-	-	[31:24]	[23:16]
Byte	00	11	1x	1110	-	-	-	[7:0]
Byte	00	10	1x	1101	-	-	[15:8]	-
Byte	00	01	0x	1110	-	-	-	[23:16]
Byte	00	00	0x	1101	-	-	[31:24]	-

Table 2-15 Big-endian write, 32-bit external bus

Access: Write, big-endian, 32-bit external bus					System data mapping on to external data bus			
Internal transfer width	**HSIZE [1:0]**	**HADDR [1:0]**	**SMADDR [1:0]**	**nSMBLS [3:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Word	10	xx	xx	0000	[31:24]	[23:16]	[15:8]	[7:0]
Halfword	01	1x	xx	1100	-	-	[15:8]	[7:0]
Halfword	01	0x	xx	0011	[31:24]	[23:16]	-	-
Byte	00	11	xx	1110	-	-	-	[7:0]
Byte	00	10	xx	1101	-	-	[15:8]	-
Byte	00	01	xx	1011	-	[23:16]	-	-
Byte	00	00	xx	0111	[31:24]	-	-	-

2.10 Memory shadowing

This section describes how the SMC can be configured to enable boot ROM to be available instead of RAM at the base memory location immediately after reset, and how the boot ROM device size can be configured at reset.

2.10.1 Booting from ROM after reset

There are two possible configurations for a system that uses boot ROM:

- internal ROM
- external ROM.

For a system with internal ROM, the AHB Decoder handles selecting this device after reset to enable the processor to start executing from address `0x00000000`. For a system with external ROM, the external memory controller must ensure that this device is available at address `0x00000000` no matter which chip select it is connected to.

In a system with internal RAM and external ROM, the most likely memory map configuration is to have the internal RAM at address `0x00000000`. Because this RAM is likely to be quite small, it is useful to overlap the normally larger external RAM with this internal RAM, so the external RAM is located on bank zero and also at address `0x00000000`. This enables an easy flow from internal RAM to external RAM, and makes it possible for the software to continue accessing the RAM without having to write to a completely different location if it exceeds the size of the internal RAM.

To accommodate this overlap of internal and external RAMs, the external boot ROM must be on a bank other than bank zero. For the Primecell SMC this boot ROM is available on bank seven.

The SMC therefore provides a **REMAP** input that is used to control the shadowing of the **SMCS[7]** memory area onto the **SMCS[0]** memory area after reset (**REMAP** = 0). This makes the memory at **SMCS[0]** unavailable until the system sets the **REMAP** input. When the **REMAP** input has been set HIGH (**REMAP** = 1), the **SMCS[7]** memory area is only available at **SMCS[7]**, and the **SMCS[0]** memory area is available for normal use.

Chip selects **SMCS[1]** to **SMCS[6]** are unaffected by the **REMAP** input and are always available for access.

Note

The **REMAP** input must be generated as part of the users system, probably as part of the system controller logic, and when LOW indicates that the system requires external boot ROM to be available at address `0x00000000`. If the remapping function is not required, then **REMAP** must be tied HIGH to disable memory bank shadowing.

2.10.2 External bank SMCS7 size configuration

The SMC enables the size of bank seven (selected with **SMCS[7]**) to be configured at reset using the external control pins **SMMWCS7[1:0]** and **SMRBLECS7**. This feature is used to enable the boot memory size to be chosen at reset.

Bank seven can therefore always have the correct setting, and no additional software programming of the SMBCR7 Register is required. However, the MW and RBLE bits of the SMBCR7 Register are still programmable, the same as the rest of the bank SMBCRx Registers. If you choose not to reconfigure the size of bank seven (by tying the pins to 11), then the bank size defaults to 8-bits.

Note

All other memory bank sizes have default settings that are set in the hardware, but can be programmed to the required values using software.

The configuration of the **SMMWCS7** bits used is shown in Table 2-16.

Table 2-16 External size configuration values for bank seven

SMMWCS7[1:0]	Memory width
00	8-bit
01	16-bit
10	32-bit
11	No change

The **SMRBLECS7** settings match those shown for bit 0 of the SMBCRx Registers in Table 3-8 on page 3-12.

2.11 Test interface controller operation

The *Test Interface Controller* (TIC) is a state machine that provides an AMBA AHB bus master for system test. It reads test write and address data from the external data bus **SMDATA**, and drives the external bus with test read data, enabling the use of only one set of output tristate buffers onto **SMDATA**.

The TIC is used to convert externally applied test vectors into internal transfers on the AHB bus. A three-wire external handshake protocol is used, with two inputs controlling the type of vector that is applied and a single output that indicates when the next vector can be applied. Typically the TIC is the highest priority AMBA bus master, which ensures test access under all conditions.

The TIC model supports address incrementing and control vectors. This means that the address for burst transfers can automatically be generated by the TIC.

The TIC is described in the following subsections:

- *Function and operation of module*
- *Test vector types* on page 2-46
- *Control vectors* on page 2-47
- *Test vector sequences* on page 2-48
- *Data bus interface operation* on page 2-55.

2.11.1 Function and operation of module

The TIC operates as a standard AHB bus master during system test when the external test pins show that the system is required to enter test mode. In this mode, the TIC requests control of the AHB, and when granted uses the AHB to perform system tests.

Table 2-17 shows the operation of the external test pins to change the TIC mode from normal operation into test mode.

Table 2-17 Test control signals during normal operation

TESTREQA	TESTREQB	TESTACK	Description
0	-	0	Normal operation
1	-	0	Enter test mode request
-	-	1	Test mode entered

During system test the external test pins are used to control the operation of the TIC. The operation of these pins is shown in Table 2-18 on page 2-46.

Table 2-18 Test control signals during test operation

TESTREQA	TESTREQB	TESTACK	Description
-	-	0	Current access incomplete
1	1	1	Address, Control, or Turnaround vector
1	0	1	Write vector
0	1	1	Read vector
0	0	1	Exit test mode

On entry into test mode the TIC indicates that it has switched to the test clock input by asserting the **TESTACK** signal.

2.11.2 Test vector types

There are five types of test vector associated with the test interface:

Address The address for all subsequent read and write transfers is sampled by the TIC.

Write The TIC performs an AHB write cycle, using the write data currently driven onto the external data bus.

Read The TIC performs an AHB read cycle, driving the read data onto the external data bus when it becomes valid.

Control Internal TIC registers are set, that control the types of read and write transfers that are performed.

Turnaround Used between a read cycle and a write cycle to avoid clashes on the external data bus.

The address, control and turnaround vectors are all indicated by the same value on the **TESTREQA** and **TESTREQB** signals. You can use the following rules to determine which type of vector is being applied:

- a read vector, or burst of read vectors, is followed by two turnaround vectors
- when a single address or control vector is applied it is an address vector
- when multiple address or control vectors are applied they are all address vectors, apart from the last which is a control vector.

2.11.3 Control vectors

The control vector is used to determine the types of transfer the TIC can perform, by setting the values of the **HSIZETIC**, **HPROTTIC,** and **HLOCKTIC** AHB master outputs. The default TIC bus master transfer type is:

- for 32-bit transfer width, **HSIZETIC[2:0]** signifies word transfer
- for privileged system access, **HPROTTIC[3:0]** signifies supervisor data access, uncachable and unbufferable.

Bit 0 of the control vector is used to indicate if the control vector is valid. Therefore if a control vector is applied with bit 0 LOW, the vector is ignored and does not update the control information. This mechanism enables address vectors that have bit 0 LOW to be applied for many cycles without updating the control information.

Although the default settings are sufficient for testing many systems, control vectors can be used to change the control signals of the transfer, and can also be used to select whether the TIC must generate fixed addresses or incrementing addresses.

Table 2-19 defines the bit positions of the control vector. The control vector bit definitions are designed to be backwards compatible with earlier versions of the TIC and therefore not all of the control bits are in obvious positions.

Table 2-19 Control vector bit definitions

Bit position	Description
0	Control vector valid
1	Reserved
2	**HSIZETIC[0]**
3	**HSIZETIC[1]**
4	**HLOCKTIC**
5	**HPROTTIC[0]**
6	**HPROTTIC[1]**
7	Address increment enable
8	Reserved
9	**HPROTTIC[2]**
10	**HPROTTIC[3]**

There is no mechanism to control the types of burst that the TIC can perform and only incrementing bursts of an undefined length are supported. The TIC only supports 8-bit, 16-bit, and 32-bit transfers and therefore **HSIZETIC[2]** cannot be altered and is always LOW.

To support burst accesses using the test interface the TIC can support incrementing of the bus address. The TIC increments eight address bits and the address range that can be covered by this incrementor depends on the size of the transfers being performed.

The control vector provides a mechanism to enable and disable the address incrementor within the TIC. This enables burst accesses to incremental addresses, as used for testing internal RAM. Alternatively the address increment can be disabled, so that successive accesses of a burst occur to the same address, as would be required to continually read from a single peripheral register.

The address incrementor is disabled by default and must be enabled using a control vector prior to use.

Note

The control vector is primarily used to change signals that have the same timing as the address bus. However the control vector also enables the lock signal to be changed, which is actually required before the locked transfer commences. If the **HLOCKTIC** signal is used during testing it must be set a transfer before it is required. This difference in timing on the **HLOCKTIC** signal can in some cases cause an additional transfer to be locked both before and after the sequence that must in fact be locked.

2.11.4 Test vector sequences

This section describes:

- *Entering test mode* on page 2-49
- *Write vectors* on page 2-50
- *Read vectors* on page 2-51
- *Control vector* on page 2-52
- *Burst vector* on page 2-52
- *Read-to-write and write-to-read transfers* on page 2-53
- *Exiting test mode* on page 2-54.

Entering test mode

In normal operating mode **TESTREQA** is LOW, indicating that test access is not required. Entering test mode enables test vectors to be applied externally that cause transfers on the internal bus.

Figure 2-28 shows the test start sequence.

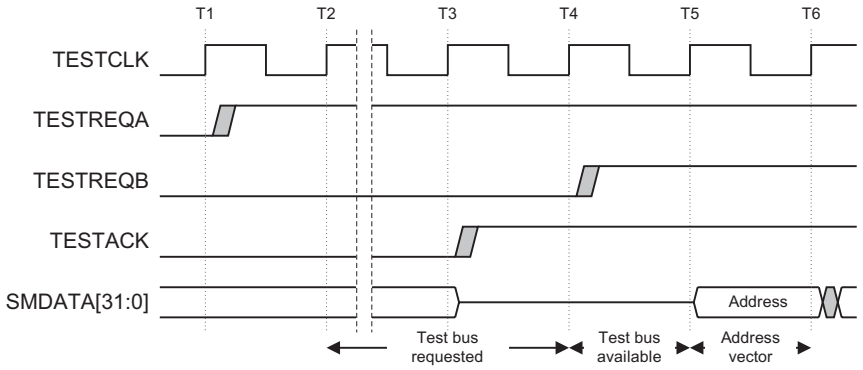


Figure 2-28 Test start sequence

The following sequence, as shown in Figure 2-28, is required to enter test mode:

1. **TESTREQA** is asserted to request test bus access.
2. Test mode is entered when the TIC is granted the internal bus and this is indicated by the assertion of the **TESTACK** signal.
3. When test mode has been entered **TESTREQB** is asserted to initiate an address vector.
4. The TIC does not perform any internal transfers until a valid address vector has been applied.

A synchronous tester is not expected to poll **TESTACK** for the bus. Usually the **TESTREQA** signal is asserted for a minimum number of cycles guaranteed to gain access to the bus (completion of the longest wait-state peripheral access or the maximum number of cycles for all bus masters to have completed their current instruction).

Write vectors

Figure 2-29 shows the sequence of events when applying a set of write test vectors. Initially an address vector is applied and this is followed by a write test vector.

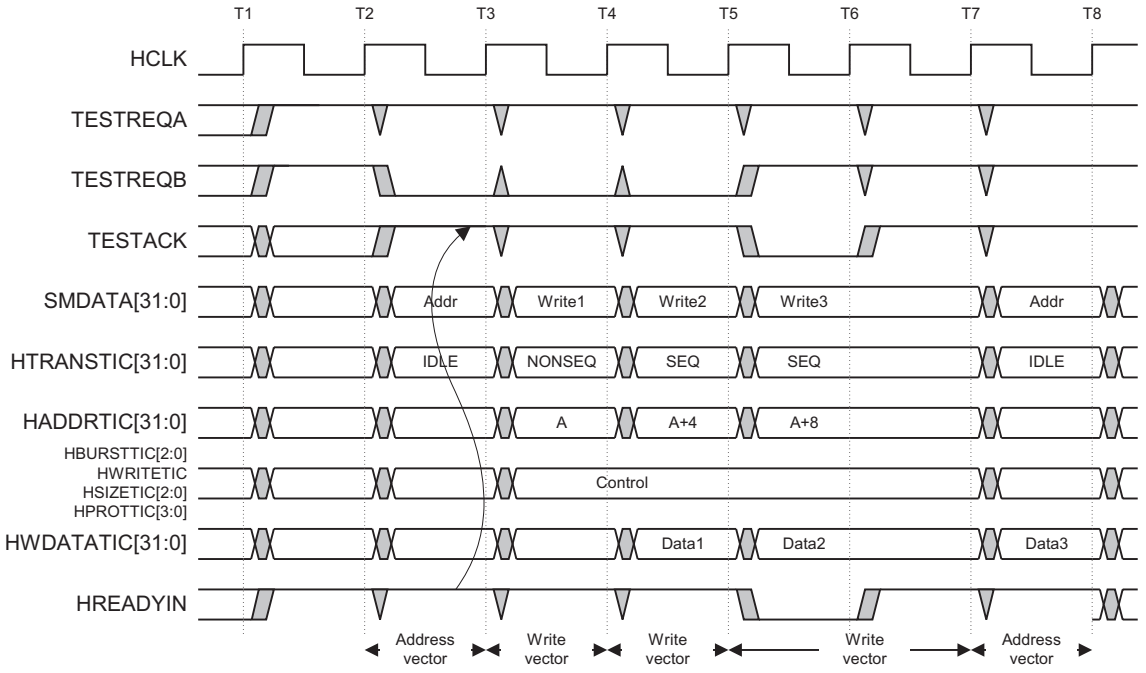


Figure 2-29 Write test vectors

The **TESTREQA** and **TESTREQB** signals are pipelined and are used to indicate what type of vector is applied in the following cycle.

The TIC samples the address and the **TESTREQA** and **TESTREQB** signals at time T3 and following this it can initiate the appropriate transfer on the AHB. In the following cycle the write data is driven onto **SMDATA** and it is then sampled on the following clock edge, T4, and driven onto the internal bus.

If the internal transfer is not able to complete then the **TESTACK** signal is driven LOW and this indicates that the external test vector must be applied for another cycle.

Read vectors

Read transfers are more complex because they require **SMDATA** to be driven in the opposite direction, and therefore additional cycles are required to prevent bus clash when changing between different drivers of **SMDATA**. Figure 2-30 shows a typical test sequence for reads.

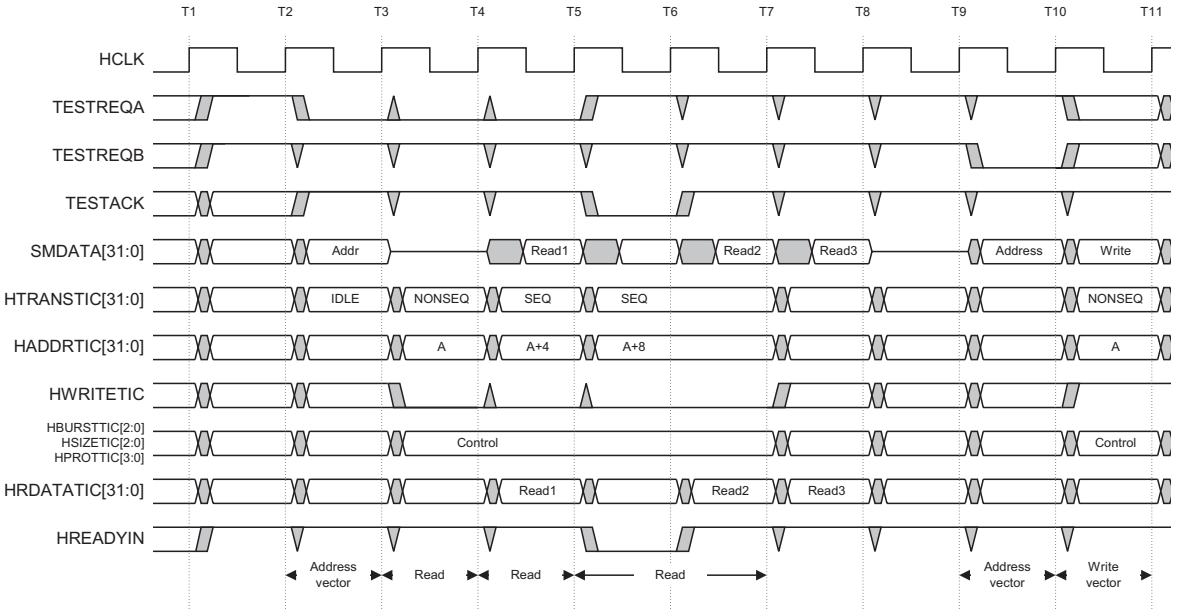


Figure 2-30 Read test vectors

The **TESTREQA** and **TESTREQB** signals are used in the same way as for a write transfer. Initially **TESTREQA** and **TESTREQB** are used to apply an address vector and then in the following cycle they are used to indicate that a read transfer is required. For the first cycle of a read **SMDATA** must be tristated, which ensures that the external tester driving **SMDATA** has an entire cycle to tristate its buffers before the TIC enables the on-chip buffers to drive out the read data.

At the end of a burst of reads it is necessary to enable time for bus turnaround. In this case the TIC must turn off the internal buffers and an entire cycle is permitted before the external test equipment starts to drive **SMDATA**.

The end of a burst of reads is indicated by both **TESTREQA** and **TESTREQB** being HIGH, as for an address vector. They must indicate an address vector for two cycles, which allows for the turnaround cycle at the start of the burst and also the turnaround cycle at the end of the burst.

Control vector

The operation of the TIC can be modified by the use of a control vector. Whenever more than one address vector is applied in succession then the last vector is considered to be a control vector and is not sampled as an address value. Bit 0 of the control vector is used to determine whether or not the control vector must be considered valid, which enables multiple address vectors to be applied without changing the control information. Figure 2-31 shows the process of inserting a control vector.

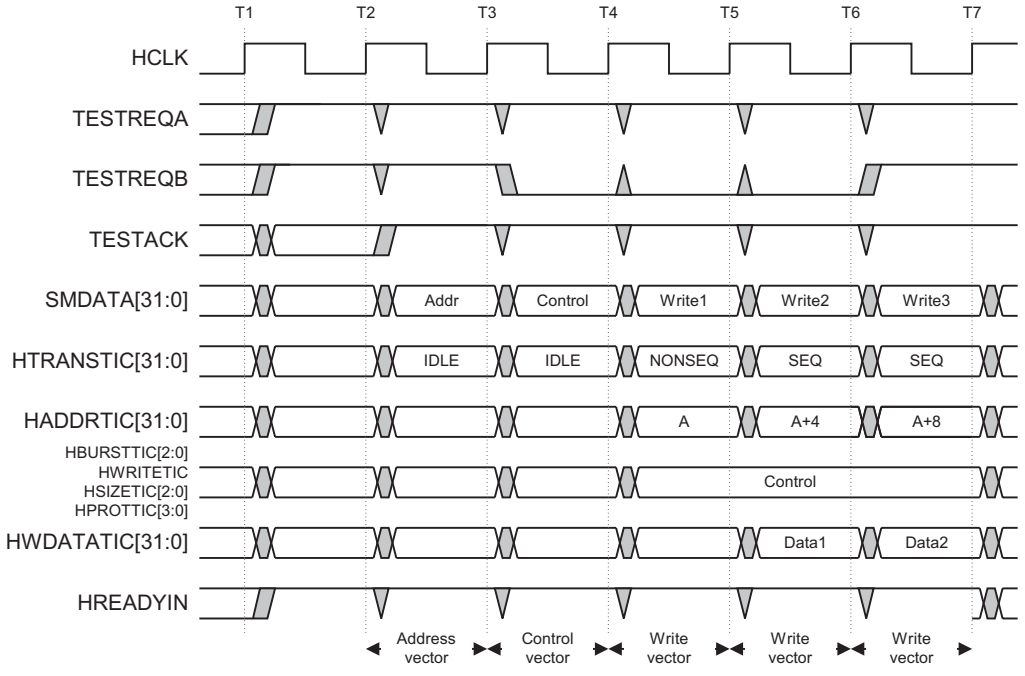


Figure 2-31 Control vector

At time T4, the TIC can determine that **SMDATA** contains a control vector. This is because the previous cycle was an address vector and **TESTREQA** and **TESTREQB** are indicating that the following cycle is either a read or a write and therefore the current cycle must be a control vector.

Burst vector

The examples of read and write transfers in Figure 2-30 on page 2-51 and Figure 2-31 show how additional transfers can be used to form burst transfers on the bus.

The TIC has limited capabilities for burst transfers and can only perform undefined length incrementing bursts. It contains an 8-bit incrementor and if an attempt is made to perform a burst that crosses the incrementor boundary, then the address wraps and the TIC signals the transfer as NONSEQUENTIAL. The exact boundary at which this occurs depends on the size of the transfer. For word transfers the incrementor overflows at 1KB boundaries, for halfword transfers it overflows at 512-byte boundaries and for byte transfers the overflow occurs at 256-byte boundaries.

Read-to-write and write-to-read transfers

It is possible to switch between read transfers and write transfers without applying a new address vector. Usually this is done with the address incrementor disabled, so that both the read transfers and the write transfers are to the same address. It is also possible to do this with the incrementor enabled if the test circumstances require it.

When moving from a read transfer to a write transfer, you must allow two cycles for bus handover, therefore **TESTREQA** and **TESTREQB** must signal an address vector for two cycles after the read. This does not change the address unless it is followed by a third address vector. Figure 2-32 shows the sequence of events.

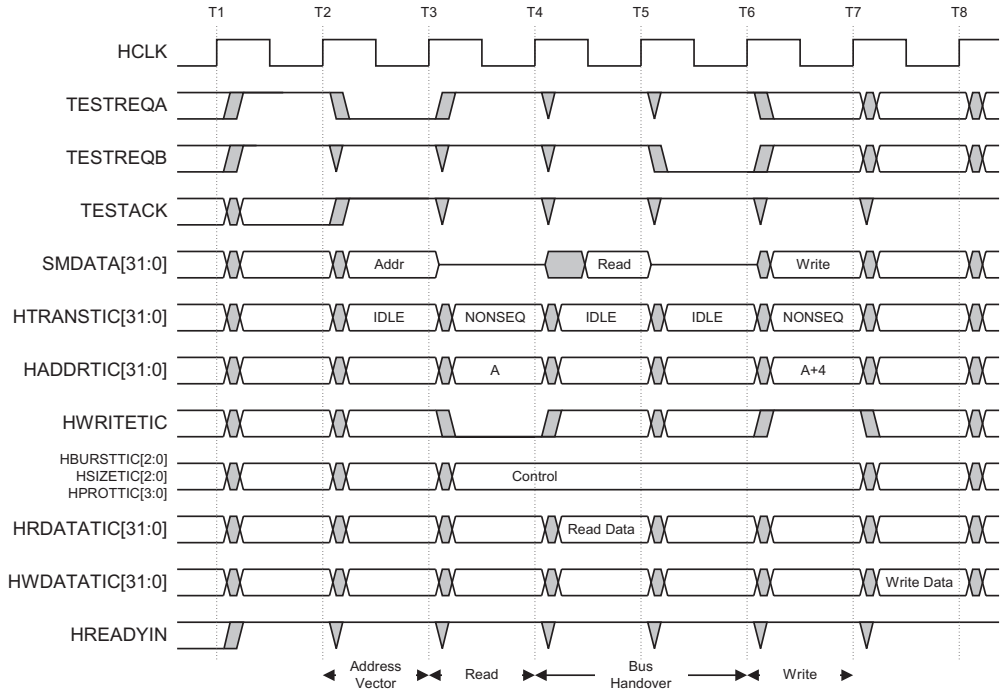


Figure 2-32 Read vector followed by a write vector

Exiting test mode

Test mode is exited using the following sequence:

1. Apply a single cycle address vector, which causes an IDLE cycle internally. This ensures any internal transfers are complete and an Address-Only transfer is performed on the internal bus.
2. **TESTREQA** and **TESTREQB** are both driven LOW to indicate that test mode is to be exited.
3. When the test interface has been configured for normal system operation, **TESTACK** goes LOW to indicate that test mode has been exited.

It is important that test mode can be entered and exited cleanly so that the TIC can be used for diagnostic test during system operation, in addition to during production testing.

2.12 Data bus interface operation

The *Data Bus Interface* (DBI) enables both parts of the SMC and the additional memory controller to share the external address and data buses, by only selecting one device to drive them at a time. The following lines are multiplexed by this block:

- **SMADDR/MCADDR**
- **SMDATAOUT/TBUSOUTEBI/MCDATAOUT**
- **nSMDATAEN/TICREADEBI/MCDATAEN**.

All other signals pass directly to the memory controllers and the test controller, including the external data input.

A simple priority-based request and grant system is used, with the TIC the highest priority device (because it must always be possible to enter test mode), and the PrimeCell SMC the lowest priority device. The bus interface degrants a lower priority device when a higher priority device requests control of the bus, but the higher priority device only gains control of the bus when the lower priority device finishes the current transfer and releases control of the bus. The SMC is granted control of the bus by default, reducing the access time during normal operation.

An example of the timing of the bus interface is shown in Figure 2-33.

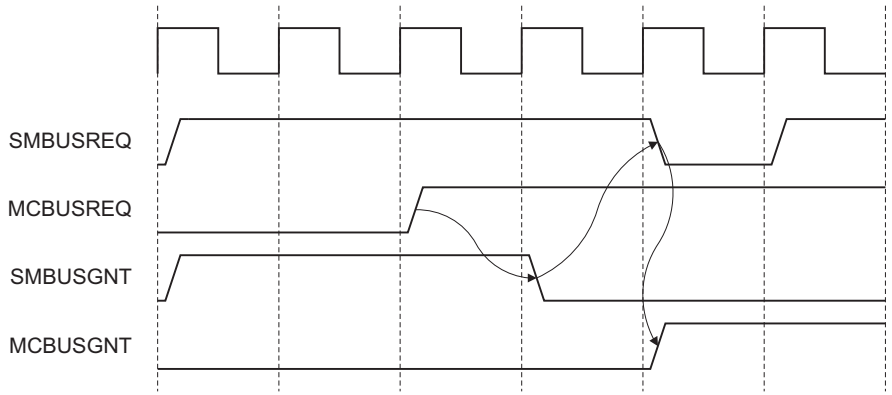


Figure 2-33 Example of bus interface timing

When the SMC requests the bus the other higher priority device is not requesting control of the bus (**MCBUSREQ** deasserted), so the SMC is granted immediately. However, during the course of the SMC transfer, the higher priority device requests the bus (**MCBUSREQ** asserted) and hence the SMC grant line is taken LOW by the DBI in the following cycle. This indicates to the SMC that it must complete the current external transfer and deassert **SMBUSREQ**. When the SMC has reached the end of this transfer it sets the **SMBUSREQ** output LOW, that indicates to the DBI that it can grant control

of the external bus to the other higher priority device. If the SMC still has further transfers to complete, for example it might have terminated a word access from a byte-wide memory after two bytes, then it can reassert the **SMBUSREQ** signal one clock cycle after deasserting it and wait to be regranted before continuing the transfer.

The timing in Figure 2-33 on page 2-55 shows the minimum time it takes for the additional memory controller to gain control of the bus if the SMC is currently granted and performing a transfer. The additional controller must wait a minimum of two cycles between asserting its bus request and the DBI asserting the bus grant. It takes one cycle for the SMC to be degranted, and a minimum of one cycle for the SMC to deassert its bus request output and for the additional controller to gain control of the bus. The maximum time that can occur before gaining control of the external bus depends on the system, because an externally waited SMC transfer has no time limit, and depends on the device driving **CANCELSMWAIT** to set a maximum time limit for a waited transfer.

2.13 Using the SMC with an external bus multiplexor or SDRAM controller

The SMC is designed to enable the use of an external bus multiplexor, instead of the internal DBI, to control the switching of the external data and address buses, for example if the SMC is to be used with an SDRAM controller that requires a bus multiplexor that can handle the necessary synchronization retiming.

The **EXTBUSMUX** input must be tied HIGH so that the internal DBI is bypassed and the external bus multiplexor is used instead. This enables the external request and grant ports instead of the internal DBI connections.

If the external bus multiplexor contains retiming registers on the address output path, for example when used with a synchronous memory controller such as an SDRAM controller, ensure that the WSTWEN values for all writable register banks are set to a minimum of one, and that the WST2 values are also set to a minimum of one for the same banks. This is because the write enable must only be valid after the address output is stable to make sure that the write is to the correct address. This is only required for writes because of the half a clock cycle timing between the address being stable and the write enable being asserted. Because reads use full cycle timing, no special settings are required.

The SMC and TIC bus request and grant ports must be connected to the external bus multiplexor, ensuring that if the TIC is used it is connected to the ports for the highest priority device because it must always be able to enter test mode. The SMC address, data, and data enable ports and the TIC data and data enable ports must also be connected.

Note

The additional memory controller ports of the SMC are not used when the internal DBI is bypassed. Any additional controller must be connected directly to the external bus multiplexor. When the SMC is used with an SDRAM controller, care must be taken to ensure that the SMC does not take control of the bus for longer than the refresh timing of the SDRAM. Because externally waited transfers have undefined lengths, the system must be able to use the **CANCELSMWAIT** signal to cancel any externally waited transfer and enable the SDRAM refresh to occur.

Chapter 3
Programmer's Model

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

3.1 About the programmer's model

The following applies to the registers used in the PrimeCell SMC:

- The base addresses for the PrimeCell SMC bank configuration registers and memory banks can be configured to suit each particular system implementation.
- The base addresses must be configured by constant definitions in the *Hardware Description Language* (HDL) code for the AMBA address decoder.
- The base addresses are not software-programmable. The PrimeCell SMC registers and memory banks have fixed address offsets from the base addresses described above.
- Reserved or unused address locations must not be accessed because this can result in unpredictable behavior of the device.
- Reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.
- All register bits are reset to a logic 0 by a system or power-on 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 the contents of a register and a read returns the contents of the register.

3.2 Summary of registers

All register addresses in the SMC are fixed relative to the SMC base address. Table 3-1 lists the registers in base offset order.

Table 3-1 PrimeCell SMC read/write register summary

Register	Offset from base	Type	Reset value	Function
SMBIDCYR0	0x00	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R0	0x04	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R0	0x08	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR0	0x0C	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.
SMBWSTWENR0	0x10	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR0	0x14	RW	0x80	See *Bank Control Registers 0-7* on page 3-11
SMBSR0	0x18	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBIDCYR1	0x1C	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R1	0x20	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R1	0x24	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR1	0x28	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.
SMBWSTWENR1	0x2C	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR1	0x30	RW	0x00	See *Bank Control Registers 0-7* on page 3-11
SMBSR1	0x34	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBIDCYR2	0x38	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R2	0x3C	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R2	0x40	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR2	0x44	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.

Table 3-1 PrimeCell SMC read/write register summary (continued)

Register	Offset from base	Type	Reset value	Function
SMBWSTWENR2	0x48	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR2	0x4C	RW	0x40	See *Bank Control Registers 0-7* on page 3-11
SMBSR2	0x50	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBIDCYR3	0x54	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R3	0x58	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R3	0x5C	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR3	0x60	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.
SMBWSTWENR3	0x64	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR3	0x68	RW	0x00	See *Bank Control Registers 0-7* on page 3-11
SMBSR3	0x6C	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBIDCYR4	0x70	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R4	0x74	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R4	0x78	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR4	0x7C	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.
SMBWSTWENR4	0x80	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR4	0x84	RW	0x80	See *Bank Control Registers 0-7* on page 3-11
SMBSR4	0x88	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBIDCYR5	0x8C	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R5	0x90	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R5	0x94	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR5	0x98	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.

Table 3-1 PrimeCell SMC read/write register summary (continued)

Register	Offset from base	Type	Reset value	Function
SMBWSTWENR5	0x9C	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR5	0xA0	RW	0x80	See *Bank Control Registers 0-7* on page 3-11
SMBSR5	0xA4	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBIDCYR6	0xA8	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R6	0xAC	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R6	0xB0	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR6	0xB4	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.
SMBWSTWENR6	0xB8	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR6	0xBC	RW	0x40	See *Bank Control Registers 0-7* on page 3-11
SMBSR6	0xC0	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBIDCYR7	0xC4	RW	0xF	See *Bank Idle Cycle Control Registers 0-7* on page 3-7
SMBWST1R7	0xC8	RW	0x1F	See *Bank Wait State 1 Control Registers 0-7* on page 3-8
SMBWST2R7	0xCC	RW	0x1F	See *Bank Wait State 2 Control Registers 0-7* on page 3-9
SMBWSTOENR7	0xD0	RW	0x0	See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-10.
SMBWSTWENR7	0xD4	RW	0x1	See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-10
SMBCR7	0xD8	RW	0x00	See *Bank Control Registers 0-7* on page 3-11
SMBSR7	0xDC	RW	0x0	See *Bank Status Registers 0-7* on page 3-13
SMBEWS	0xE0	RO	0x0	See *External Wait Status Register* on page 3-15
SMCPeriphID0	0xFE0	RO	0x92	See *Peripheral Identification Register 0* on page 3-17
SMCPeriphID1	0xFE4	RO	0x10	See *Peripheral Identification Register 1* on page 3-17
SMCPeriphID2	0xFE8	RO	0x04	See *Peripheral Identification Register 2* on page 3-17
SMCPeriphID3	0xFEC	RO	0x00	See *Peripheral Identification Register 3* on page 3-18

Table 3-1 PrimeCell SMC read/write register summary (continued)

Register	Offset from base	Type	Reset value	Function
SMCPCellID0	0xFF0	RO	0x0D	See *PrimeCell Identification Register 0* on page 3-19
SMCPCellID1	0xFF4	RO	0xF0	See *PrimeCell Identification Register 1* on page 3-19
SMCPCellID2	0xFF8	RO	0x05	See *PrimeCell Identification Register 2* on page 3-19
SMCPCellID3	0xFFC	RO	0xB1	See *PrimeCell Identification Register 3* on page 3-20

3.3 Register descriptions

This section describes the PrimeCell SMC registers. Table 3-1 on page 3-3 and Table 3-2 provide cross references to individual registers.

3.3.1 Bank Idle Cycle Control Registers 0-7

The eight SMBIDCYRx Registers are read and write. They are the SMC Bank Idle Cycle Control Registers that must be programmed for the configuration of memory banks 0-7. Each register is identical in structure.

Figure 3-1 shows the register bit assignments.

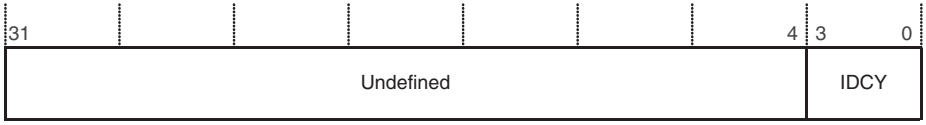


Figure 3-1 SMBIDCYRx Register bit assignments

Table 3-2 lists the register bit assignments.

Table 3-2 SMBIDCYRx Register bit assignments

Bit	Name	Function
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	IDCY	Read/write. Idle or turn-around cycles. This field controls the number of bus turn-around cycles added between read and write accesses, to prevent bus contention on the external memory data bus. Turn around time = IDCY x t_{HCLK}[a] Defaults to 1111 at reset.

a. t_{HCLK} = period of **HCLK**

3.3.2 Bank Wait State 1 Control Registers 0-7

The eight SMBWST1Rx Registers are read and write. They are the SMC Bank Wait State One Control Registers that must be programmed for the configuration of memory banks 0-7. Each register is identical in structure.

Figure 3-2 shows the register bit assignments.

Figure 3-2 SMBWST1Rx Register bit assignments

Table 3-3 lists the register bit assignments.

Table 3-3 SMBWST1Rx Register bit assignments

Bit	Name	Function
[31:5]	-	Reserved, read undefined, do not modify.
[4:0]	WST1	Read/write. Wait State 1: **SRAM and ROM** The WST1 field controls the number of wait states for read accesses. **Burst ROM** The WST1 field controls the number of wait states for the first read access only. Wait state time = WST1 x t_{HCLK}[a] Defaults to 11111 at reset.

a. t_{HCLK} = period of **HCLK**

3.3.3 Bank Wait State 2 Control Registers 0-7

The eight SMBWST2Rx Registers are read and write. They are the SMC Bank Wait State Two Control Registers that must be programmed for the configuration of memory banks 0-7. Each register is identical in structure.

Figure 3-3 shows the register bit assignments.

Figure 3-3 SMBWST2Rx Register bit assignments

Table 3-4 lists the register bit assignments.

Table 3-4 SMBWST2Rx Register bit assignments

Bit	Name	Function
[31:5]	-	Reserved, read undefined, do not modify.
[4:0]	WST2	Read/write. Wait state 2: **SRAM** The WST2 field controls the number of wait states for write accesses, and the external wait assertion timing for writes. This wait state time is WST2 x t_{HCLK}[a] in the case of SRAM. **Burst ROM** The WST2 field controls the number of wait states for the burst read accesses after the first read. This wait state time is WST2 x t_{HCLK} in the case of burst ROM. **ROM** WST2 does not apply to ROM devices. Defaults to 11111 at reset.

a. t_{HCLK} = period of **HCLK**

3.3.4 Bank Output Enable Assertion Delay Control Registers 0-7

The eight SMBWSTOENRx Registers are read and write. They are the SMC Bank Output Enable Assertion Delay Control Registers, that must be programmed for the configuration of memory banks 0-7. Each register is identical in structure.

Figure 3-4 shows the register bit assignments.

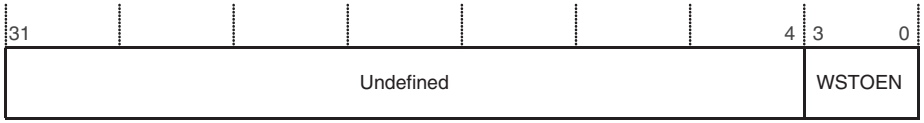


Figure 3-4 SMBWSTOENRx Register bit assignments

Table 3-5 lists the register bit assignments.

Table 3-5 SMBWSTOENRx Register bit assignments

Bit	Name	Function
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	WSTOEN	Read/write. Output enable assertion delay from chip select assertion. Defaults to 0000 at reset.

3.3.5 Bank Write Enable Assertion Delay Control Registers 0-7

The eight SMBWSTWENRx Registers are read and write. They are+ the SMC Bank Write Enable Assertion Delay Control Registers that must be programmed for the configuration of memory banks 0-7. Each register is identical in structure.

Figure 3-5 shows the register bit assignments.

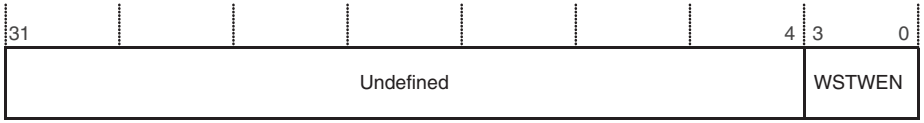


Figure 3-5 SMBWSTWENRx Register bit assignments

Table 3-6 lists the register bit assignments.

Table 3-6 SMBWSTWENRx Register bit assignments

Bit	Name	Function
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	WSTWEN	Read/write. Write enable assertion delay from chip select assertion. Defaults to 0000 at reset.

3.3.6 Bank Control Registers 0-7

The eight SMBCRx Registers are read and write. They are the SMC Bank Control Registers that must be programmed for the configuration of memory banks 0-7. Each register is identical in structure.

Table 3-7 shows the memory bank default external memory width at reset.

Table 3-7 PrimeCell SMC reset default memory width

PrimeCell SMC memory bank	Default memory width
Bank 0	32-bit
Bank 1	8-bit
Bank 2	16-bit
Bank 3	8-bit
Bank 4	32-bit
Bank 5	32-bit
Bank 6	16-bit
Bank 7	8-bit or **SMMWCS7[1:0]** at reset. If the system boots from a Bank7 external ROM device, the memory width for that bank must be configured using the external control pins **SMMWCS7[1:0]** and **SMRBLECS7** to provide the correct default external memory width at reset.

Figure 3-6 on page 3-12 shows the register bit assignments.

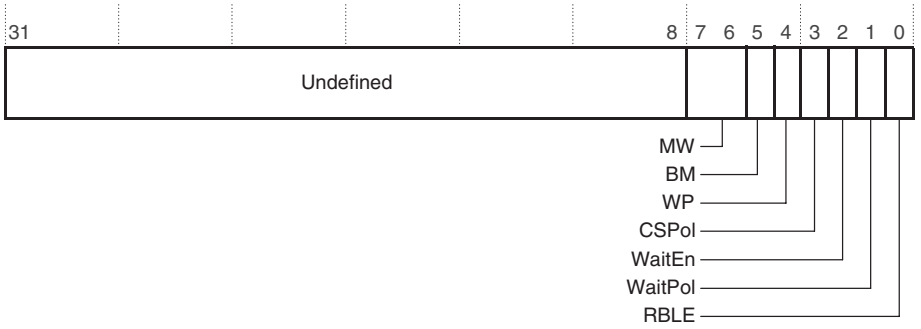


Figure 3-6 SMBCRx Register bit assignments

Table 3-8 lists the register bit assignments.

Table 3-8 SMBCRx Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined, do not modify.
[7:6]	MW	Read/write. Memory Width: 00 = 8-bit 01 = 16-bit 10 = 32-bit 11 = Reserved. Different default reset values for each bank, see Table 3-7 on page 3-11.
[5]	BM	Read/write. Burst mode: 0 = Nonburst memory devices (default at reset). 1 = Burst ROM memory.
[4]	WP	Read/write. Write protect: 0 = No write protection, for example, SRAM or write enabled Flash (default at reset). 1 = Device is write protected, for example, ROM, burst ROM, read only Flash, or SRAM.
[3]	CSPol	Read/write. Chip select polarity bit indication for each bank: 0 = Active LOW **SMCS** (default at reset) 1 = Active HIGH **SMCS**. Set this bit to the correct value if the device has the output enable lines tied off so that the outputs are always driven. If the incorrect polarity chip select is used, the device can continuously drive data out.

Table 3-8 SMBCRx Register bit assignments (continued)

Bit	Name	Function
[2]	WaitEn	Read/write. External memory controller wait signal enable: 0 = The PrimeCell SMC is not controlled by the external wait signal (default at reset). 1 = The PrimeCell SMC looks for the external wait input signal, **SMWAIT.**
[1]	WaitPol	Read/write. Polarity of the external wait input for activation: 0 = The **SMWAIT** signal is active LOW (default at reset). 1 = The **SMWAIT** signal is active HIGH.
[0]	RBLE	Read/write. Read byte lane enable: 0 = **nSMBLS**[3:0] all deasserted HIGH during system reads from external memory. This is for 8-bit devices where the byte lane enable is connected to the write enable pin so must be deasserted during a read (default at reset). The **nSMBLS** signals are used as write enables in this configuration. 1 = **nSMBLS**[3:0] all asserted LOW during system reads from external memory. This is for 16 or 32-bit devices where the separate write enable signal is used and the byte lane selects must be held asserted during a read. The **nSMWEN** signal is used as the write enable in this configuration. If the system boots from a Bank 7 external ROM device, the RBLE setting for that bank must be configured using the external control pin SMRBLECS7 to provide the correct default read byte lane enable setting at reset.

3.3.7 Bank Status Registers 0-7

The eight SMBSRx Registers are read and write. They are the SMC Bank Status Registers. These registers indicate the state of various conditions, for example:

- errors on write protected regions
- time outs because of external waits
- any bus transfer errors.

The software can clear each error condition by writing 1 to the appropriate bit position.

Figure 3-7 on page 3-14 shows the register bit assignments.

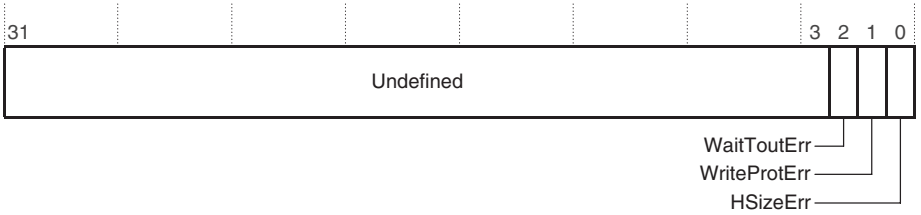


Figure 3-7 SMBSRx Register bit assignments

Table 3-9 lists the register bit assignments.

Table 3-9 SMBSRx Register bit assignments

Bit	Name	Function
[31:3]	-	Reserved, read undefined, do not modify.
[2]	WaitToutErr	Read/write. External wait timeout error flag. Reading from this bit: 0 = No error (default at reset). 1 = External wait timeout error. Writing to this bit: 1 = Clears the wait timeout error status flag. 0 = Has no effect.
[1]	WriteProtErr	Read/write. Write protect error status flag: Reading from this bit: 0 = No error (default at reset). 1 = Write protect error. Writing to this bit: 1 = Clears the write protect error status flag. 0 = Has no effect.
[0]	HSizeErr	Read/write. Bus transfer size error status flag: Reading from this bit: 0 = No error (default at reset). 1 = Bus transfer size error. Writing to this bit: 1 = Clears the bus transfer size error status flag. 0 = Has no effect.

3.3.8 External Wait Status Register

The SMBEWS Register are read-only. They show the current status of the external wait during an externally waited transfer.

Figure 3-8 shows the register bit assignments.

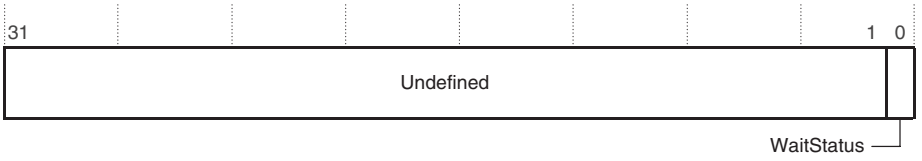


Figure 3-8 SMBEWS Register bit assignments

Table 3-10 lists the register bit assignments.

Table 3-10 SMBEWS Register bit assignments

Bit	Name	Function
[31:1]	-	Reserved, read undefined.
[0]	WaitStatus	Read. External wait status: 0 = **SMWAIT** deasserted. 1 = **SMWAIT** asserted. After an externally waited transfer that was terminated early, this bit value can be used to detect when **SMWAIT** is deasserted. At all other times this bit reads zero.

3.3.9 Peripheral Identification Registers 0-3

The SMCPeriphID0-3 Registers are read-only. They are four 8-bit registers that span address locations `0xFE0-0xFEC`. The registers can conceptually be treated as a single 32-bit register.

These read-only registers provide the following options of the peripheral:

Table 3-11 SMCPeriphID Register options

Bit	Function
[11:0]	Identifies the part number of the peripheral. The three-digit product code 092 is used for the PrimeCell SMC.
[19:12]	Identifies the designer. ARM Ltd. is 0x41, ASCII A.
[23:20]	Identifies the peripheral. The revision number starts from 0.
[31:24]	Configures the peripheral. The configuration value is 0.

Figure 3-9 shows the bit assignments for the SMCPeriphID Registers.

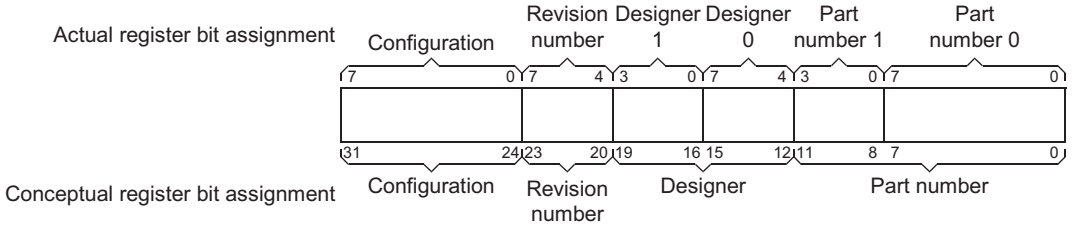


Figure 3-9 SMCPeriphID Register bit assignments

Note

When you design a systems memory map you must remember that the register has a 4KB memory footprint. The 4-bit revision number is implemented by instantiating a component called `RevisionAnd` four times with its inputs tied off as appropriate, and the output sent to the read multiplexor. All memory accesses to the peripheral identification registers must be 32-bit, using the `LDR` and `STR` instructions.

Peripheral Identification Register 0

The SMCPeriphID0 Register is read-only. It is hard-coded and the fields in the registers determine the reset value.

Table 3-12 lists the register bit assignments.

Table 3-12 SMCPeriphID0 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:0]	Partnumber0	These bits read back as `0x92`.

Peripheral Identification Register 1

The SMCPeriphID1 Register is read-only. It is hard-coded and the fields in the registers determine the reset value.

Table 3-13lists the register bit assignments.

Table 3-13 SMCPeriphID1 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:4]	Designer0	These bits read back as `0x0`.
[3:0]	Partnumber1	These bits read back as `0x1`.

Peripheral Identification Register 2

The SMCPeriphID2 Register is hard coded and the fields in the registers determine the reset value.

Table 3-14 shows the bit assignments of the SMCPeriphID2 Register.

Table 3-14 SMCPeriphID2 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:4]	Revision	These bits read back as `0x0`.
[3:0]	Designer1	These bits read back as `0x4`.

Peripheral Identification Register 3

The SMCPeriphID3 Register is read-only. It is hard-coded and the fields in the registers determine the reset value.

Table 3-15 lists the register bit assignments.

Table 3-15 SMCPeriphID3 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:0]	Configuration	These bits read back as `0x0`.

3.3.10 PrimeCell Identification Registers 0-3

The SMCPCellID0-3 Registers are read-only. They are four 8-bit registers, that span address locations `0xFF0-0xFFC`. The read-only registers can conceptually be treated as a single 32-bit register. The register is used as a standard cross-peripheral identification system.

Figure 3-10 lists the register bit assignments.

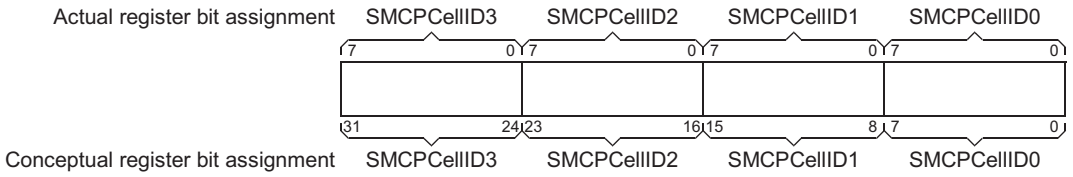


Figure 3-10 SMCPCellID Register bit assignments

PrimeCell Identification Register 0

The SMCPCellID0 Register is a read register. It is hard-coded and the fields in the registers determine the reset value.

Table 3-16 lists the register bit assignments.

Table 3-16 SMCPCellID0 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:0]	SMCPCellID0	These bits read back as `0x0D`.

PrimeCell Identification Register 1

The SMCPCellID1 Register is a read register. It is hard-coded and the fields in the registers determine the reset value.

Table 3-17 lists the register bit assignments.

Table 3-17 SMCPCellID1 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:0]	SMCPCellID1	These bits read back as `0xF0`.

PrimeCell Identification Register 2

The SMCPCellID2 Register is a read register. It is hard-coded and the fields in the registers determine the reset value.

Table 3-18 lists the register bit assignments.

Table 3-18 SMCPCellID2 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:0]	SMCPCellID2	These bits read back as `0x05`.

PrimeCell Identification Register 3

The SMCPCellID3 Register is a read register. It is hard-coded and the fields in the registers determine the reset value.

Table 3-19 lists the register bit assignments.

Table 3-19 SMCPCellID3 Register bit assignments

Bit	Name	Function
[31:8]	-	Reserved, read undefined.
[7:0]	SMCPCellID3	These bits read back as `0xB1`.

Chapter 4
Programmer's Model for Test

This chapter describes the additional logic for integration testing. It contains the following sections:

4.1 Scan testing

The SMC has been designed to simplify:

- the insertion of scan test cells
- the use of *Automatic Test Pattern Generation* (ATPG).

This is the recommended method of manufacturing test.

Appendix A
Signal Descriptions

This appendix describes the signals that interface with the SMC. It contains the following sections:

A.1 AMBA AHB interface signals

Table A-1 describes the common AMBA AHB signals.

Table A-1 Common AMBA AHB signals

Signal name	Type	Source/ destination	Description
nHCLK	Input	Clock control	Inverted bus clock input. It is used only for the generation of the write enable output.
HCLK	Input	Clock control	Bus clock input, that times all bus transfers. All signal timings are related to the rising edge.
HRESETn	Input	Reset controller	Bus reset input, active LOW, used to reset the system and the bus when asserted LOW.
HREADYIN	Input	Other AHB slaves	Transfer completed input. When HIGH the **HREADYIN** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.

A.2 AMBA AHB slave interface signals

Table A-2 describes the AMBA AHB slave interface signals.

Table A-2 AMBA AHB slave interface signals

Signal name	Type	Source/ destination	Description
HADDR[28:0]	Input	AMBA AHB master	System address bus, least significant 29 bits, driven by the active bus master.
HTRANS[1:0]	Input	AMBA AHB master	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY.
HWRITE	Input	AMBA AHB master	Transfer direction signal. When HIGH, this signal indicates a write to the SMC and when LOW, a read from the SMC block.
HSIZE[2:0]	Input	AMBA AHB master	Transfer size signal. This signal indicates the size of the current transfer, which can be byte (8-bit), halfword (16-bit), or word (32-bit).
HBURST[2:0]	Input	AMBA AHB master	Indicates if the transfer forms part of a burst. Four, eight, and 16 beat bursts are supported and the burst can either be incrementing or wrapping.
HWDATA[31:0]	Input	AMBA AHB master	Write data bus, used to transfer data to this block.
HSELSMC	Input	AMBA AHB decoder	Slave select signal for SMC memory banks.
HSELREG	Input	AMBA AHB decoder	Slave select signal for SMC SMBCRx and SMBSRx Registers.
HRDATA[31:0]	Output	AMBA AHB slave	Read data bus, used to read data from this block.
HREADYOUT	Output	AMBA AHB slave	Transfer done output. When HIGH indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESP[1:0]	Output	AMBA AHB slave	The transfer response provides additional information on the status of a transfer. Two different responses are provided, OKAY and ERROR.

A.3 AMBA AHB master interface signals

Table A-3 describes the AMBA AHB master interface signals.

Table A-3 AMBA AHB master interface signals

Signal name	Type	Source/ destination	Description
HRESPTIC[1:0]	Input	AMBA AHB slave	The transfer response provides additional information on the status of a transfer. The TIC supports both SPLIT and RETRY responses.
HRDATATIC[31:0]	Input	AMBA AHB slave	The read data bus is used to transfer data from bus slaves to the bus master during test read operations.
HGRANTTIC	Input	AMBA AHB arbiter	This signal indicates that the TIC is currently the highest priority master. Ownership of the address or control signals changes at the end of the transfer when **HREADYIN** is HIGH.
HADDRTIC[31:0]	Output	AMBA AHB slave	The 32-bit system address bus.
HTRANSTIC[1:0]	Output	AMBA AHB slave	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, or IDLE. The TIC does not use the BUSY transfer type.
HWRITETIC	Output	AMBA AHB slave	Transfer direction signal. When HIGH, this signal indicates a write to the PrimeCell AHB SMC and when LOW, a read from the PrimeCell AHB SMC block.
HSIZETIC[2:0]	Output	AMBA AHB slave	Transfer size signal. This signal indicates the size of the current transfer, which can be byte (8-bit), halfword (16-bit) or word (32-bit). The TIC does not support larger transfer sizes.
HBURSTTIC[2:0]	Output	AMBA AHB slave	Indicates if the transfer forms part of a burst. The TIC always performs incrementing bursts of unspecified length.
HPROTTIC[3:0]	Output	AMBA AHB slave	The protection control signals indicate if the transfer is an opcode fetch or data access, and if the transfer is a Supervisor mode access or User mode access. These signals can also indicate whether the current access is cacheable or unbufferable.

Table A-3 AMBA AHB master interface signals (continued)

Signal name	Type	Source/ destination	Description
HWDATATIC[31:0]	Output	AMBA AHB slave	The write data bus is used to transfer data from the master to bus slaves during write operations. A minimum data bus width of 32 bits is recommended, however this can easily be extended to enable for higher bandwidth operation.
HBUSREQTIC	Output	AMBA AHB arbiter	A signal from the TIC to the bus arbiter that indicates that it requires the bus.
HLOCKTIC	Output	AMBA AHB arbiter	When HIGH this signal indicates that the TIC requires locked access to the bus and no other master must be granted the bus until this signal is LOW.

A.4 Internal signals

Table A-4 describes the internal signals.

Table A-4 Internal signal descriptions

Signal name	Type	Source/ destination	Description
BIGENDIAN	Input	System	This static configuration bit indicates the type of endianness of the memory system: 0 = little-endian 1 = big-endian.
EXTBUSMUX	Input	System	This static configuration bit indicates if the internal bus multiplexor (DBI) is used, or if an external bus multiplexor (for example EBI) is used instead: 0 = internal bus multiplexor 1 = external bus multiplexor
MCBUSREQ	Input	Additional memory controller	Bus request signal, indicates that the additional memory controller block has requested use of the external bus.
MCADDR[25:0]	Input	Additional memory controller	Address output to external bus.
MCDATAOUT[31:0]	Input	Additional memory controller	Data output, used to write data from the additional memory controller to the external data bus.
MCDATAEN[3:0]	Input	Additional memory controller	Tristate input/output pad enable for the byte lanes of the external memory data bus **MCDATAOUT[31:0]**. Enables the byte lanes [31:24], [23:16], [15:8], and [7:0] of the data bus independently.
MCBUSGNT	Output	Additional memory controller	Bus grant signal, indicates that the additional memory controller is granted control of the external bus.
SMBUSREQEBI	Output	External bus multiplexor	Bus request signal, indicates that the SMC has requested use of the external bus. Only used when **EXTBUSMUX** tied to one.
SMBUSGNTEBI	Input	External bus multiplexor	Bus grant signal, indicates that the SMC is granted control of the external bus. Only used when **EXTBUSMUX** tied to one.

Table A-4 Internal signal descriptions (continued)

Signal name	Type	Source/ destination	Description
TICBUSREQEBI	Output	External bus multiplexor	Bus request signal, indicates that the TIC has requested use of the external bus. Only used when **EXTBUSMUX** tied to one.
TICBUSGNTEBI	Input	External bus multiplexor	Bus grant signal, indicates that the TIC is granted control of the external bus. Only used when **EXTBUSMUX** tied to one.
TICREADEBI	Output	External bus multiplexor	Data bus enable for **TBUSOUTEBI**. Only used when **EXTBUSMUX** tied to one.
TBUSOUTEBI[31:0]	Output	External bus multiplexor	TIC data bus output used during reads from the external tester. Only used when **EXTBUSMUX** tied to one.
REMAP	Input	System	Indicates the state of the memory map: 0 = reset memory map (SMCS7 mapped to SMCS0) 1 = normal memory map.
SCANENABLE	Input	System	Dummy pin used as a dedicated scan enable input.
SCANINnHCLK	Input	System	Dummy pin used as a dedicated **nHCLK** scan chain input.
SCANINHCLK	Input	System	Dummy pin used as a dedicated **HCLK** scan chain input.
SCANOUTnHCLK	Output	System	Dummy pin used as a dedicated **nHCLK** scan chain output.
SCANOUTHCLK	Output	System	Dummy pin used as a dedicated **HCLK** scan chain output.

A.5 Input/output pad signals

Table A-5 describes the signals to the input/output pads.

Table A-5 Input/output pad signals

Signal name	Type	Source/ destination	Description
SMMWCS7[1:0]	Input	Input pad	These static configuration bits indicate the memory width used for boot memory bank seven: 00 = 8-bit 01 = 16-bit 10 = 32-bit 11 = no change.
SMWAIT	Input	Input pad	Wait mode input from external memory controller. Active HIGH, or active LOW, (default), as programmed in the SMC Control Registers for each bank.
CANCELSMWAIT	Input	Input pad	This signal enables the system to recover from an externally waited transfer that takes longer than expected to finish. Active HIGH.
SMDATAIN[31:0]	Input	Input pad	External input data bus used to read data from memory bank.
SMDATAOUT[31:0]	Input	Input pad	External output data used to write data from SMC to memory bank.
SMRBLECS7	Input	Input pad	This static configuration bit indicates the memory device read byte lane setting used for boot memory bank seven: 0 = 8-bit, or devices using the byte lane enable as write enable. 1 = 16 or 32-bit, or devices with separate byte lane and write enable inputs.
SMADDR[25:0]	Output	Output pad	External memory address bus, to external memory banks.
SMCS[7:0]	Output	Output pad	Chip select for external memory banks 7-0, default active LOW.
nSMDATAEN[3:0]	Output	Output pad	Tristate input/output pad enable for the byte lanes of the external memory data bus **SMDATA[31:0]**, active LOW. Enables the byte lanes [31:24], [23:16], [15:8], and [7:0] of the data bus independently.
nSMWEN	Output	Output pad	Write enable for the external memory banks, active LOW.

Table A-5 Input/output pad signals (continued)

Signal name	Type	Source/ destination	Description
nSMBLS[3:0]	Output	Output pad	Byte lane select signals, active LOW. The signals **nSMBLS[3:0]** select byte lanes [31:24], [23:16], [15:8], [7:0] on the data bus.
nSMOEN	Output	Output pad	Output enable for external memory banks, active LOW.
TESTREQA	Input	Input pad	This is the Test Bus Request A input signal and is required as a dedicated device pin. During normal system operation the **TESTREQA** signal is used to request entry into the test mode. During test **TESTREQA** is used, in combination with **TESTREQB**, to indicate the type of test vector that is applied in the following cycle.
TESTREQB	Input	Input pad	During test this signal is used, in combination with **TESTREQA**, to indicate the type of test vector that is to be applied in the following cycle.
TESTACK	Output	Output pad	The test bus acknowledge signal gives external indication that the TIC is granted and also indicates when a test access is complete. When **TESTACK** is LOW, the current test vector must be extended until **TESTACK** becomes HIGH.

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 High-performance Bus (AHB)
The AMBA Advanced High-performance Bus system connects embedded processors such as an ARM core to high-performance peripherals, DMA controllers, on-chip memory, and interfaces. It is a high-speed, high-bandwidth bus that supports multi-master bus management to maximize system performance.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)
The ARM open standard for on-chip buses. AHB conforms to this standard.

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.

Advanced Peripheral Bus (APB)
The AMBA Advanced Peripheral Bus is a simpler bus protocol than AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

See also Advanced High-performance Bus.

AHB *See* Advanced High-performance Bus.

Aligned Aligned data items are stored so that their address is divisible by the highest power of two that divides their size. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively. Other related terms are defined similarly.

AMBA *See* Advanced Microcontroller Bus Architecture.

APB *See* Advanced Peripheral Bus.

Application Specific Integrated Circuit (ASIC)
An integrated circuit that has been designed to perform a specific application function. It can be custom-built or mass-produced.

ASIC *See* Application Specific Integrated Circuit.

Automatic Test Pattern Generation (ATPG)
The process of automatically generating manufacturing test vectors for an ASIC design, using a specialized software tool.

Big-endian Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.

See also Little-endian and Endianness.

Big-endian memory Memory in which:- a byte or halfword at a word-aligned address is the most significant byte or halfword within the word at that address - a byte at a halfword-aligned address is the most significant byte within the halfword at that address.

See also Little-endian memory.

Burst A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AHB buses are controlled using the **HBURST** signals to specify if transfers are single, 4-beat, 8-beat, or 16-beat bursts, and to specify how the addresses are incremented.

See also Beat.

Byte An 8-bit data item.

Cold reset Also known as power-on reset. Starting the processor by turning power on. Turning power off and then back on again clears main memory and many internal settings. Some program failures can lock up the processor and require a cold reset to enable the system to be used again. In other cases, only a warm reset is required.

See also Warm reset.

DNM *See* Do Not Modify.

Do Not Modify (DNM)

In Do Not Modify fields, the value must not be altered by software. DNM fields read as Unpredictable values, and must only be written with the same value read from the same field on the same processor. Throughout this manual, DNM fields are sometimes followed by RAZ or RAO in parentheses to show which way the bits should read for future compatibility, but programmers must not rely on this behavior.

Endianness Byte ordering. The scheme that determines the order in which successive bytes of a data word are stored in memory. An aspect of the system's memory mapping.

See also Little-endian and Big-endian

Halfword A 16-bit data item.

Little-endian Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory.

See also Big-endian and Endianness.

Little-endian memory

Memory in which: - a byte or halfword at a word-aligned address is the least significant byte or halfword within the word at that address - a byte at a halfword-aligned address is the least significant byte within the halfword at that address.

See also Big-endian memory.

Microprocessor *See* Processor.

Power-on reset *See* Cold reset.

Processor A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

Remapping Changing the address of physical memory or devices after the application has started executing. This is typically done to allow RAM to replace ROM when the initialization has been completed.

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 not zero. 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 0 and read as 0.

Warm reset Also known as a core reset. Initializes the majority of the processor excluding the debug controller and debug logic. This type of reset is useful if you are using the debugging features of a processor.

Word A 32-bit data item.

Index

R

S

T

W

