# PrimeCell™ Synchronous Static Memory Controller (PL093)

**Revision: r0p4**

## Technical Reference Manual

ARM®


ARM DDI 0236H

# PrimeCell Synchronous Static Memory Controller (PL093)
## Technical Reference Manual



### Release Information

**Change history**

| Date | Issue | Change |
|---|---|---|
| 7 December 2001 | A | First release. |
| 14 June 2002 | B | Changes to signals and incorporation of errata. |
| 26 March 2003 | C | Update to r0p1 and incorporation of errata. |
| 20 August 2003 | D | Update to include errata, corrections, and additions. |
| 27 January 2004 | E | Update to r0p2 and incorporation of errata. |
| 07 June 2004 | F | Update to r0p3. Corrections and addition of note. |
| 06 October 2004 | G | Update to include errata and corrections. |
| 24 January 2005 | H | Update to r0p4 and incorporation of errata. |

### Proprietary Notice



### Confidentiality Status

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

### Product Status

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

### Web Address

http://www.arm.com

# Contents
# PrimeCell Synchronous Static Memory Controller (PL093) Technical Reference Manual

# List of Tables
# PrimeCell Synchronous Static Memory Controller (PL093) Technical Reference Manual

# List of Figures
# PrimeCell Synchronous Static Memory Controller (PL093) Technical Reference Manual

# Preface

This preface introduces the *PrimeCell Synchronous Static Memory Controller (PL093) Revision r0p4 Technical Reference Manual*. It contains the following sections:

- *About this manual* on page xii
- *Feedback* on page xvi.

## About this manual

This is the Technical Reference Manual for the PrimeCell *Synchronous Static Memory Controller* (SSMC).

### 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 SSMC architecture. The SSMC 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, appendix and glossary:

**Chapter 1 *Introduction***

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

**Chapter 2 *Functional Overview***

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

**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 SSMC 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*
- *Signals* on page xiv
- *Numbering* on page xv.

### Typographical

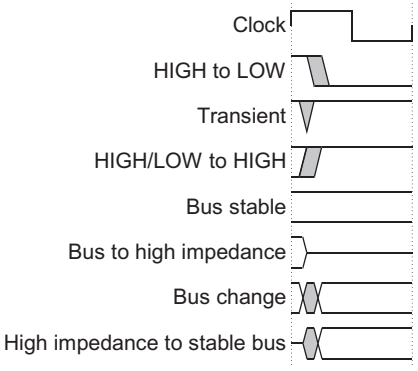
The typographical conventions are:

| | |
|---|---|
| ***italic*** | Highlights important notes, introduces special terminology, denotes internal cross-references, and citations. |
| **bold** | Highlights interface elements, such as menu names. Denotes 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:<br>• `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`<br>• The Opcode_2 value selects which register is accessed. |

### Timing diagrams

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

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




**Key to timing diagram conventions**

## 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 AHB signals.

**Prefix n** Denotes active-LOW signals except in the case of AXI, AHB, or *Advanced Peripheral Bus* (APB) reset signals.

**Prefix P** Denotes APB signals.

**Prefix R** Denotes AXI read channel signals.

**Prefix W** Denotes AXI write channel signals.

**Suffix n** Denotes AXI, AHB, and APB reset 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.

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 PrimeCell SSMC. See the following document for other relevant information:

- *AMBA® Specification (Rev 2.0)* (ARM IHI 0011).

## Feedback

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

### Feedback on this product

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 manual

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

- the title
- the number
- the relevant 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 SSMC and contains the following sections:

## 1.1 About the SSMC

The SSMC is an *Advanced Microcontroller Bus Architecture* (AMBA 2) compliant System-on-Chip peripheral that is developed, tested, and licensed by ARM.

The SSMC 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 aim of reducing time-to-market for ASIC development. It contains an AHB *Test Interface Controller* (TIC) master block that you can use to test the system using externally applied TIC vectors.

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

For more information on AMBA 2 see the *AMBA Specification (Rev 2.0)*.

### 1.1.1 Features

The SSMC 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 2 AHB on-chip bus systems.

The SSMC supports:

- asynchronous static memory-mapped devices including RAM, ROM, and flash
- synchronous static memory-mapped devices including synchronous burst flash
- asynchronous page mode read operation in non-clocked memory subsystems
- asynchronous burst mode read access to 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
- support for interfacing to another memory controller using an *External Bus Interface (*EBI)
- multiple memory clock frequencies available, **HCLK**, **HCLK/**2, and **HCLK/**3
- eight word, 32-bit, wrapping reads from 16-bit or 32-bit memory.

**Note**

The SSMC supports wrapping reads, but does not support wrapping writes.

### 1.1.2 Programmable parameters

You can program the following key parameters each memory bank:

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

Figure 1-1 on page 1-4 shows a diagram of a typical AHB-based microcontroller system. The SSMC has three AHB connections, and two slave interfaces for the SSMC and the master interface for the TIC.

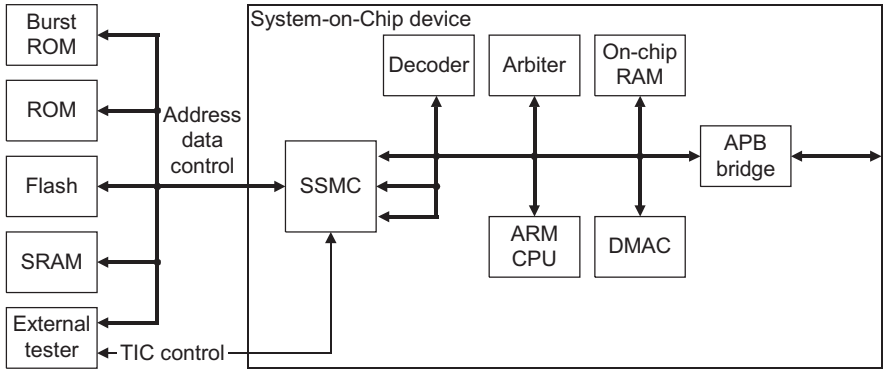



**Figure 1-1 Typical AHB-based microcontroller system**

The SSMC also supports the connection of a synchronous AHB memory controller, such as an SDRAM controller. Figure 1-2 shows this.

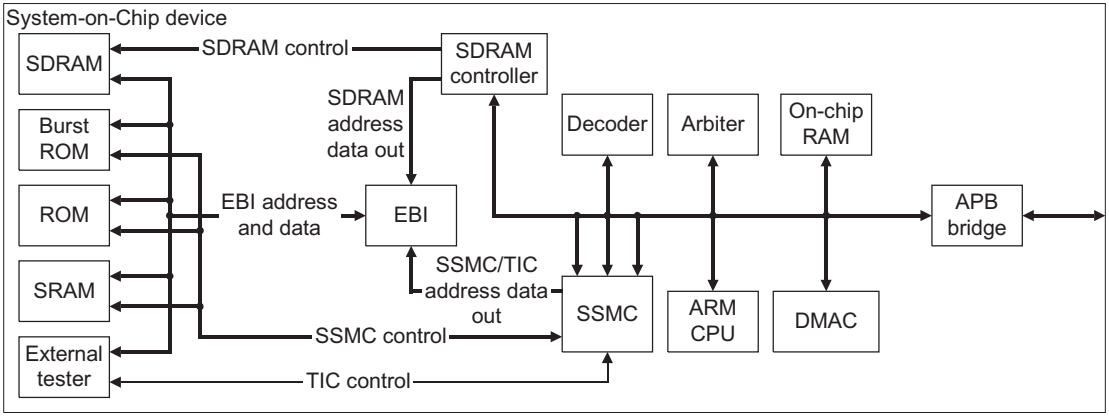



**Figure 1-2 Typical AHB-based microcontroller system using an SDRAM controller**

Figure 1-3 on page 1-5 shows the input and output connections for the SSMC.

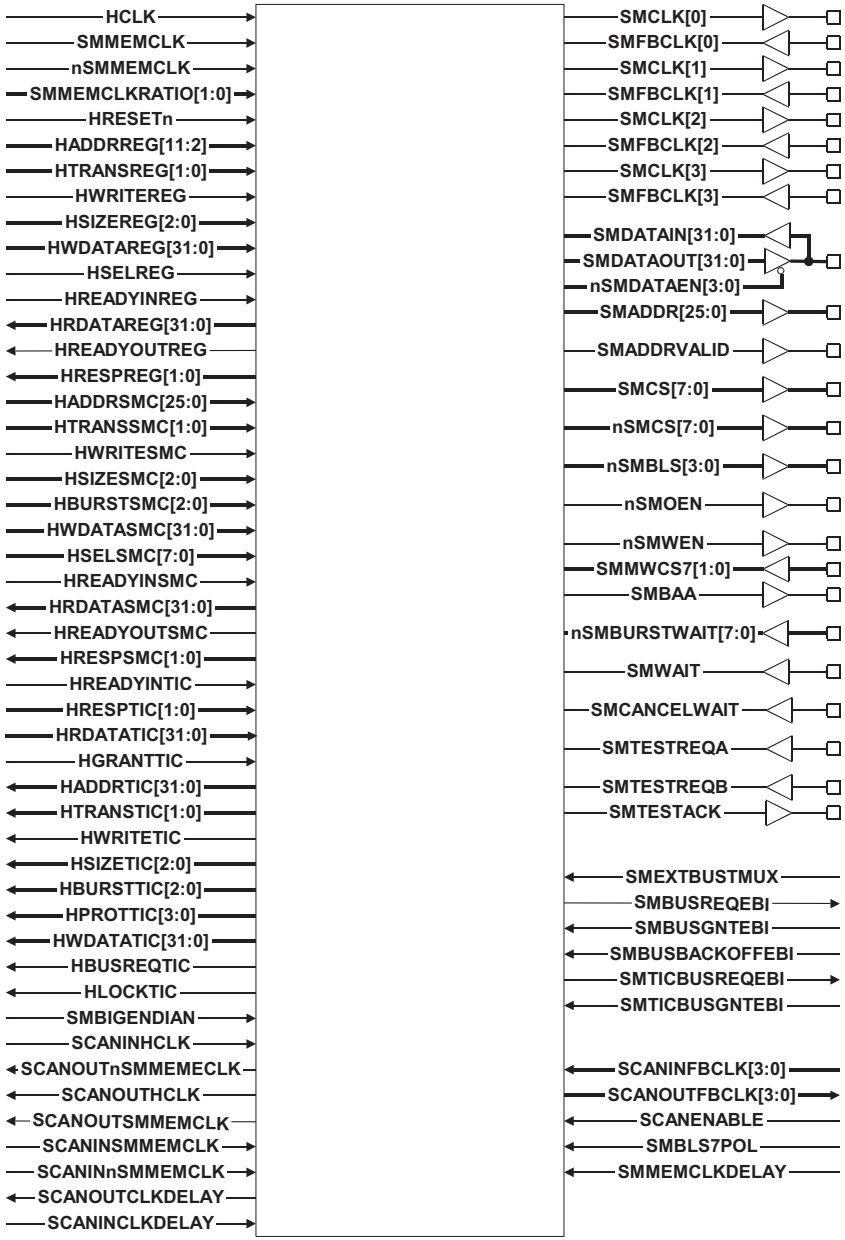



**Figure 1-3 SSMC input and output connections**

## 1.2 Supported memory devices

You can connect the SSMC to synchronous and asynchronous memory devices.

### 1.2.1 Asynchronous memory devices

The following devices are examples of the type of asynchronous devices you can connect to the SSMC:

**AMD** 16Mb page mode flash, Am29PL160C (2M x 8B only).

**Micron** 4Mb flash MT28F004B5.

**Samsung**
- 128Mb ROM K3N9V(U)10000M-YC
- 4Mb SRAM K6R4016C1C
- 8Mb SRAM K6T8016C3M.

### 1.2.2 Synchronous memory devices

The following devices are examples of synchronous devices you can connect to the SSMC:

**Note**

You must program flash devices to start in synchronous mode at power-on reset. Do this by passing code from another memory device. When the flash device is programmed then the device powers up in the standard synchronous mode. You must then program the SSMC to operate with the synchronous flash device.

**Intel**
- 3V synchronous Strata flash, 28F640K3
- 1.8V wireless flash memory with 3V input or output and SRAM, 28F6408W30, 28F3204W30, 28F320W30, and 28F640W30
- 1.8V wireless flash memory (W18), 28F320W18, 28F640W18, and 28F128W18.

**Micron**
- 2Mx16 burst flash, MT28F322D15
- 2Mx16 burst flash, MT28F322D18
- 2Mx16 burst flash, MT28F322D20.

**AMD** 3V-only high-performance burst mode and page mode flash memories:
- 8Mb (512K x 16-bit boot sector, Am29BL802C)
- 16Mb (1M x 16-bit boot sector, Am29BL162C)
- 1.8V burst mode flash, 64 Mb (4Mx16-bit), Am29BDS640G.

## 1.3 Product revisions

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

**r0p0-r0p1** Contains the following differences in functionality:

- two consecutive writes of differing HSIZE now give the correct byte lane
- 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.

**r0p1-r0p2** Contains the following differences in functionality:

- A second access of burst to memory immediately after reset is no longer lost if WSTRD <=2.
- There is now no restriction in changing from 1:1 to 1:2 to 1:3 clock ratio.
- A non-word write access to a register no longer updates the configuration registers.
- There is no longer any difficulty in integration testing the **SMBUSGNTEBI** signal.
- The SSMC now supports SMBAA with WSTRD=0.
- There is an improvement in performance for HSIZE > MSIZE transfers when the SSMC memory transfer state machine is in the turnaround state.
- **nSMWEN** is driven with regard to **nSMMEMCLK** for asynchronous memories.
- SSMCPeriphID2 Register bits [7:4] now read back as `0x1`.

**r0p2-r0p3** Contains the following differences in functionality:

- Incorrectly handled synchronous burst transfers have been remedied.

**r0p3-r0p4** Contains the following differences in functionality:

- There is no change to the functionality described in this manual. See the engineering errata that accompanies the product deliverables for more information.

# Chapter 2
# **Functional Overview**

This chapter describes the SSMC operation. It contains the following sections:

## 2.1 About the SSMC

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

The SSMC provides simultaneous support for up to eight memory banks that you can configure independently. Each memory bank supports:

- SRAM
- ROM
- flash EPROM
- burst SRAM, ROM, and flash.

You can configure each memory bank to use 8, 16, or 32-bit external memory data paths. You can configure the SSMC to support either little-endian or big-endian operation.

You can configure the SSMC memory banks to support:

- nonburst read and write accesses to high-speed CMOS asynchronous static RAM
- nonburst write accesses, nonburst read accesses, and asynchronous page mode read accesses to fast-boot block flash memory
- synchronous single and burst read and write accesses to synchronous static RAM.

Figure 2-1 shows a block diagram of the SSMC.

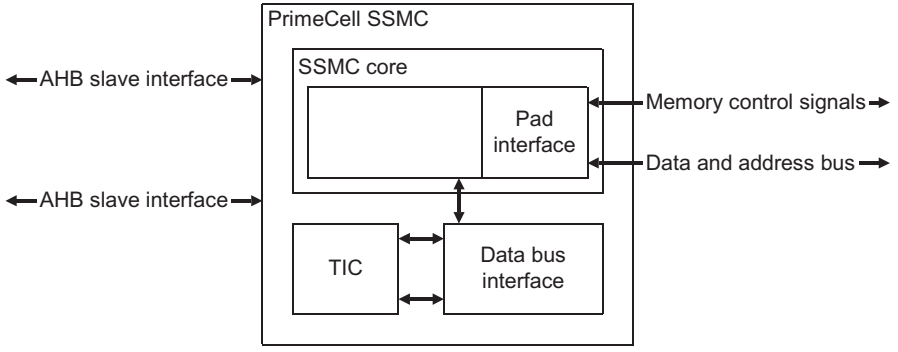



**Figure 2-1 SSMC block diagram**

The four main sub-blocks in the SSMC design are:

**SSMC core** The SSMC core performs read and write accesses to external memory through the AMBA AHB slave interface. See *SSMC core* on page 2-3.

**Data bus interface** The data bus interface selects either the SSMC core or the TIC as the current user of the external data bus and address bus.

**TIC** During testing, the TIC reads external test vectors and applies them to the system through the AMBA AHB master interface.

**Pad interface** The pad interface drives out the control and address signals to the memory relative to **SMMEMCLK**, **nSMMEMCLK**, and **SMMEMCLKDELAY**. It resynchronizes all the external signals to the feedback clock **SMFBCLK**. The pad interface registers the read data with regard to **FBCLK** for synchronous access and **SMMEMCLK** for asynchronous memory access.

### 2.1.1 SSMC core

The SSMC core performs read and write accesses to external memory through the AMBA AHB slave interface. Figure 2-2 on page 2-4 shows a block diagram of the SSMC core.

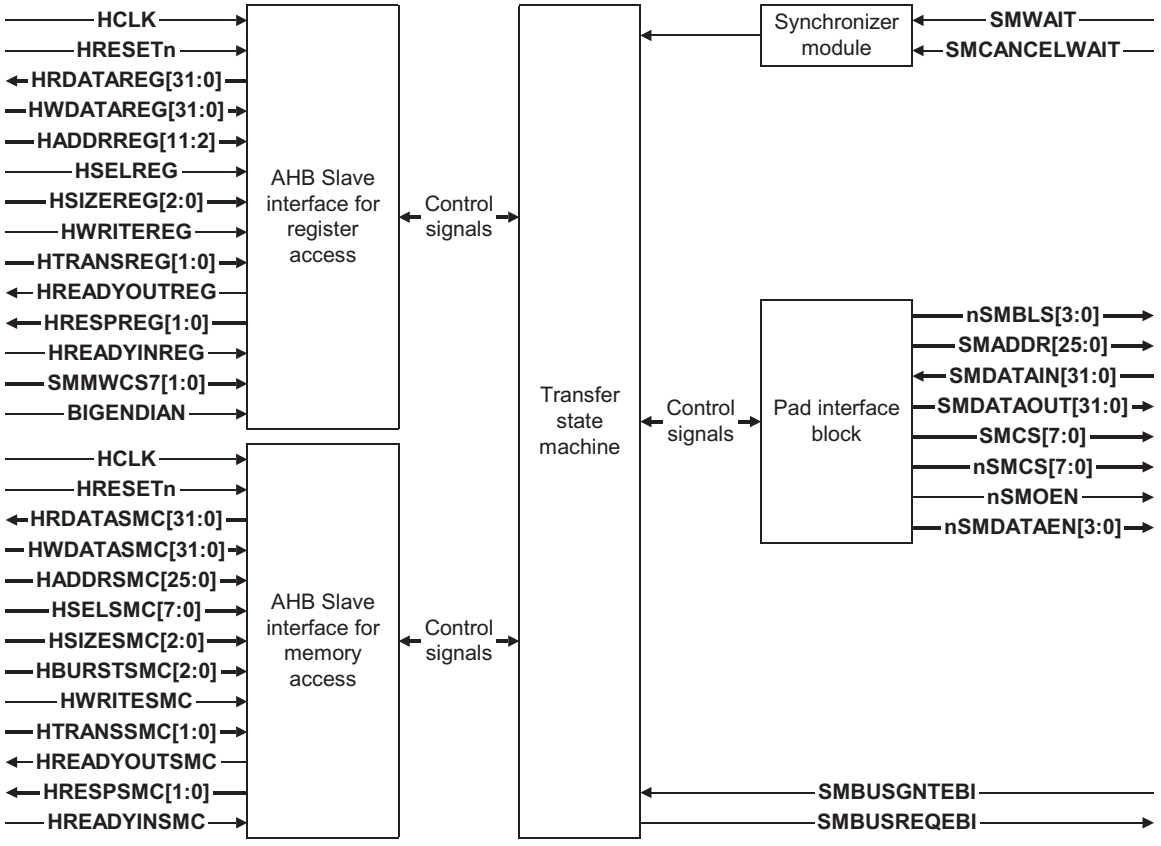



**Figure 2-2 SSMC core block diagram**

The SSMC Core logic consists of the following major sub-blocks:

- The AHB memory interface provides the interface between the SSMC and the AHB ports.
- AHB register interface provides the interface between the SSMC registers and the AHB register port.
- The transfer state machine controls all of the SSMC transactions to the external memory device.
- The pad interface multiplexes the appropriate control, address, and data signals out to the external memory device.

## 2.2 Operation

This section describes the operation of the SSMC, including the timing of standard transfers for different memory types and externally waited transfers, and example configurations for different memory device and bank sizes. The functions of the SSMC are described in the following sections:

- *Clock frequency selection*
- *Memory bank select* on page 2-6
- *Access sequencing and memory width* on page 2-8
- *Wait state generation* on page 2-8
- *Write protection* on page 2-9
- *Asynchronous static memory read control* on page 2-9
- *Synchronous static memory read control* on page 2-16
- *Asynchronous static memory write control* on page 2-21
- *Synchronous static memory write control* on page 2-26
- *Bus turnaround* on page 2-27
- *Synchronous memory devices bus turnaround* on page 2-30
- *Asynchronous external wait control* on page 2-30
- *Synchronous external wait control* on page 2-34.

### 2.2.1 Clock frequency selection

The SSMC requires the following clock inputs:

**HCLK** The AHB interface is clocked using **HCLK**.

**SMMEMCLK** **SMMEMCLK** clocks the memory interface. You must derive **SMMEMCLK** externally from the same source as **HCLK**.

**nSMMEMCLK** Pad interface uses **nSMMEMCLK**, which is inverted **SMMEMCLK**.

**SMMEMCLKDELAY**

Pad interface uses **SMMEMCLKDELAY**. You are recommended to use inverted **SMCLK** for this.

The frequency can be equal to $\mathbf{F_{HCLK}}$, $\mathbf{F_{HCLK/2}}$, or $\mathbf{F_{HCLK/3}}$. You define the frequency ratio at reset by the **SMMEMCLKRATIO[1:0]** signal. Software can then control it by writing to the SSMCCR Register.

**Note**

You must set the SSMCCR register to match the externally defined clock ratio.

This procedure describes how to change the clock frequency ratio:

1. Ensure that there are no memory transfers.
2. Switch the **SMMEMCLK** to the new frequency ratio externally (at the source of **SMMEMCLK**).
3. Ensure that there are no memory transfers.
4. Switch **SMMEMCLK** externally to the new ratio.
5. Ensure that **SMMEMCLK** has switched to the new ratio then write to the Clock Ratio Register:
   - when switching 1:2, the update must occur as you do it.
   - when switching 1:3 or 2:3, the update must be on an aligned **HCLK** and **SMMEMCLK** cycle.
6. Wait for four **SMMEMCLK** clock cycles.
7. Start the memory accesses.

See *SSMC Control Register* on page 3-17 and *Non-AMBA signals* on page A-7 for more information.

### 2.2.2 Memory bank select

Eight memory banks are supported. A pair of chip selects are available for each bank, one active LOW and one active HIGH. You only have to bond out one of these as required for the memory system. Each bank is individually selected by using a separate **HSEL** line.

The memory bank selection is controlled by the AMBA AHB **HSELSMC[7:0]** lines. Table 2-1 on page 2-7 lists these.

The addresses of the external memory banks and the base address of the SSMC memory bank registers are defined in the AMBA AHB address decoder, that generates the AHB slave select signals **HSELSMC** and **HSELREG**. This enables you to provide address space for the real memory in the system. You do not have to leave room for eight 64MB banks of memory if you are never going to use this size of memory in the system.

**Note**

If you use flash memory that requires instruction writes, you must take care to enable all of the required address bits to be available. Therefore, in this case it might be necessary to enable a 64MB region.

**Table 2-1 Static memory bank select coding**

| HSELSMC[7:0] | SMCS[7:0] | nSMCS[7:0] | Memory bank |
|---|---|---|---|
| 00000001 | 00000001 | 11111110 | Bank 0 |
| 00000010 | 00000010 | 11111101 | Bank 1 |
| 00000100 | 00000100 | 11111011 | Bank 2 |
| 00001000 | 00001000 | 11110111 | Bank 3 |
| 00010000 | 00010000 | 11101111 | Bank 4 |
| 00100000 | 00100000 | 11011111 | Bank 5 |
| 01000000 | 01000000 | 10111111 | Bank 6 |
| 10000000 | 10000000 | 01111111 | Bank 7 |

Table 2-2 lists an example address mapping of **HADDRSMC[31:0]** for external memory banks. You do not have to use all 26 bits if the device is smaller than 64MB.

**Table 2-2 Address mapping for external memory banks**

| 31-26 | 25-0 |
|---|---|
| Base address for SSMC memory bank for decode by the system decoder | Up to 64MB memory banks address space |

Table 2-3 lists the address mapping of **HADDREG[31:0]** for memory bank configuration registers.

**Table 2-3 Address mapping for memory bank registers**

| 31-12 | 11-2 | 1-0 |
|---|---|---|
| Base address for PrimeCell AHB SSMC registers for decode by system decoder | Memory bank register select | Registers only accessible as 32 bits. Lowest two bits are not used for selecting bytes within register. |

**Note**

**HADDRSMC[31:26]** and **HADDRREG[31:12]** are not inputs to the SSMC. Some or all bits perform decoding in the AMBA address decoder.

### 2.2.3 Access sequencing and memory width

You must configure the data width of each external memory bank by programming the appropriate Bank Control Register, SMBCRx. 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, in the case that Bank 0 is configured as 8-bit wide memory and a 32-bit read is initiated, the AMBA AHB bus stalls while the SSMC 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 **SMBIGENDIAN** input signal to the SSMC. For more details, see *Byte lane control and data bus steering for little and big-endian configurations* on page 2-40.

### 2.2.4 Wait state generation

You must configure each bank of the SSMC for external transfer wait states in read and write accesses. Do this by programming the appropriate fields of the Bank Control Registers:

- SMBIDCYRx
- SMBWSTRDRx
- SMBWSTBRDRx
- SMBWSTWRRx.

The number of cycles required to complete an AMBA transfer is also controlled by:

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

Each bank of the PrimeCell AHB SSMC 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 WSTRD 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 devices.

You can program the WSTBRD wait state field to select up to 31 wait states in the case of burst mode reads from burst devices.

You can program the WSTWR wait state field to select up to 31 wait states for write access to SRAM.

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 is 12 wait states for the first access and six for the subsequent accesses.

### 2.2.5 Write protection

You can configure each memory bank for write protection. Usually, SRAM is unprotected and you must write-protect ROM devices, but you can set the WP field in the Bank Control Registers, SMBCRx, to write-protect SRAM and ROM devices.

**Note**

If a write access is made to a write protected memory bank, an error is indicated by the **HRESP[1:0]** signals. 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.

### 2.2.6 Asynchronous static memory read control

The static memory read controls are described in:

- *Output enable programmable delay*
- *Asynchronous memory device accesses* on page 2-10
- *Asynchronous burst and page mode devices* on page 2-13.

#### 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 Bank Control Registers. This delay reduces 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 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 WSTRD programmed value, or an invalid access is generated. The access is timed by the WSTWR value and not by the WSTOEN value.

In the External Wait enabled mode, the timing of the transfer (controlled by **SMWAIT**) is not known, so **SMOEN** is asserted along with **SMCS** and the WSTOEN delay value is not used.

### Asynchronous memory device accesses

The SSMC 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 WSTRD field of the Bank Read Wait State Control Register, SMBWSTRDRx. The IDCY field in the Idle Cycle Control Register, SMBIDCYRx, determines the number of bus turnaround wait states added between external read and write transfers.

All the timing diagrams in this section, unless otherwise noted, show the signals timed relative to **HCLK**. This assumes that **SMMEMCLK** is at the same frequency as **HCLK**. Because the accesses are asynchronous, the memory devices ignore the external clock. Some examples are included to show accesses when **SMMEMCLK** is a ratio of **HCLK**.

The **SMADDRVALID** signal might be required for synchronous static memory devices when you use it in asynchronous mode. You can disable this using the AddrValidReadEn bit in the SMBCRx Register. This bit defaults to being set (enable) to enable a system to boot from synchronous memory. You can then clear it if you do not require it. When disabled, the signal is driven HIGH continuously.

Figure 2-3 shows a single external memory read transfer with the minimum zero wait states, WSTRD=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.

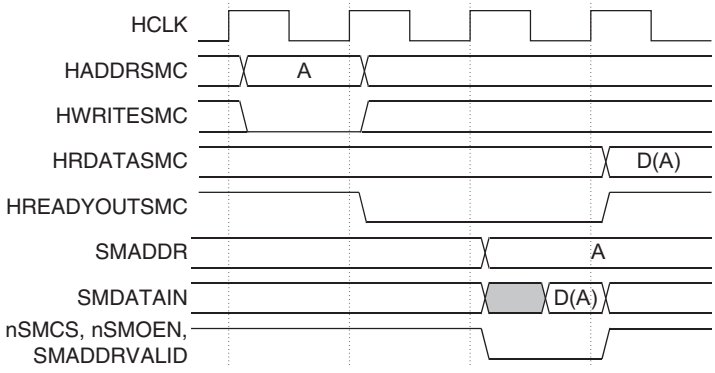



**Figure 2-3 External memory zero wait state read**

Figure 2-4 on page 2-11 shows a single external memory read transfer with the minimum zero wait states, WSTRD=0, and the minimum zero output enable delay states, WSTOEN=0. For this access, **SMMEMCLK=HCLK**/2. A minimum of three AHB wait states are inserted.

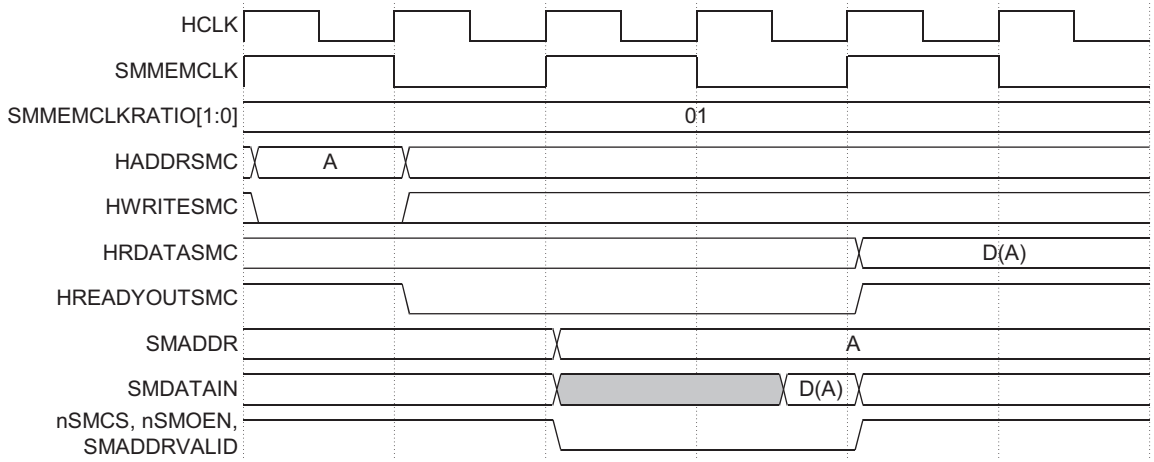



**Figure 2-4 External memory zero wait state read with SMMEMCLK=HCLK/2**

Figure 2-5 shows a single external memory read transfer with the minimum zero wait states, WSTRD=0, and the minimum zero output enable delay states, WSTOEN=0. For this access **SMMEMCLK=HCLK/**3. The diagram shows that there are five AHB wait states inserted. The number of wait states is dependent on the time the access occurs with the minimum number of wait states being four and the maximum being six.

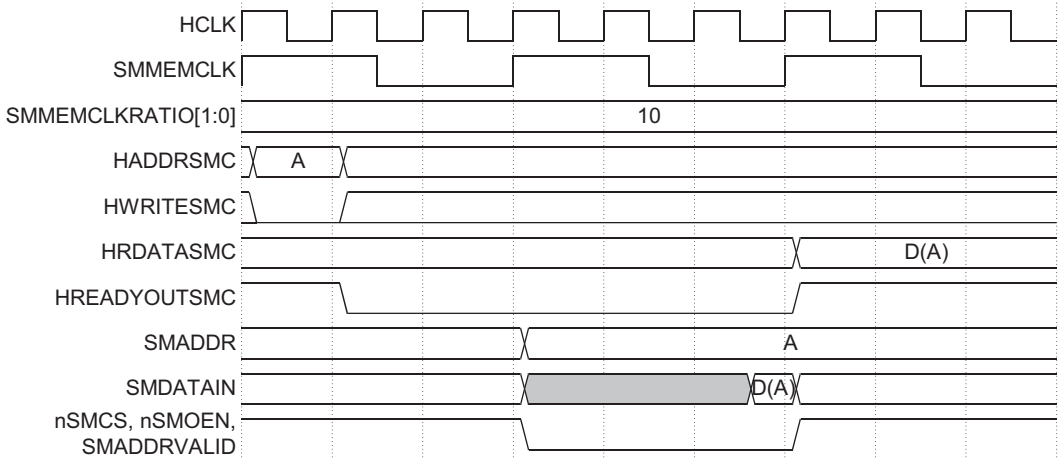



**Figure 2-5 External memory zero wait state read with SMMEMCLK=HCLK/3**

Figure 2-6 shows an external memory read transfer with two wait states, WSTRD=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
- an additional two because of the programmed wait states added.

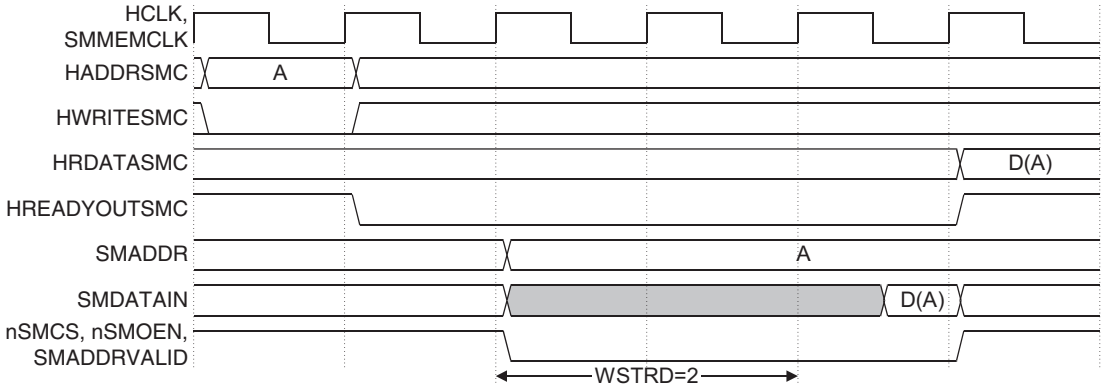



**Figure 2-6 External memory two wait state read**

Figure 2-7 shows an external memory read transfer with two output enable delay states, WSTOEN=2, and two wait states, WSTRD=2. 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.

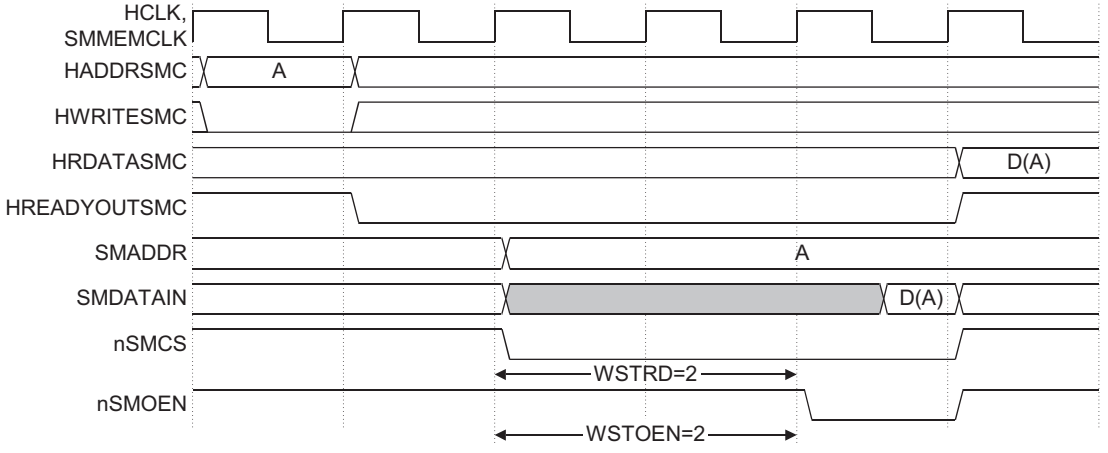



**Figure 2-7 External memory two output enable delay state read**

Figure 2-8 shows two external memory read transfers with zero wait states, WSTRD=0. These might be nonsequential transfers, or sequential transfers of unspecified burst length when in non-burst mode, BMRead=0 in the SMBCRx Register. All transfers are treated as separate reads, and so they have the minimum of two AHB wait states added.

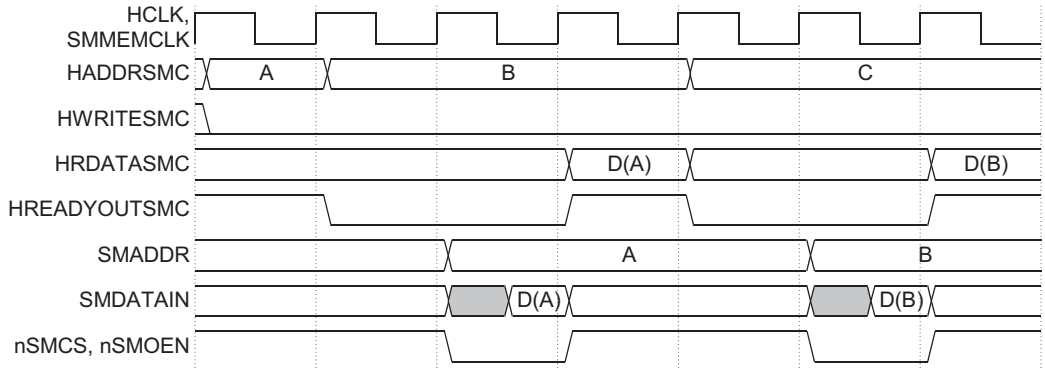



**Figure 2-8 Two zero wait state read transfers**

### Asynchronous burst and page mode devices

The SSMC supports sequential access asynchronous burst reads to four or eight consecutive locations in 8, 16, or 32-bit memories, as set using the BurstLenRead bits of the Control Register SMBCRx. Burst mode is enabled by setting the Burst Mode bits, BMRead or BMWrite, in the Control Register. This feature supports burst mode devices and increases the bandwidth by using a reduced access time (that you can configure) for the sequential reads, WSTBRD, following the first read, WSTRD. 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.

Asynchronous page mode read operation is supported. This is enabled by setting the BMRead bit and by setting the burst length using BurstLebRead in the SMBCRx Register. Sequential bursts of up to four or eight beats are the only type of access supported for page mode operation.

The following diagrams show examples of burst and page mode accesses.

**Note**

In 4-transfer burst mode, bursts cannot cross quad boundaries. These are:

- **HADDRSMC[1:0]**=11 for 8-bit transfers
- **HADDRSMC[2:1]**=11 for 16-bit transfers (**HADDRSMC[0]** ignored)

- **HADDRSMC[3:2]**=11 for 32-bit transfers (**HADDRSMC[1:0]** ignored).

They are split up so that the first transfer after the boundary uses the slow read (WSTRD) timing. For example, a four-byte transfer starting at address **HADDRSMC[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.

In 8-transfer burst mode, bursts cannot cross double quad boundaries. These are:

- **HADDRSMC[2:0]**=111 for 8-bit transfers
- **HADDRSMC[3:1]**=111 for 16-bit transfers (**HADDRSMC[0]** ignored)
- **HADDRSMC[4:2]**=111 for 32-bit transfers (**HADDRSMC[1:0]** ignored).

Figure 2-9 shows 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.

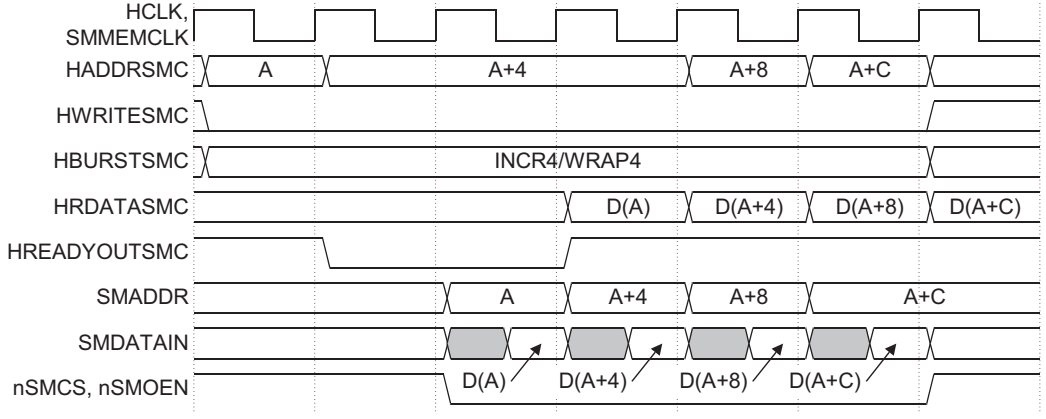



**Figure 2-9 External memory zero wait fixed length burst read**

Figure 2-10 on page 2-15 shows 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.

**Note**

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

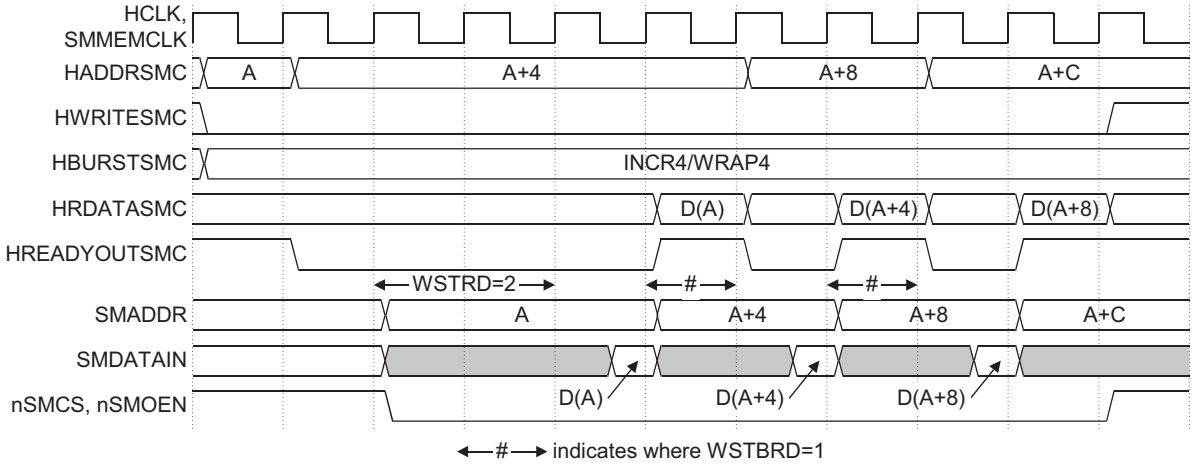



**Figure 2-10 External burst ROM with WSTRD=2 and WSTBRD=1 fixed length burst read**

Figure 2-11 shows a 32-bit read from an 8-bit burst 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.

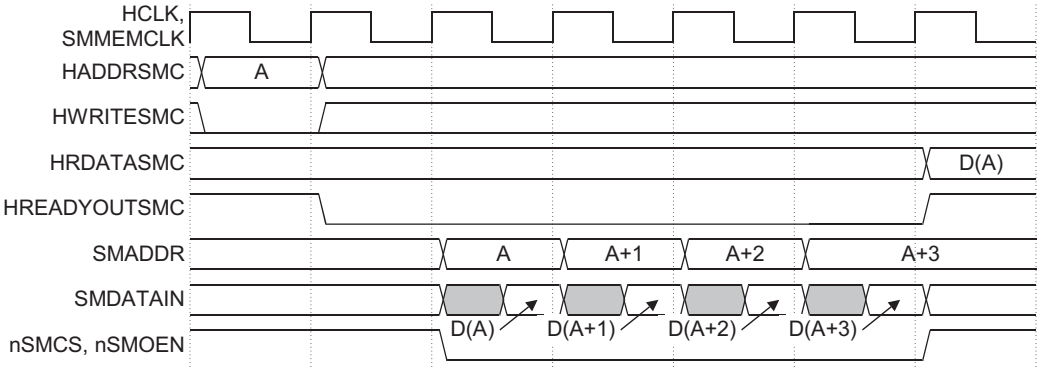



**Figure 2-11 External memory 32-bit burst read from 8-bit memory**

### 2.2.7 Synchronous static memory read control

Single synchronous read operations have the same control signal timing as an asynchronous read operation, but with different timing requirements for setup and hold relative to the clock. Because the output signals of the SSMC are generated internally from clocked logic, the timing for single synchronous reads is the same as for asynchronous reads.

Synchronous burst read transfers are performed differently to asynchronous burst reads, because of the internal address incrementing performed by synchronous burst devices. The **SMADDR** outputs are held with the initial address value, and the **SMADDRVALID** output is asserted during the transfer to indicate that the address is valid.

For some memory devices, the **SMBAA** signal causes the address to be advanced. This signal is only active if the BIReadEn or BIWriteEn bit is set in the Bank Control Register, SMBCRx.

On some memory parts, there is a signal named **IND**. This indicates when address A0 to A4 has reached b11111. This condition is sensed automatically by the SSMC and therefore no connection is made to memory.

Four, eight, or continuous synchronous burst lengths are supported, and are controlled by the BurstLenRead bits in the Bank Control Register SMBCRx when the SyncEnRead and BMRead bits indicate that the device supports synchronous bursts.

Figure 2-12 on page 2-17 shows a single synchronous read, where WSTRD=3 and WSTBRD=0.

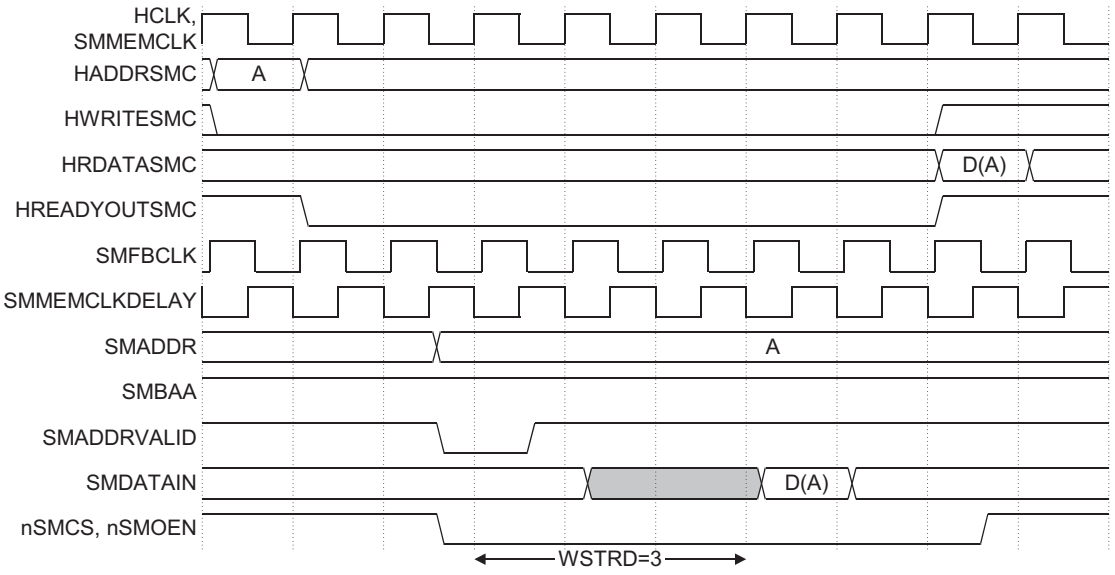



**Figure 2-12 External synchronous single transfer read**

Figure 2-13 on page 2-18 shows a fixed-length synchronous burst of four read transfers, where WSTRD=3 and WSTBRD=0. An eight-transfer burst is performed in a similar way if **HBURST** indicates a burst of eight or more transfers.

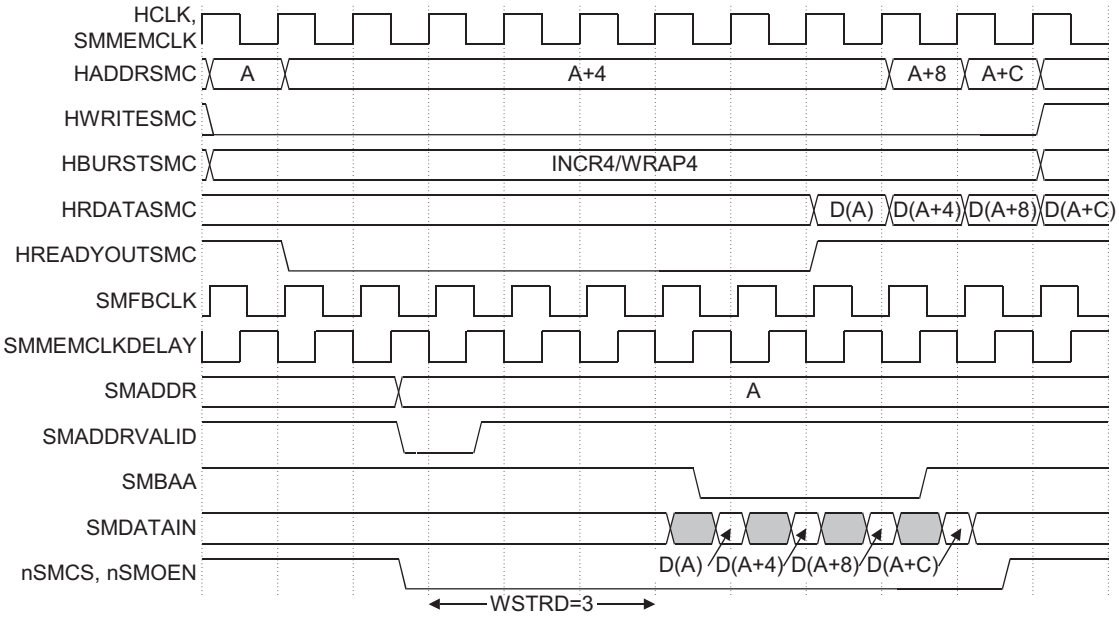



**Figure 2-13 External synchronous fixed length four transfer burst read**

As with asynchronous bursts, AHB reads with a size greater than the external memory might generate external burst transfers. For example, a 32-bit read from an 8-bit burst device generates a four-transfer burst to the external device.

Figure 2-14 on page 2-19 shows a zero wait continuous burst where, WSTRD=3 and WSTBRD=0.

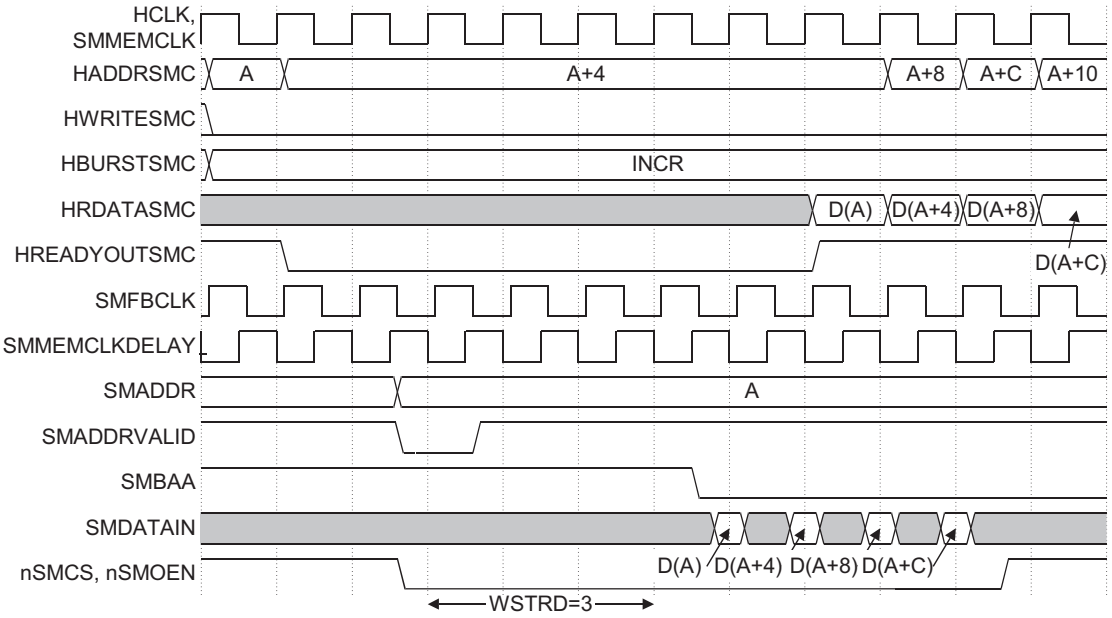



**Figure 2-14 External synchronous zero wait continuous length burst read**

Figure 2-15 on page 2-20 shows a fixed length synchronous burst of four read transfers, where WSTRD=3 and WSTBRD=0. An eight-beat transfer burst is performed in a similar way if **HBURST** indicates a burst of eight or more transfers.

For this access **SMMEMCLK=HCLK/**2. A minimum of 16 AHB wait states are inserted.

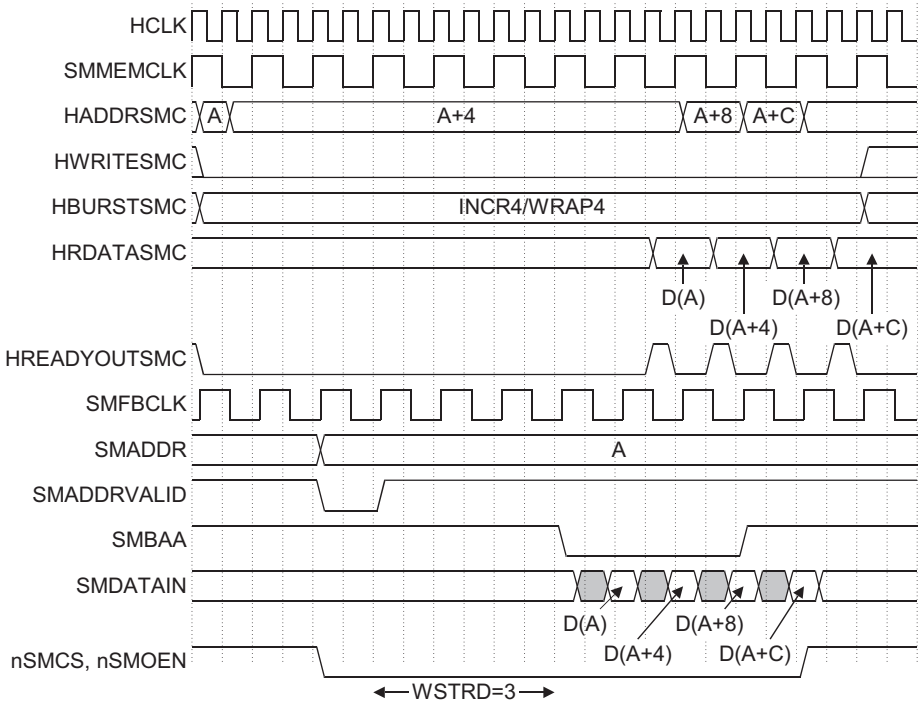



**Figure 2-15 External memory zero wait state read, WSTRD=3, WSTBRD=0, and SMMEMCLK=HCLK/2**

### 2.2.8 Asynchronous static memory write control

Write timing is described in:

- *Write enable programmable delay*
- *SRAM* on page 2-22
- *Flash memory* on page 2-25.

#### 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 Bank Write Enable Assertion Delay Control Register, SMBWSTWENRx. This reduces the power consumption for memories. The write enable is asserted on the rising edge of **nSMMEMCLK**, half a clock after the assertion of chip select.

For most asynchronous memory devices an **SMMEMCLK** cycle is required before the assertion of **nWE** otherwise there is the hazard that **nCS** changes after **nWE**. You can add extra cycles before **nWE** is asserted using the WSTWEN bits in the Bank Write Enable Assertion Delay Control Registers. For example, setting WSTWR=WSTWEN=1 extends the transfer by one cycle and delays the assertion of **nWE** by one cycle.

The write enable is always deasserted half a cycle 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 WSTWEN programmed value must be equal to, or less than the WSTWR programmed value otherwise an invalid access sequence is generated. The access is timed by the WSTWR value and not by the WSTWEN value.

In the External Wait enabled mode, the timing of the transfer (controlled by **SMWAIT**) is not known. WSTWEN still delays the assertion of **nSMWEN**. **nSMWEN** is delayed more by the external wait signal if it has not been asserted when **SMWAIT** is asserted.

---

You might require the **SMADDRVALID** signal for synchronous static memory devices when you use it in asynchronous mode. You can disable it using the AddrValidWriteEn bit in the SMBCRx Register. This bit defaults to being set (enable). You can then clear it if you do not require it. When you disable it, the signal is driven HIGH continuously.

### 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 WSTWR field of the Bank Write Wait State Control Registers, SMBWSTWRRx. The IDCY field in the Bank Idle Cycle Control Register SMBIDCYRx determines the number of bus turnaround wait states added between external read and write transfers.

Figure 2-16 shows a single external memory write transfer with the minimum zero wait states, WSTWR=0, and the minimum zero write enable delay states, WSTWEN=0. A single AHB wait state is inserted during all single write transfers.

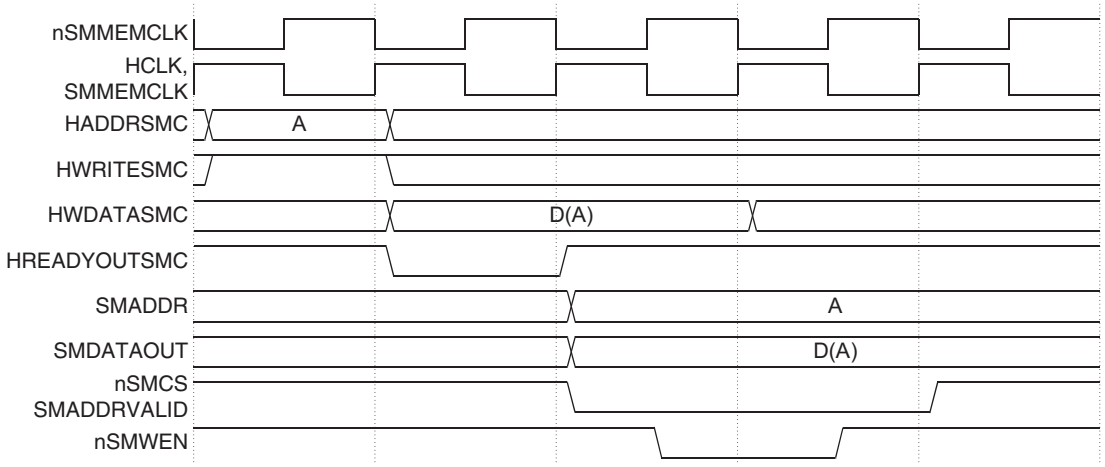



**Figure 2-16 External memory zero wait state write**

Figure 2-17 on page 2-23 shows a single external memory write transfer with two wait states, WSTWR=2, and the minimum zero write enable delay states, WSTWEN=0. A single AHB wait state is inserted.

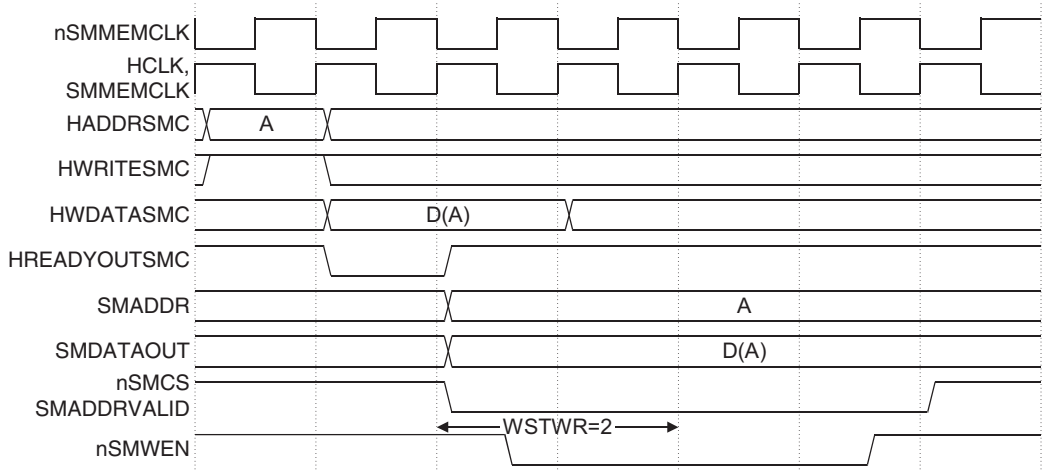



**Figure 2-17 External memory two wait state write**

Figure 2-18 shows a single external memory write transfer with two write enable delay states, WSTWEN=2, and two wait states, WSTWR=2. A single AHB wait state is inserted.

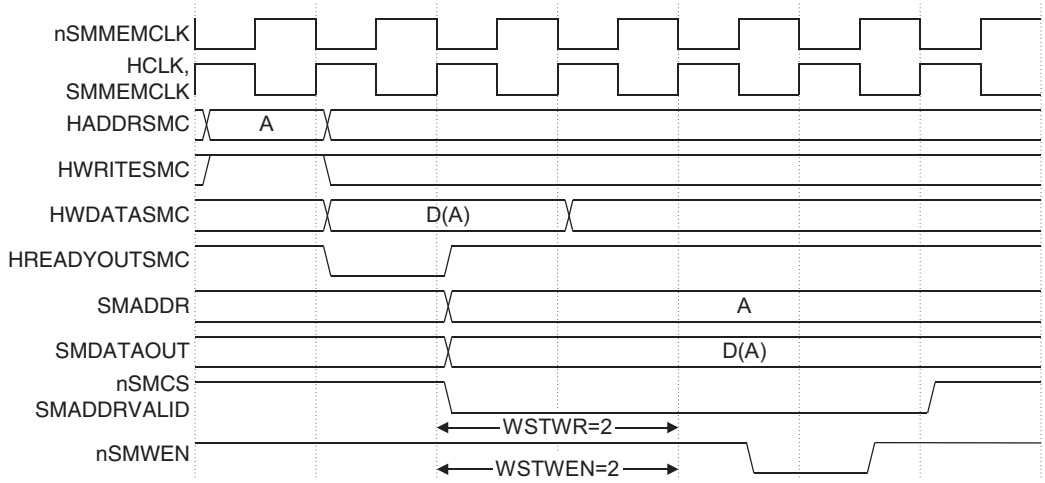



**Figure 2-18 External memory two write enable delay state write**

Figure 2-19 on page 2-24 shows a single external memory write transfer with a minimum zero wait states where the SSMC:

- is using an external bus multiplexor

- does not have control of the bus
- must request control of the bus.

In this example, nothing else is requesting the bus, so the SSMC is granted straight away, showing the minimum timing when the bus is requested.

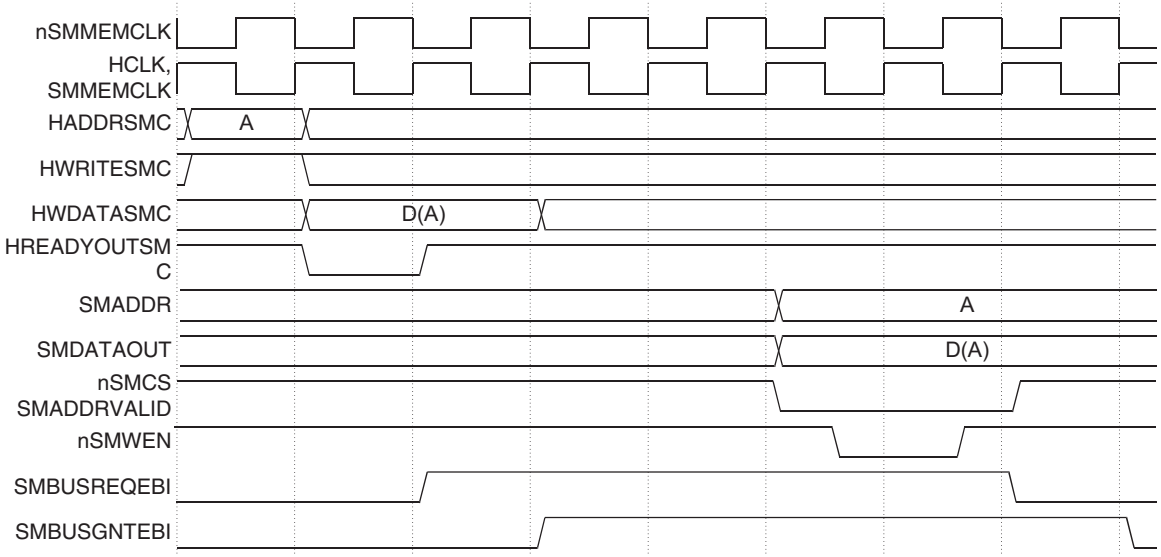



**Figure 2-19 External memory zero wait state write, bus not granted**

Figure 2-20 on page 2-25 shows two external memory write transfers with zero wait states, WSTWR=0. One AHB wait state is added to each 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 cycle that write enable is asserted, and the cycle that write enable is deasserted.

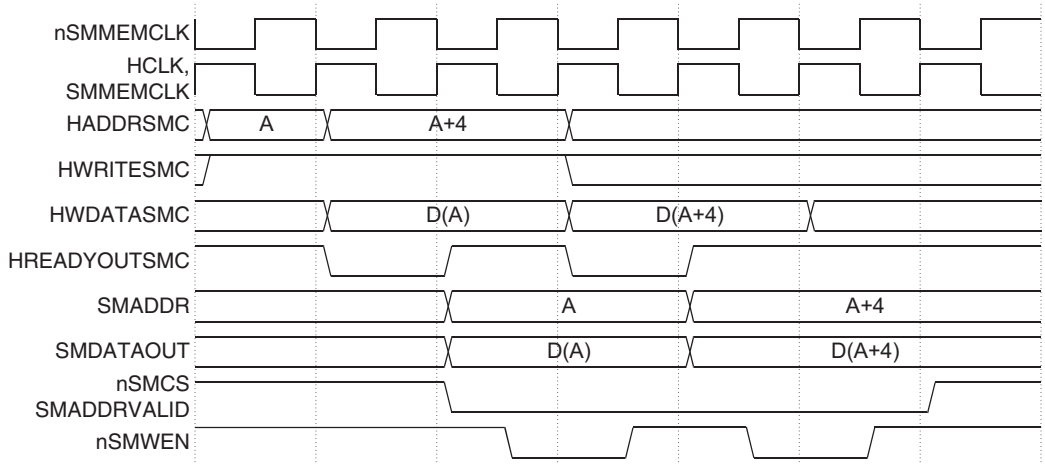



**Figure 2-20 External memory two zero wait writes**

### Flash memory

Write timing for flash memory devices is the same as for SRAM devices. For details of burst flash devices see *Synchronous static memory write control* on page 2-26.

### 2.2.9 Synchronous static memory write control

Figure 2-21 shows an example synchronous write operation. In this example the signal **SMADDRVALID** provides a one-cycle pulse. This behavior is enabled by setting the SyncWriteDev bit in the SMBCRx Register. You must also set the AddrValidWriteEn bit for synchronous write.

The signal **nSMWEN** is only active for one cycle. This is active at the start of the transfer unless it is delayed using the control bits WSTWEN to delay it.

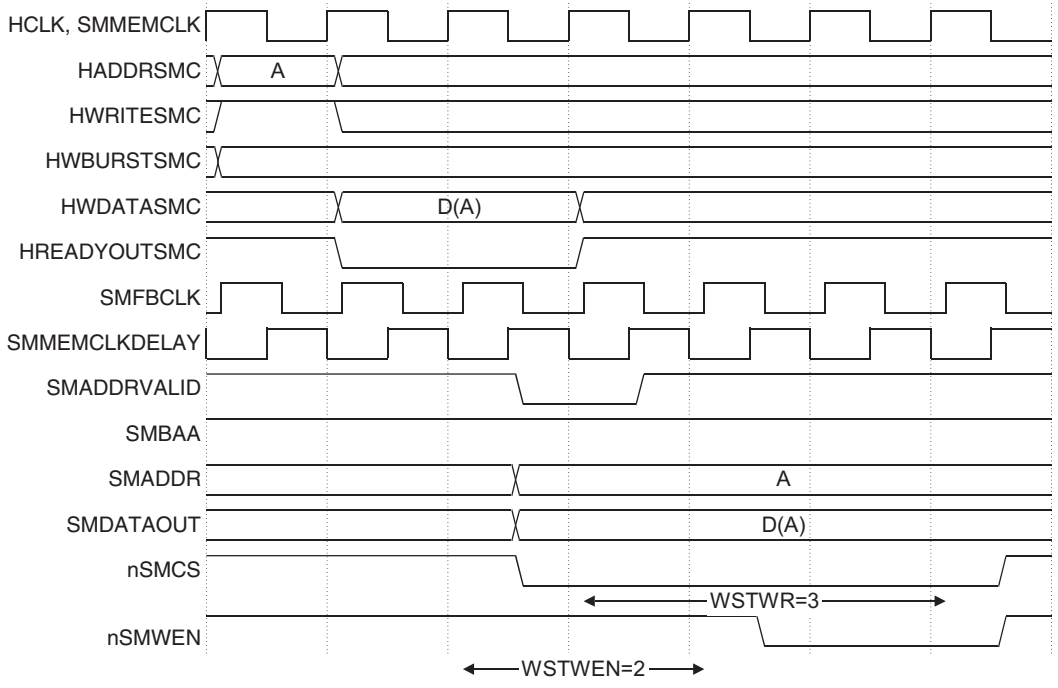



**Figure 2-21 Synchronous two wait state write**

Synchronous burst writes are supported by the SSMC. There is no write buffer so you must delay the AHB transfer to enable the data to be output onto the **SMDATA** bus. You can control the write in the same way as reads using the bits AddrValidWriteEn, BurstLenWrite, SyncEnWrite, and BMWrite contained in the Bank Control Register, SMCRx.

Figure 2-22 on page 2-27 shows an example of a burst write.

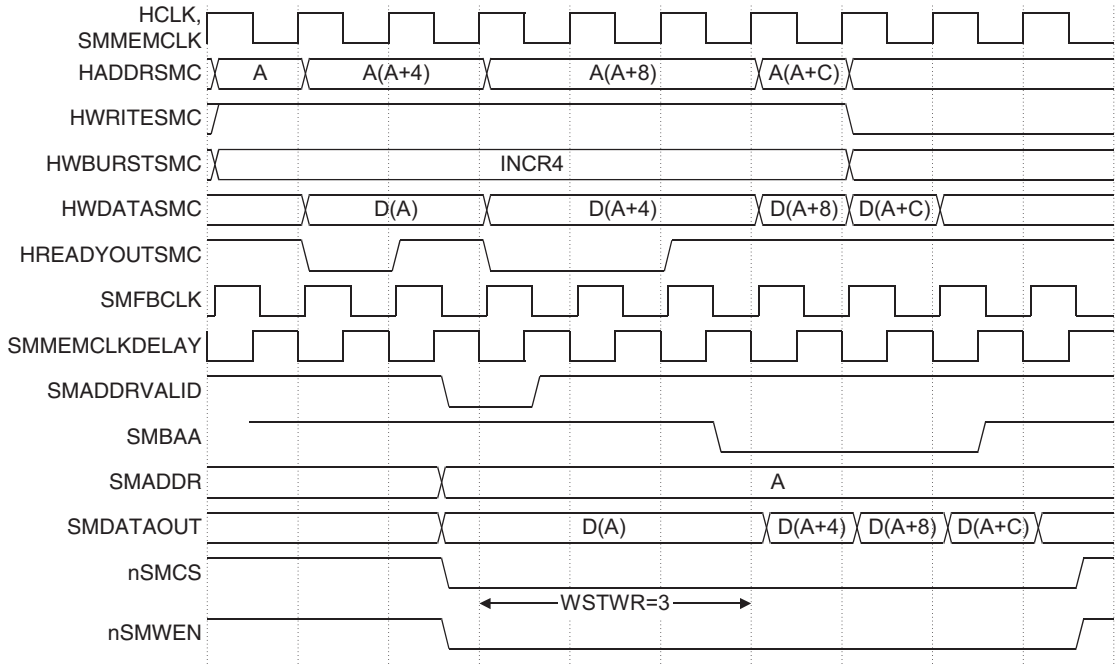



**Figure 2-22 Synchronous two wait state burst write with WSTWR=3 and WSTWEN=0**

### 2.2.10 Bus turnaround

You can configure the PrimeCell AHB SSMC for each memory bank to use external bus turnaround cycles between read and write memory accesses. You can program the IDCY field for up to 15 bus turnaround wait states. This avoids 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
- after single burst read and if IDCYC values are programmed before deasserting **SMBUSREQ** because there are no more memory accesses
- when a TimeOutErr has occurred for a read access and if IDCYC values are programmed

- when a WaitEnabled transfer has completed successfully for read access and if IDCYC values are programmed before deasserting **SMBUSREQ** because there are no more memory accesses.

Figure 2-23 shows a zero wait asynchronous read, WSTRD=0, followed by a zero wait asynchronous write, WSTWR=0, with default turnaround between the transfers. Standard AHB wait states are added to the transfers, two for the read, and one for the write.

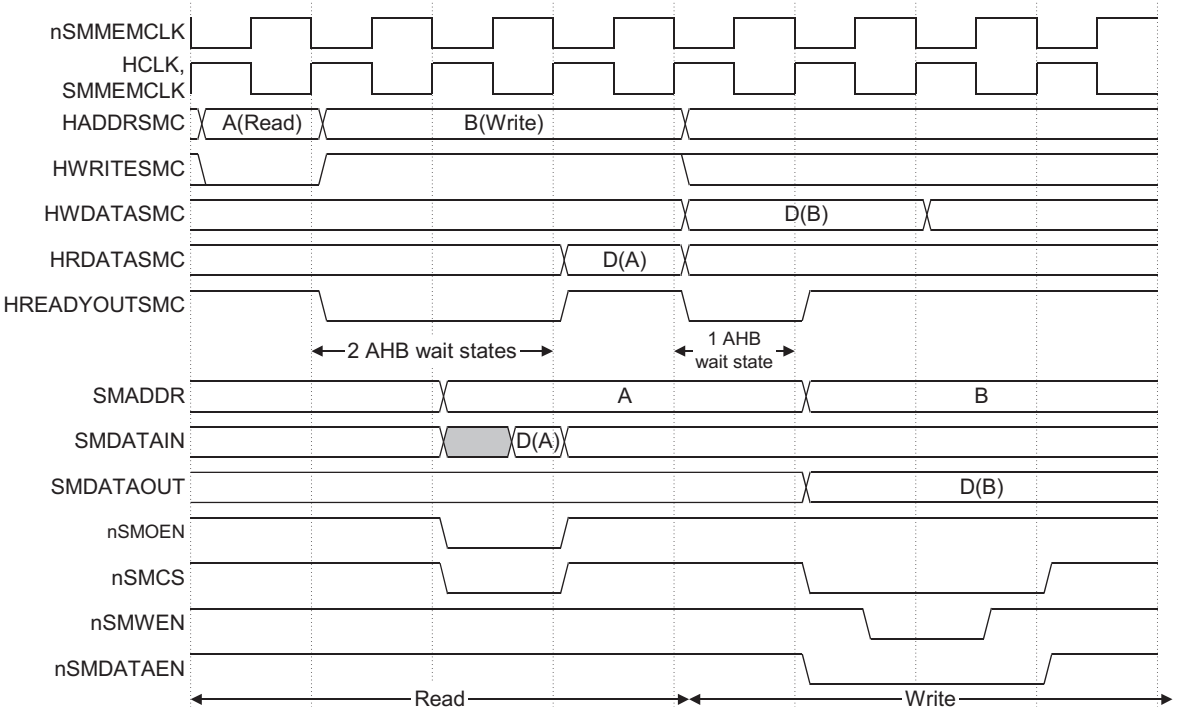



**Figure 2-23 Read followed by write (WSTRD=WSTWR=0) with no turnaround (IDCY=0)**

Figure 2-24 on page 2-29 shows a zero wait asynchronous write followed by a zero wait asynchronous read with default turnaround between the transfers. One AHB wait state is added to the write transfer, and four are added to the read:

- three to enable the write to complete before the read is started
- and then one is for the read transfer.

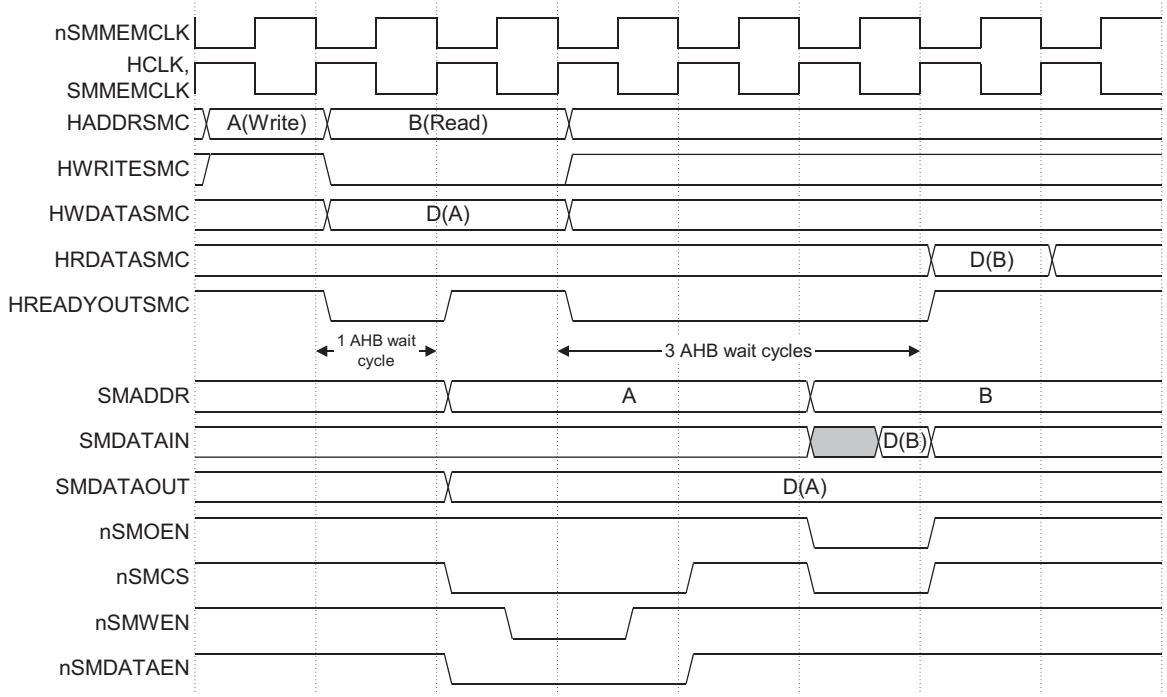



**Figure 2-24 Write followed by read (WSTRD=WSTWR=0) with no turnaround (IDCY=0)**

Figure 2-25 on page 2-30 shows a zero wait asynchronous read followed by two zero wait asynchronous writes with two turnaround cycles added. The standard minimum of two AHB wait states are added to the read transfer, one is added to the first write, as for any read-write transfer sequence, and three are added to the second write because of insertion of the two turnaround cycles that are only generated after the first write transfer has been detected, and the standard one wait state added when a write transfer is buffered.

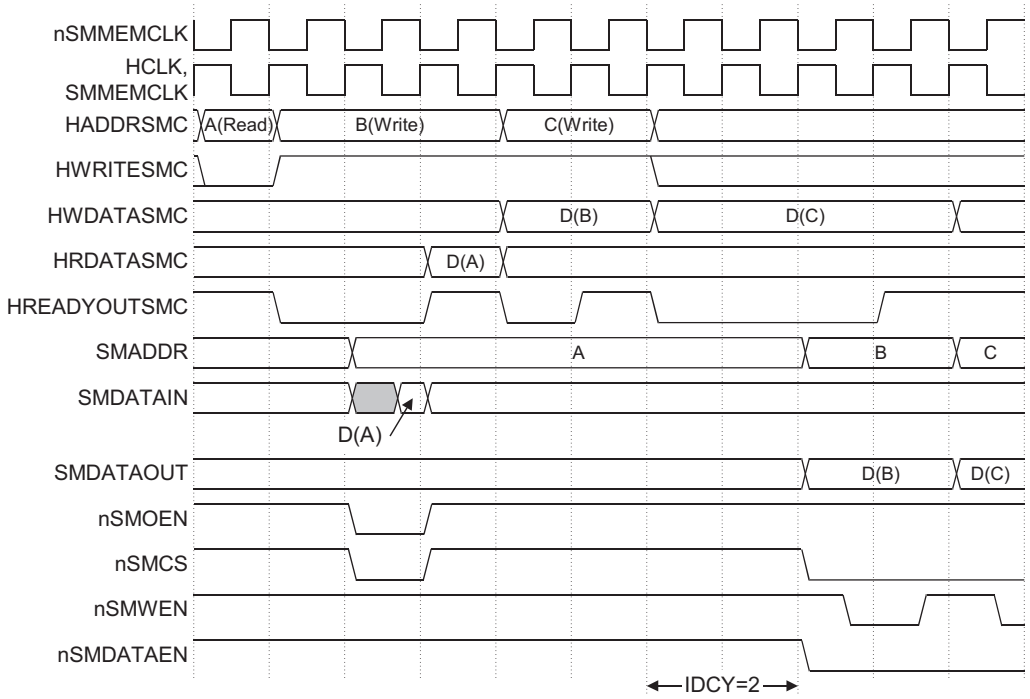



**Figure 2-25 Read, then two writes (WSTRD=WSTWR=0), two turnaround cycles (IDCY=2)**

### 2.2.11 Synchronous memory devices bus turnaround

The bus turnaround is the same in synchronous and asynchronous modes. Only the timing of the control signals changes that are timed relative to **SMFBCLK** and **SMMEMCLKDELAY** instead of **HCLK**. Additional cycles are added as shown in the diagrams for asynchronous devices.

### 2.2.12 Asynchronous external wait control

The SSMC supports extension of the access by an external device like a memory controller through use of the **SMWAIT** input pin of the SSMC. For this, you must program the WaitEn bit of the Bank Control Registers SMBCRx appropriately. You program the polarity of the external **SMWAIT** input 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 **SMCANCELWAIT**) are asynchronous inputs, they are synchronized before use. This gives all operations using external waits a two-cycle delay because of the synchronization time.

When external wait mode is enabled, the SSMC 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 by using bit-2, WaitEn, of the relevant Bank Control Register (SMBCRx), then the SSMC ignores the **SMWAIT** input and the access time is generated normally according to the values programmed in the WSTRD and WSTWR Registers.

### SMWAIT assertion timing

In the external wait control mode, when the SSMC is waiting for the **SMWAIT** assertion, it also starts counting down according to the values programmed in the wait state count field WSTRD or WSTWR, that perform read and write transfers respectively. You can use this feature to ensure that adequate time is available to the SSMC to detect the assertion of **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.

### SMWAIT deassertion timing

A waited transfer only ends when the **SMWAIT** input has been deasserted. The **SMCANCELWAIT** input is provided to enable the system to recover if the external device waits the transfer for longer than expected. You must use a timer or watchdog to control the assertion of the active HIGH **SMCANCELWAIT** input, and you must assert it for a minimum of one **HCLK** 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 Bank Status Register is asserted.

Because an external wait stops any external transfers being performed on the external bus, then the SSMC 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.

### SMWAIT timing diagrams

Figure 2-26 shows the timing 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 another two clock cycles onto the timing of the transfer.

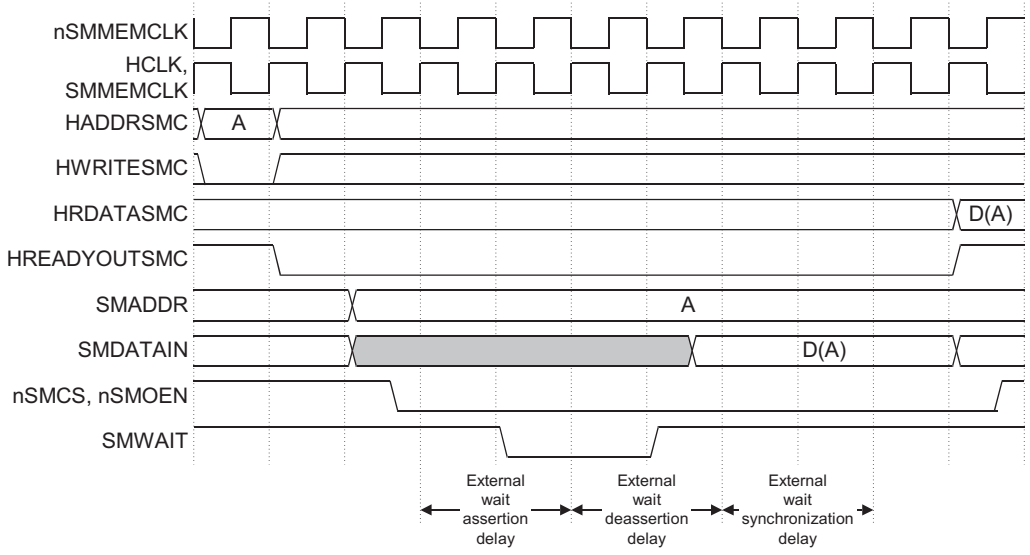



**Figure 2-26 External wait timed read transfer**

Figure 2-27 on page 2-33 shows the timing 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.

A transfer that is waited for externally is also waited for on the AHB, unlike a standard write transfer, that enables an error response (because of a timeout) to be generated correctly.

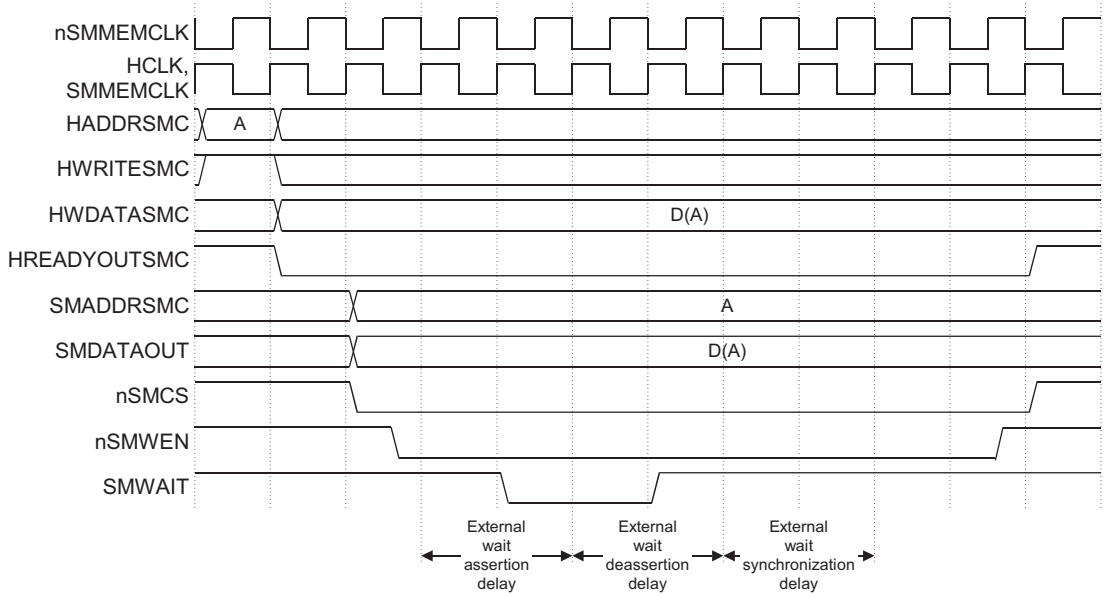



**Figure 2-27 External wait timed write transfer**

Figure 2-28 shows the timing for an aborted externally waited transfer.

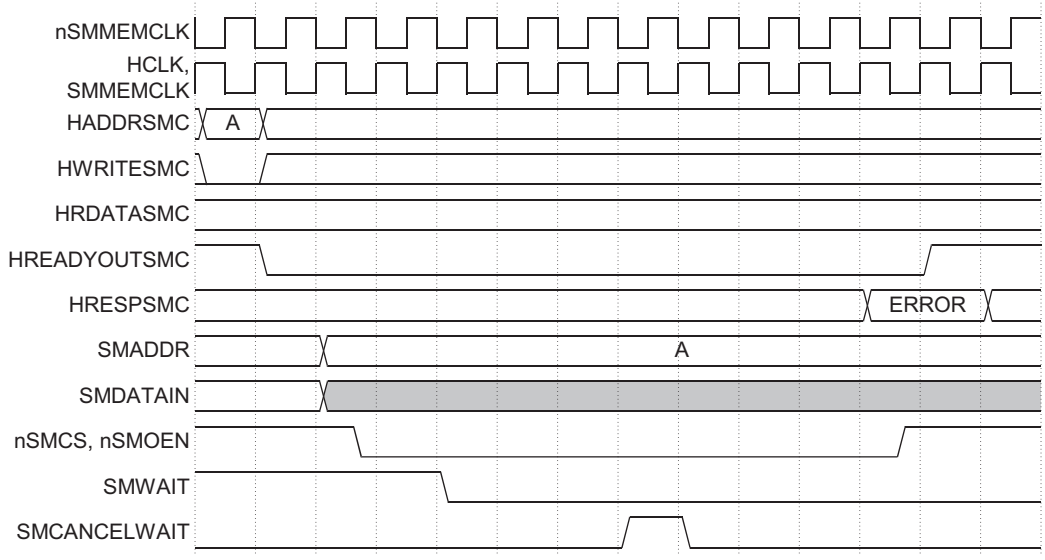



**Figure 2-28 External wait timed read transfer with external abort**

### 2.2.13 Synchronous external wait control

Burst transfers can be waited externally. Do this using the **nSMBURSTWAIT[7:0]** bus. One signal is provided for each bank of memory. The bank being accessed can assert (active LOW) the appropriate bit of the bus to delay the transfer. You must program the memory so that **nSMBURSTWAIT** is asserted in the cycle before the delay is to apply. Figure 2-29 shows an example transfer where the transfer is delayed by two cycles.

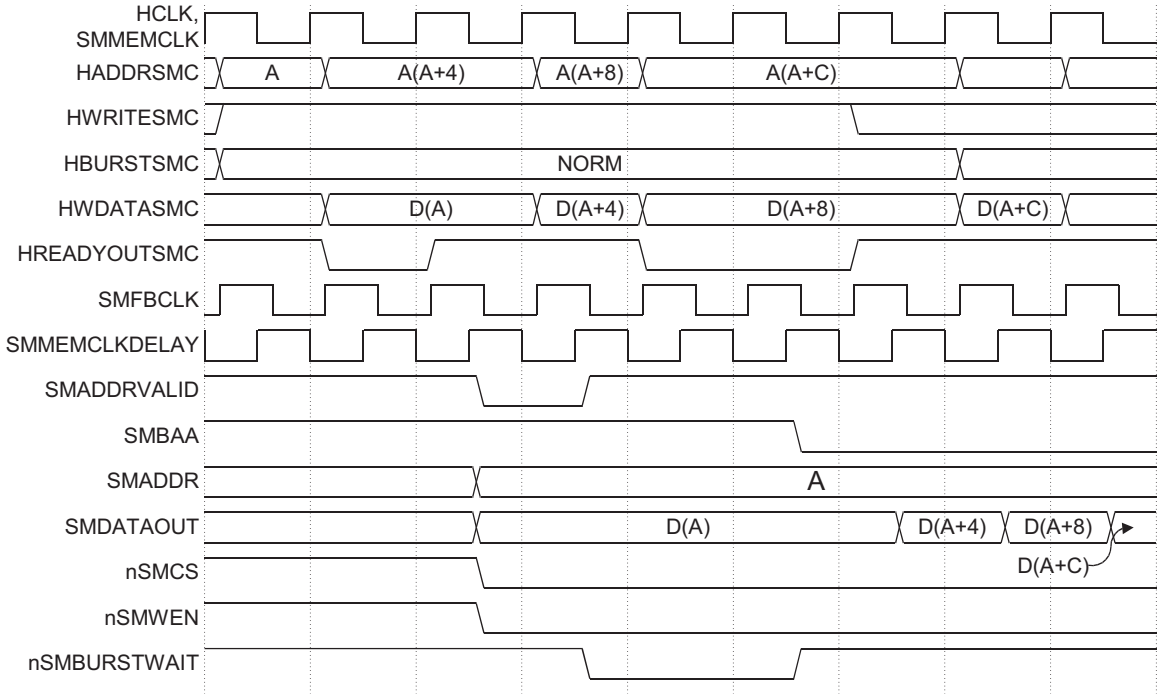



**Figure 2-29 Synchronous burst write, two-cycle external delay (WSTWR=2, WSTWEN=0)**

Figure 2-30 on page 2-35 shows an external synchronous wait applied to a zero wait continuous burst, where the read from location A+18 is delayed by three cycles.

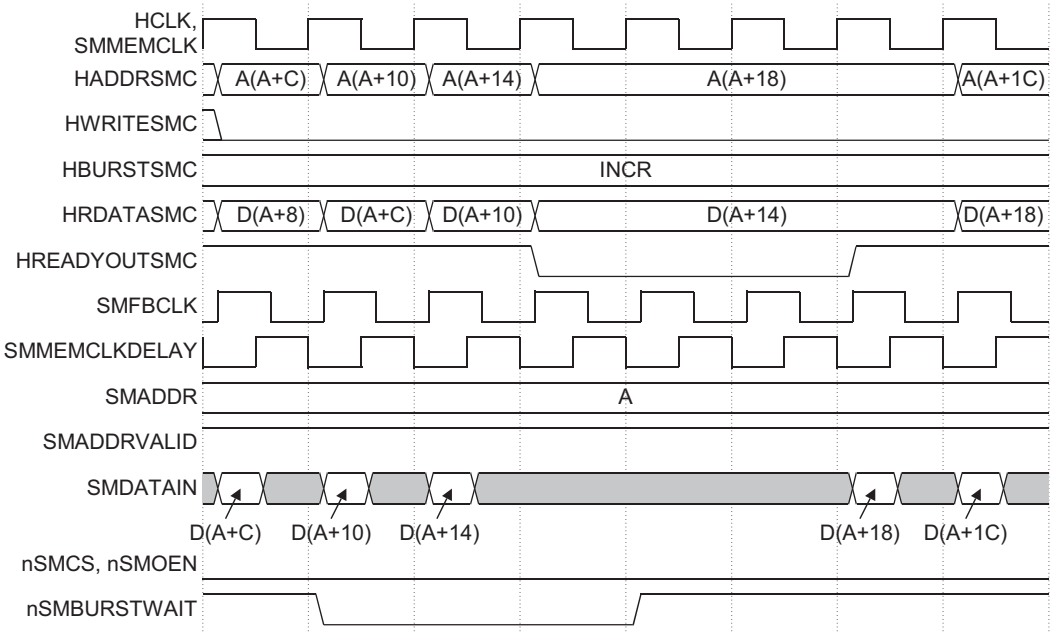



**Figure 2-30 External burst wait during synchronous continuous length burst read**

The external device uses the **nSMBURSTWAIT** signal to indicate that the current burst transfer is delayed, for example when crossing an address boundary. You can assert the **nSMBURSTWAIT** input synchronously at any time. You must deassert the **nSMBURSTWAIT** signal a cycle before the read data is valid. Figure 2-30 shows this.

The signals **SMWAIT** and **CANCESMWAIT** are not used during synchronous transfers. This means there is no way to cancel the wait as for asynchronous devices.

## 2.3 System-on-chip design considerations

This section describes the following design considerations:

- *nSMBURSTWAIT*
- *Byte lane control*
- *Clock feedback in SSMC* on page 2-44.

### 2.3.1 nSMBURSTWAIT

The **nSMBURSTWAIT[7:0]** signal enables a separate wait signal for each bank of memory. During synchronous accesses, this signal delays the transfer. You must provide one bit per bank because some memory devices drive this signal continuously. If you do not use a signal, then you must tie it to one.

You can use a single pin with memory parts that tristate the **nSMBURSTWAIT** signal if not selected. You must connect this signal to all bits of **nSMBURSTWAIT[7:0]**.

### 2.3.2 Byte lane control

The SSMC 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 in each Bank Control Register
- external memory bank type, byte, halfword, or word
- the decoded **HADDR[1:0]** value for write accesses only.

Word transfers are the largest size transfers supported by the SSMC. Any access attempted with a size greater than a word causes the generation of an error response. Each memory bank can be 8, 16, or 32 bits wide. The memory configuration for a particular bank determines how the **nSMWEN** and **nSMBLS** signals are connected to provide byte, halfword, and word access. For read accesses, you must control the **nSMBLS** signals by driving them either all HIGH or all LOW.

Do this by programming the *Read Byte Lane Enable* (RBLE) bit in each Bank Control Register. *Accesses to memory banks constructed from 8-bit or non byte-partitioned memory devices* on page 2-37 and *Accesses to memory banks constructed from 16 or 32-bit memory devices* on page 2-38 explain why different connections in respect of **nSMWEN** and **nSMBLS[3:0]** are required for different memory configurations.

The **SMBLS7POL** input controls the reset value of **nSMBLS**:

- 0 = **nSMBLS** is active LOW. This is the default.
- 1 = **nSMBLS** is active HIGH.

### Accesses to 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 you clear the RBLE bit to zero within the respective Bank Control 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 2-31 shows 8-bit memory devices configuring 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.

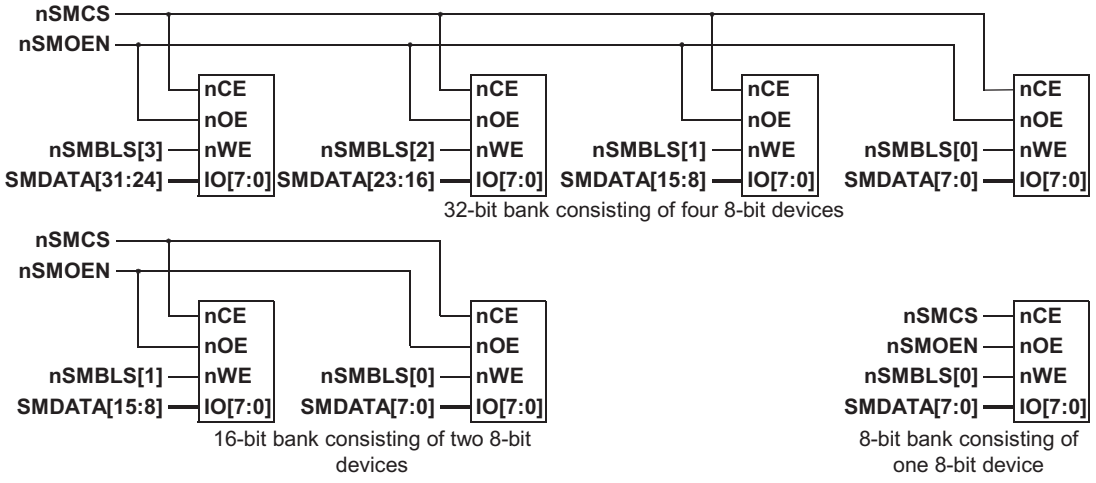



**Figure 2-31 Memory banks constructed from 8-bit memory**

**Note**

The **nSMWEN** signal from the SSMC 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. This enables you to define the external bus for at least the width of the accessed memory.

### Accesses to 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 you set the RBLE bit to 1 within the respective Bank Control Register. This asserts all **nSMBLS[3:0]** lines LOW during a read access to that particular bank, because during a read, you must select all bytes of the devices to avoid un-driven byte lanes on the read data value.

For 16 and 32-bit wide memory devices, byte select signals exist and you must control these appropriately, as Figure 2-32 and Figure 2-33 show.

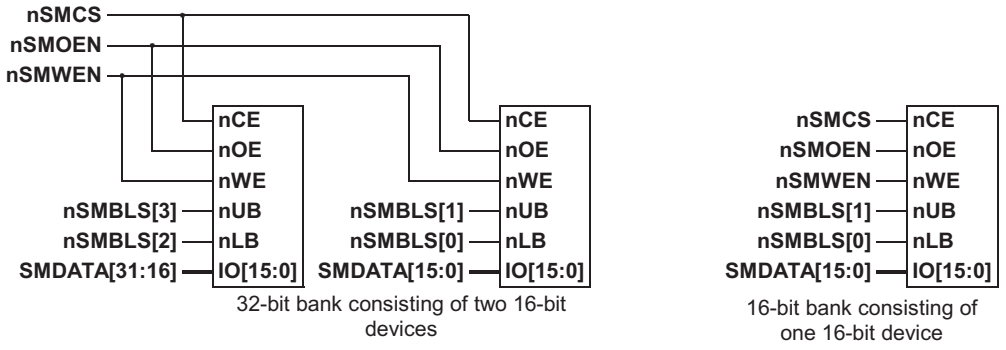



**Figure 2-32 Memory banks constructed from 16-bit memory**

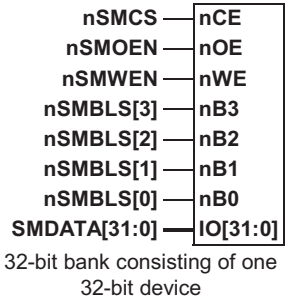



**Figure 2-33 Memory banks constructed from 32-bit memory**

Figure 2-34 on page 2-39 shows connections for a typical memory system with different data width memory devices.

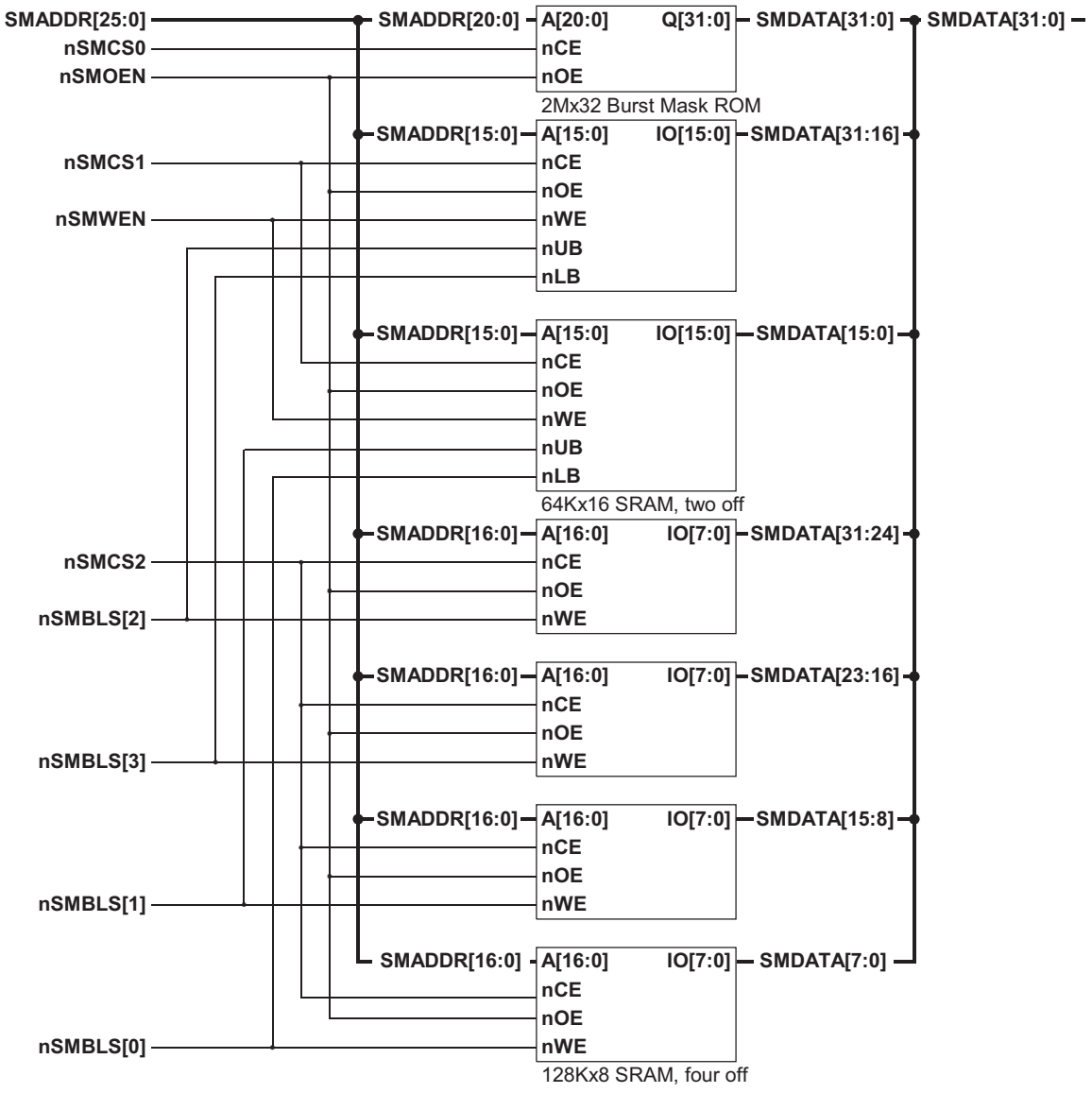



**Figure 2-34 Typical memory connection**

### Elimination of floating bytes on the external interface

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

**nSMDATAEN[3:0]** controls the input/output pads cells for the external write data bus lines as Table 2-4 shows. Data is driven out on **SMDATAOUT** when **nSMDATAEN** is asserted LOW.

**Table 2-4 SMDATAOUT controlled by nSMDATAEN**

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

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

Table 2-5 on page 2-41 to Table 2-10 on page 2-44 list how the internal databuses **HRDATASMC[31:0]** and **HWDATASMC[31:0]** map onto the external databus. The two internal databuses are listed as **HDATA[31:0]** in the tables. This mapping of the data is affected by:

- the external databus width
- big/little-endian operation
- size of the AHB access (word, halfword, or byte as determined by **HSIZE[1:0]**)
- alignment within word of byte and halfword accesses, as controlled by **HADDR[1:0]**.

Table 2-5 on page 2-41 to Table 2-10 on page 2-44 lists how byte lane control and data bus steering is achieved for all combinations of:

- big-endian configuration
- little-endian configuration
- bus width
- **HSIZE[1:0]**
- **HADDR[1:0]**.

For each access, the external bits on the **SMDATA** bus that map to the internal AHB bus **HDATASMC** are listed.

**Note**

For read transfers, ensure that the RBLE bit is cleared to zero in the respective Bank Control Register. This forces all **nSMBLS[3:0]** lines to be deasserted HIGH and enables you to define the external bus for at least the width of the accessed memory. See *Bank Control Registers 0-7* on page 3-10.

Table 2-5 lists the little-endian read accesses for an 8-bit external bus.

**Table 2-5 Little-endian read, 8-bit external bus**

| Access: Little-endian, 8-bit external bus | | | | External data mapping onto system data bus SMDATA[31:0] onto HDATASMC | | | |
|---|---|---|---|---|---|---|---|
| **Internal width** | **HSIZE[1:0]** | **HADDR[1:0]** | **SMADDR[1:0]** | **[31:24]** | **[23:16]** | **[15:8]** | **[7:0]** |
| Word (4 transfers) | 10 | xx | 11 | 7:0 | - | - | - |
| | 10 | xx | 10 | - | 7:0 | - | - |
| | 10 | xx | 01 | - | - | 7:0 | - |
| | 10 | xx | 00 | - | - | - | 7:0 |
| Halfword (2 transfers) | 01 | 1x | 11 | 7:0 | - | - | - |
| | 01 | 1x | 10 | - | 7:0 | - | - |
| Halfword (2 transfers) | 01 | 0x | 01 | - | - | 7:0 | - |
| | 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 2-6 lists the little-endian read accesses for a 16-bit external bus.

**Table 2-6 Little-endian read, 16-bit external bus**

| Access: Little-endian, 16-bit external bus | | | | External data mapping onto system data bus SMDATA[31:0] onto HDATASMC | | | |
|---|---|---|---|---|---|---|---|
| **Internal width** | **HSIZE[1:0]** | **HADDR[1:0]** | **SMADDR[1:0]** | **[31:24]** | **[23:16]** | **[15:8]** | **[7:0]** |
| Word (2 transfers) | 10 | xx | 1x | 15:8 | 7:0 | - | - |
| | 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 2-7 lists the little-endian read accesses for a 32-bit external bus.

**Table 2-7 Little-endian read, 32-bit external bus**

| Access: Little-endian, 32-bit external bus | | | External data mapping onto system data bus SMDATA[31:0] onto HDATASMC | | | |
|---|---|---|---|---|---|---|
| **Internal width** | **HSIZE[1:0]** | **HADDR[1:0]** | **[31:24]** | **[23:16]** | **[15:8]** | **[7:0]** |
| Word | 10 | xx | 31:24 | 23:16 | 15:8 | 7:0 |
| Halfword | 01 | 1x | 31:24 | 23:16 | - | - |
| Halfword (2 transfers) | 01 | 0x | - | - | 15:8 | 7:0 |
| Byte | 00 | 11 | 31:24 | - | - | - |
| Byte | 00 | 10 | - | 23:16 | - | - |
| Byte | 00 | 01 | - | - | 15:8 | |
| Byte | 00 | 00 | - | - | - | 7:0 |

Table 2-8 lists the big-endian read accesses for an 8-bit external bus.

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

| Access: Big-endian, 8-bit external bus | | | | External data mapping onto system data bus SMDATA[31:0] onto HDATASMC | | | |
|---|---|---|---|---|---|---|---|
| **Internal width** | **HSIZE[1:0]** | **HADDR[1:0]** | **SMADDR[1:0]** | **[31:24]** | **[23:16]** | **[15:8]** | **[7:0]** |
| Word (4 transfers) | 10 | xx | 11 | - | - | - | 7:0 |
| | 10 | xx | 10 | - | - | 7:0 | - |
| | 10 | xx | 01 | - | 7:0 | - | - |
| | 10 | xx | 00 | 7:0 | - | - | - |
| Halfword (2 transfers) | 01 | 1x | 11 | - | - | - | 7:0 |
| | 01 | 1x | 10 | - | - | 7:0 | - |
| Halfword (2 transfers) | 01 | 0x | 01 | - | 7:0 | - | - |
| | 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 2-9 lists the big-endian read accesses for a 16-bit external bus.

**Table 2-9 Big-endian read, 16-bit external bus**

| Access: Big-endian, 16-bit external bus | | | | External data mapping onto system data bus SMDATA[31:0] onto HDATASMC | | | |
|---|---|---|---|---|---|---|---|
| **Internal width** | **HSIZE[1:0]** | **HADDR[1:0]** | **SMADDR[1:0]** | **[31:24]** | **[23:16]** | **[15:8]** | **[7:0]** |
| Word (2 transfers) | 10 | xx | 1x | - | - | 15:8 | 7:0 |
| | 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 |

**Table 2-9 Big-endian read, 16-bit external bus (continued)**

| Access: Big-endian, 16-bit external bus | | | | External data mapping onto system data bus SMDATA[31:0] onto HDATASMC | | | |
|---|---|---|---|---|---|---|---|
| **Internal width** | **HSIZE[1:0]** | **HADDR[1:0]** | **SMADDR[1:0]** | **[31:24]** | **[23:16]** | **[15:8]** | **[7:0]** |
| Byte | 00 | 10 | 1x | - | - | 15:8 | - |
| Byte | 00 | 01 | 0x | - | 7:0 | - | - |
| Byte | 00 | 00 | 0x | 15:8 | - | - | - |

Table 2-10 lists the big-endian read accesses for a 32-bit external bus.

**Table 2-10 Big-endian read, 32-bit external bus**

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

### 2.3.3 Clock feedback in SSMC

Figure 2-35 on page 2-45 shows the pad interface block. This registers data address and control signals. A bypass mode is supplied for using asynchronous devices. Each byte of the data bus is registered using one of the four feedback clocks. The control signals are registered using a delayed version of **SMMEMCLK**.

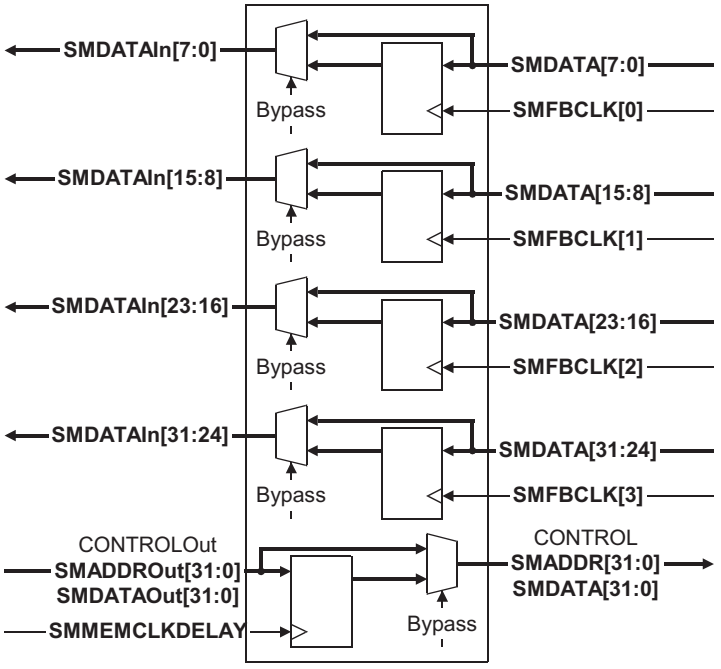



**Figure 2-35 Pad interface**

The external clocks **SMCLK[3:0]** run continuously unless the SMClockEn bit in the SSMC Control Register, SSMCCR, only enables the clock during accesses. Figure 2-36 shows how **SMCLK[3:0]** is derived.


**Figure 2-36 Clock gating**

**Note**

In Figure 2-36, the **ClkStpd** signal is a control signal from the SSMC core.

The delay between **SMMEMCLKDELAY** and **SMMEMCLK** must be sufficient to meet the setup and hold requirements for the memory devices that you are using. You must determine these delays. A simple way to provide **SMMEMCLKDELAY** is to use an inverted version of **SMCLK**.

### Example of 8-bit memory device interconnection

Figure 2-37 shows 8-bit memory interconnection. Group 0, chip select 0, has four x8 memory devices providing a 32-bit databus width. Group 1 has two 8-bit memory devices providing a 16-bit databus width.

**SMCLK[0]** clocks the memory devices providing the low 8 bits of data. **SMFBCLK[0]** is generated from **SMCLK[0]** and registers the low 8 bits of the databus for reads as it enters the SSMC.

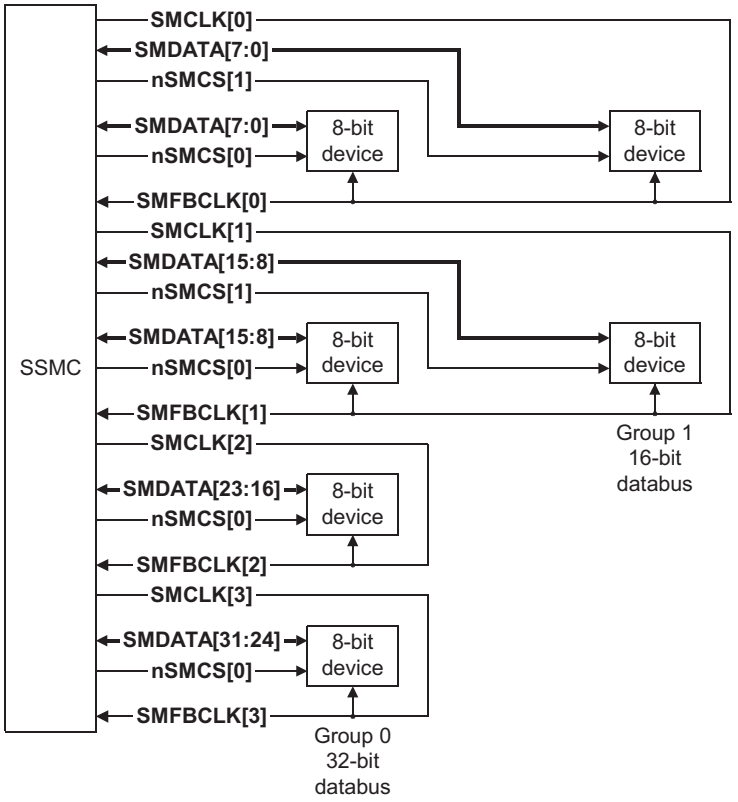



**Figure 2-37 8-bit memory device interconnection example**

### Example of 16-bit memory device interconnection

Figure 2-38 on page 2-47 shows 16-bit memory interconnection. Group 0, chip select 0, has two x16 memory devices providing a 32-bit databus width. Group 1 has one 16-bit memory device providing 16-bit databus width.

**SMCLK[0]** clocks the memory devices providing the low 16 bits of data. **SMFBCLK[0]** and **SMFBCLK[1]** are generated from **SMCLK[0]**, and register the low 16 bits of the databus for reads as it enters the SSMC.

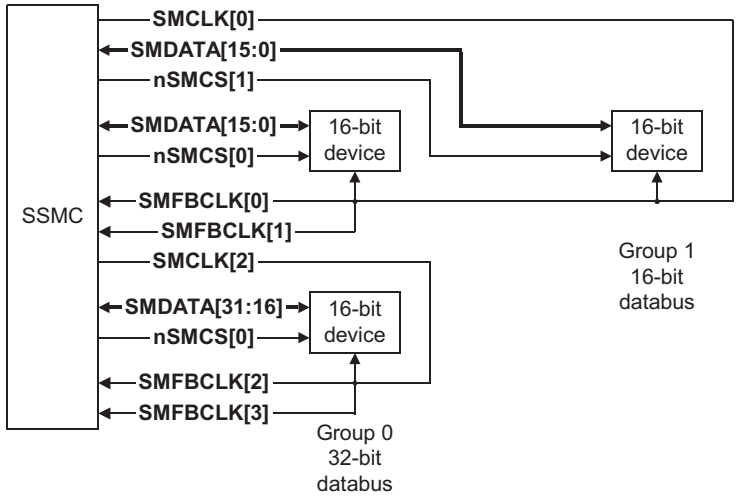



**Figure 2-38 16-bit memory device interconnection example**

**Note**

**SMCLK[1]** and **SMCLK[3]** are unused in this example. You can use these clock signals to clock more groups of memory devices to spread the load on the clocks.

### Example of 32-bit memory device interconnection

Figure 2-39 on page 2-48 shows x32 memory interconnection. Group 0, chip select 0, has one x32 memory devices providing a 32-bit databus width. Group 1 has one x32 memory device providing 32-bit databus width.

**SMCLK[0]** clocks the memory devices providing the 32 bits of data. **SMFBCLK[0]** and **SMFBCLK[1]**, **SMCLK[2]** and **SMCLK[3]** are generated from **SMCLK[0]** and registers the 32 bits of the databus for reads as it enters the SSMC.

**Note**

**SMCLK[1]**, **SMCLK[2]**, and **SMCLK[3]** are unused in this example. You can use these clock signals to clock more groups of memory devices to spread the load on the clocks.

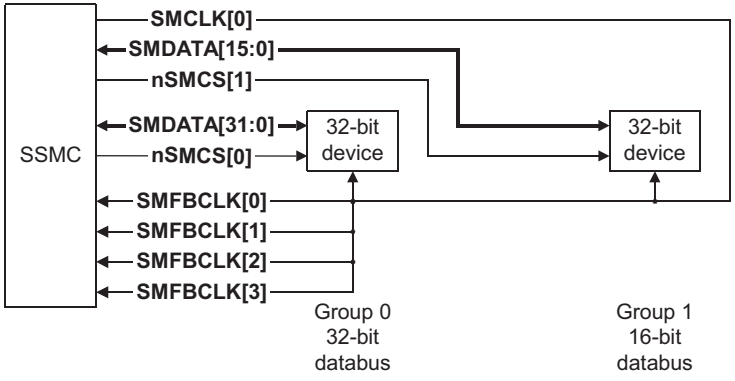



**Figure 2-39 32-bit memory device interconnection example**

### 2.3.4 Example of system with single output clock

In systems that are pad limited, you can use a single clock. One of the output clocks, **SMCLK[0]**, is connected to the output of a bidirectional pad. You can leave the other three clocks unconnected. The pad output enable is tied LOW so that the pad is always in output mode. The input from the pad is connected to **SMFBCLK[3-0]** to provide four feedback clocks. Figure 2-40 shows this.

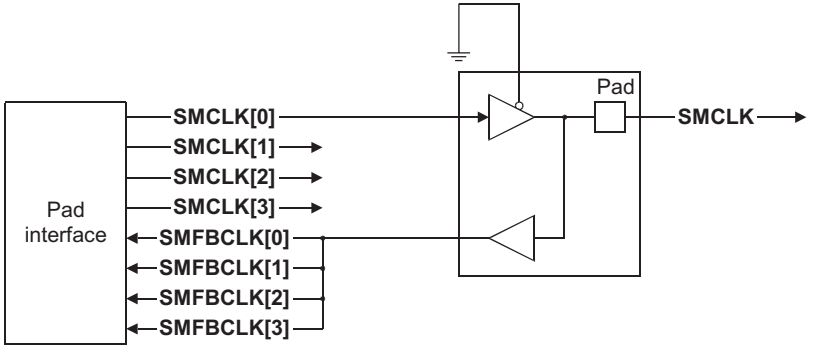



**Figure 2-40 Connection of SSMC to pad when using a single clock**

## 2.4 Slave interface connection to the AHB

There are two separate AHB slave interfaces to connect the SMC to the AHB system bus:

- one gives access to the Control Registers
- one enables access to the memory.

**Note**

This enables the interfaces to be on separate buses if required. You must take care in this case that the registers are not accessed during a memory access.

The memory interface has eight separate **HSEL** signals. You can use these to define the memory map as required. See *Memory shadowing* on page 2-50 for details of how to use the separate **HSEL** lines.

## 2.5 Memory shadowing

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

**Note**

The memory controller resets to asynchronous mode. Software must configure the interface to synchronous mode and also configure the memory device. You might require additional memory in the system to contain this code because it is not possible to configure SSMC and external memory at the same time. If the code is being executed from the memory bank being configured then the system can fail.

This section describes:

- *Booting from ROM after reset*
- *External bank SMCS7 size configuration* on page 2-52.

### 2.5.1 Booting from ROM after reset

An **HSEL** signal is available for each memory bank. This enables you to configure the AHB decoder to meet the system requirement. The CPU starts executing code from address `0x00000000`, and therefore, you must take care to ensure that the boot code (ROM) is available from this location.

After the system has booted it is normal to remap the memory system to make RAM available at address `0x00000000` for the vector table. The ROM is then available at an alternative memory address.

The SSMC does not support a remap function directly. If you require remapping then you must use a system decoder or similar. On reset, the decoder interprets the access to `0x0` as access to ROM, the alias of the real ROM address, to enable the reset vector to be present at power-on. Your system must implement a **REMAP** signal function or similar.

The decoder then does the following or similar:

```
case ADDR(31:24) is
when “0x00”
if REMAP = ‘0’ then
select ROM
else
select SRAM
when “0x0F”
select ROM
```

When the system powers up:

```
nFetch RESET vector at 0x00000000
n(alias of ROM)
nExecute RESET vector
LDR PC, = 0x0F000004
nreal address of next ROM instruction
nWrite to REMAP register
set REMAP = '1'
nRun Exception Vector copy routine
copy (slow) ROM to (fast) SRAM
```

You must use bank 7 for external boot ROM. Configure bank 7 at reset using external signals, as described in *External bank SMCS7 size configuration* on page 2-52.

Figure 2-41 shows one possible memory map implementation. This shows the reset memory map with the boot code contained in bank 7 accessible at address `0x00000000`. You can also access this memory at a higher address. This is required to enable the software to branch to this location prior to remapping the memory system. After remapping the memory system the RAM is available from bank 0. This is overlapped by a small internal RAM. This enables you to place the vector code in the fast internal RAM, and enables easy flow from internal to external memory in a continuous address space. This enables software to continue to execute without having to branch to a different location even if the code overflows the internal RAM without having to branch to a different location.

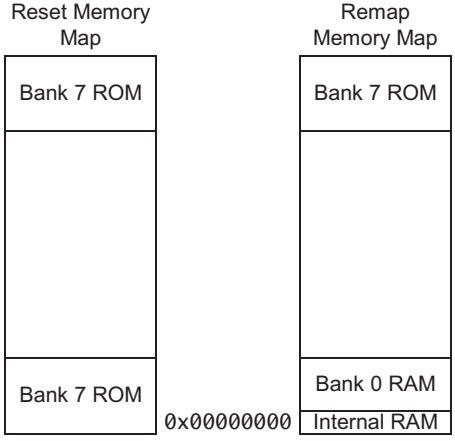



**Figure 2-41 Example memory map**

### 2.5.2 External bank SMCS7 size configuration

The SSMC enables you to configure the size of bank 7 (selected with **SMCS[7**]) at reset using the external control pins **SMMWCS7[1:0]**. This feature enables you to chose the boot memory size at reset. Bank 7 can therefore always have the correct setting, and no additional software programming of the Bank 7 Control Register is required. However, the MW bits of SMBCR7 are still programmable, the same as the rest of the bank configuration registers.

**Note**

All other memory bank sizes have default settings that are set in the hardware, but software can program them to the values they require.

Table 2-11 lists the configuration of the bits used.

**Table 2-11 External size configuration values for bank 7**

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

**Note**

The control bit RBLE in the SMBCRx Registers defaults to 0. This enables writes to 8-bit devices only. If you require other device widths, then software must set this bit before the first write to the memory.

## 2.6 Test interface controller

The TIC must not be the default master on the AHB bus. The TIC is only synthesized to 10MHz. If the TIC is in function operation mode (for example, if it is the default bus master) then it is not able to respond in the required time. This frequency restriction does not cause problems with test mode because this mode is usually used at a lower speed.

You can make the TIC the highest priority bus master. For details of the TIC, see the *AMBA Specification Rev 2.0*.

## 2.7 Using the SSMC with an EBI

The SSMC is designed to enable an EBI to control the switching of the external data and address buses. For example, if you use the SSMC with an SDRAM controller that requires a bus multiplexor that can handle the necessary synchronization re-timing.

You must tie the **SMEXTBUSMUX** input HIGH to signal to use the EBI. This enables the external request and grant ports.

You must connect the SSMC and TIC bus request and grant ports to the EBI to ensure that, if you use the TIC, you connect it to the highest priority port because it must always be able to enter test mode. You must connect the SSMC address, data, and data enable to the relevant port on the EBI. You must also connect the **SMBUSBACKOFFEBI** signal to the EBI. The EBI uses this signal to tell the SSMC to release the external bus. There is no equivalent signal for the TIC because it is not necessary to release the bus when in test mode.

The handshaking between the EBI and the memory controller consists of a three-wire interface, **EBIREQ**, **EBIGNT**, and **EBIBACKOFF**, all active HIGH:

- **EBIREQ** signals are asserted by memory controllers to indicate that they require external bus access
- the respective arbitrated **EBIGNT** is issued to the highest priority memory controller
- **EBIBACKOFF** is output by the EBI to signal that the memory controller must complete the current transfer and release the bus.

The EBI arbitration scheme keeps track of the memory controller that is currently granted and waits for the transaction from that memory controller to finish (**EBIREQ** taken LOW by the memory controller) before it grants the next memory controller. If a higher priority memory controller requests the bus then the **EBIBACKOFF** signal tells the currently granted memory controller to terminate the current transfer as soon as possible. Memory controller 1 has the highest priority and memory controller 3 the lowest priority.

When two memory controllers simultaneously request the EBI, the controller with the higher priority is granted use.

The EBI, as a peripheral, relies on the memory controllers to release their external requests for the external bus when they are idle, because it has no other knowledge of when a transfer starts or completes.

Figure 2-42 on page 2-55 shows a simple handshake example. In this example, a device requests the external bus and is immediately granted because no other devices are requesting the bus.

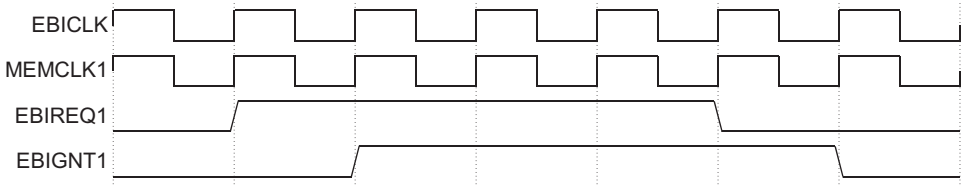



**Figure 2-42 EBIREQ and EBIGNT signals when access granted immediately**

If a higher priority device requests the bus, when a lower priority device is in control of the external bus then the **EBIBACKOFF** signal tells the lower priority device to release the bus as soon as possible. Figure 2-43 shows an example of this.

In Figure 2-43, a higher priority device requests the bus, shortly after a device has been granted the bus. The **EBIBACKOFF** signal tells the device to end the access early. Device 1 is granted the bus and completes its transfer. When the transfer is complete then device 2 is granted the bus and completes the transfer that was interrupted. The **EBIREQ2** signal must be LOW for at least one clock cycle and after this, you can reassert it.

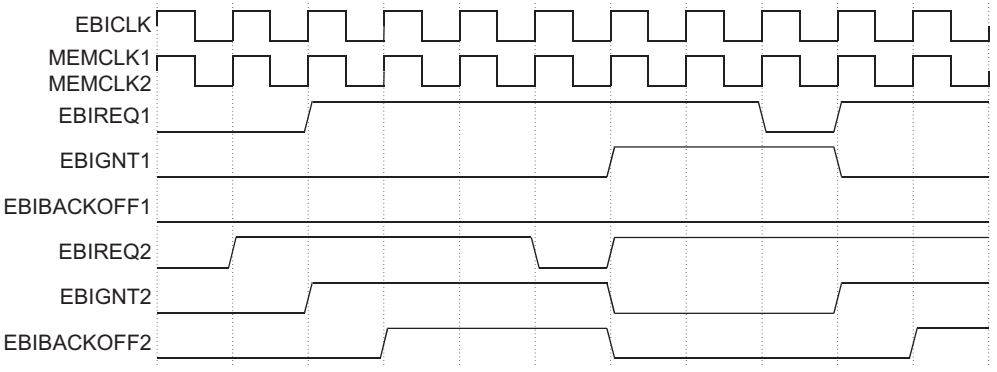



**Figure 2-43 Example use of EBIBACKOFF signal (EBICLK=MemCLK1=MemCLK2)**

# Chapter 3
# **Programmer's Model**

This chapter describes the SSMC registers and provides the details that you require when programming the controller. It contains the following:

- *About the programmer's model* on page 3-2
- *Register summary* on page 3-3
- *Register descriptions* on page 3-7.

## 3.1 About the programmer's model

You can configure the base addresses for the SSMC Bank Configuration Registers and memory banks to suit each particular system implementation. Use constant definitions in the *Hardware Description Language* (HDL) code for the AMBA address decoder to configure the base addresses. These base addresses are not software-programmable.

The SSMC registers and memory banks have fixed address offsets from the base addresses. These are listed inTable 3-1 on page 3-3.

The SSMC only supports 32-bit word accesses to all registers. Accesses made using any other size generate an ERROR response on **HRESP**.

## 3.2 Register summary

Table 3-1 lists the SSMC control and status registers.

**Table 3-1 Register summary**

| Name | Base offset | Type | Reset value | Description |
|---|---|---|---|---|
| SMBIDCYR0 | 0x00 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR0 | 0x04 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR0 | 0x08 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |
| SMBWSTOENR0 | 0x0C | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR0 | 0x10 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR0 | 0x14 | R/W | 0x303020 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR0 | 0x18 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |
| SMBWSTBRDR0 | 0x1C | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| SMBIDCYR1 | 0x20 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR1 | 0x24 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR1 | 0x28 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |
| SMBWSTOENR1 | 0x2C | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR1 | 0x30 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR1 | 0x34 | R/W | 0x303000 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR1 | 0x38 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |
| SMBWSTBRDR1 | 0x3C | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| SMBIDCYR2 | 0x40 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR2 | 0x44 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR2 | 0x48 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |

**Table 3-1 Register summary (continued)**

| Name | Base offset | Type | Reset value | Description |
|---|---|---|---|---|
| SMBWSTOENR2 | 0x4C | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR2 | 0x50 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR2 | 0x54 | R/W | 0x303010 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR2 | 0x58 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |
| SMBWSTBRDR2 | 0x5C | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| SMBIDCYR3 | 0x60 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR3 | 0x64 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR3 | 0x68 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |
| SMBWSTOENR3 | 0x6C | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR3 | 0x70 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR3 | 0x74 | R/W | 0x303000 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR3 | 0x78 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |
| SMBWSTBRDR3 | 0x7C | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| SMBIDCYR4 | 0x80 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR4 | 0x84 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR4 | 0x88 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |
| SMBWSTOENR4 | 0x8C | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR4 | 0x90 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR4 | 0x94 | R/W | 0x303020 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR4 | 0x98 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |

**Table 3-1 Register summary (continued)**

| Name | Base offset | Type | Reset value | Description |
|---|---|---|---|---|
| SMBWSTBRDR4 | 0x9C | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| SMBIDCYR5 | 0xA0 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR5 | 0xA4 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR5 | 0xA8 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |
| SMBWSTOENR5 | 0xAC | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR5 | 0xB0 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR5 | 0xB4 | R/W | 0x303020 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR5 | 0xB8 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |
| SMBWSTBRDR5 | 0xBC | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| SMBIDCYR6 | 0xC0 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR6 | 0xC4 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR6 | 0xC8 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |
| SMBWSTOENR6 | 0xCC | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR6 | 0xD0 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR6 | 0xD4 | R/W | 0x303010 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR6 | 0xD8 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |
| SMBWSTBRDR6 | 0xDC | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| SMBIDCYR7 | 0xE0 | R/W | 0xF | See *Bank Idle Cycle Control Registers 0-7* on page 3-7 |
| SMBWSTRDR7 | 0xE4 | R/W | 0x1F | See *Bank Read Wait State Control Registers 0-7* on page 3-7 |
| SMBWSTWRR7 | 0xE8 | R/W | 0x1F | See *Bank Write Wait State Control Registers 0-7* on page 3-8 |

**Table 3-1 Register summary (continued)**

| Name | Base offset | Type | Reset value | Description |
|---|---|---|---|---|
| SMBWSTOENR7 | 0xEC | R/W | 0x0 | See *Bank Output Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBWSTWENR7 | 0xF0 | R/W | 0x1 | See *Bank Write Enable Assertion Delay Control Registers 0-7* on page 3-9 |
| SMBCR7 | 0xF4 | R/W | 0x303000 | See *Bank Control Registers 0-7* on page 3-10 |
| SMBSR7 | 0xF8 | R/W | 0x0 | See *Bank Status Registers 0-7* on page 3-15 |
| SMBWSTBRDR7 | 0xFC | R/W | 0x1F | See *Bank Burst Read Wait Delay Control Registers 0-7* on page 3-16 |
| Reserved | 0x100-0x1FC | - | - | Reserved. |
| SSMCSR | 0x200 | RO | 0x0 | See *SSMC Status Register* on page 3-17 |
| SSMCCR | 0x204 | R/W | 0x1 | See *SSMC Control Register* on page 3-17 |
| SSMCITCR | 0x208 | R/W | 0x0 | See *SSMC Test Control Register* on page 4-3 |
| SSMCITIP | 0x20C | R/W | -[a] | See *SSMC Test Input Register* on page 4-4 |
| SSMCITOP | 0x210 | R/W | -[a] | See *SSMC Test Output Register* on page 4-3 |
| Reserved | 0x214-0xFDC | - | - | Reserved. |
| SSMCPeriphID0 | 0xFE0 | RO | 0x93 | See *Peripheral Identification Register 0* on page 3-19 |
| SSMCPeriphID1 | 0xFE4 | RO | 0x10 | See *Peripheral Identification Register 1* on page 3-20 |
| SSMCPeriphID2 | 0xFE8 | RO | 0x14 | See *Peripheral Identification Register 2* on page 3-20 |
| SSMCPeriphID3 | 0xFEC | RO | 0x00 | See *Peripheral Identification Register 3* on page 3-20 |
| SSMCPCellID0 | 0xFF0 | RO | 0x0D | See *PrimeCell Identification Register 0* on page 3-21 |
| SSMCPCellID1 | 0xFF4 | RO | 0xF0 | See *PrimeCell Identification Register 1* on page 3-22 |
| SSMCPCellID2 | 0xFF8 | RO | 0x05 | See *PrimeCell Identification Register 2* on page 3-22 |
| SSMCPCellID3 | 0xFFC | RO | 0xB1 | See *PrimeCell Identification Register 3* on page 3-22 |

a. System-dependent

## 3.3 Register descriptions

This section contains descriptions of the SSMC registers. Table 3-1 on page 3-3 provides cross references to individual registers.

### 3.3.1 Bank Idle Cycle Control Registers 0-7

The eight read/write SMBIDCYRx Registers are the bank 0-7 idle cycle control registers. Each register is identical in structure. Figure 3-1 shows the bit assignments of an SMBIDCYRx Register.

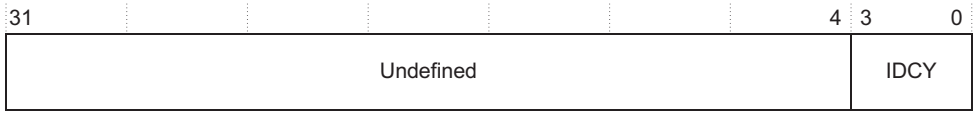



**Figure 3-1 SMBIDCYRx Register bit assignments**

Table 3-2 lists the bit assignments of an SMBIDCYRx Register.

**Table 3-2 SMBIDCYRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:4] | - | Read undefined. Write as zero. |
| [3:0] | IDCY | Idle or turnaround cycles. Defaults to 1111 at reset.<br>This field controls the number of bus turnaround cycles added between read and write accesses to prevent bus contention on the external memory data bus.<br>Turnaround time = IDCY x $t_{HCLK}$[a] |

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

### 3.3.2 Bank Read Wait State Control Registers 0-7

The eight read/write SMBWSTRDRx Registers are the bank 0-7 read wait state control registers. Each register is identical in structure. Figure 3-2 shows the bit assignments of an SMBWSTRDRx Register.

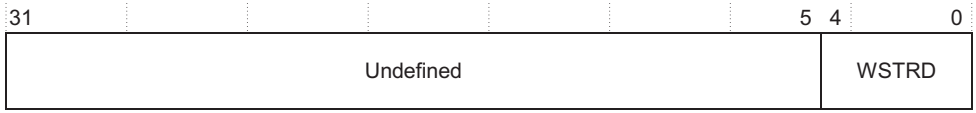



**Figure 3-2 SMBWSTRDRx Register bit assignments**

Table 3-3 lists the bit assignments of an SMBWSTRDRx Register.

**Table 3-3 SMBWSTRDRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:5] | - | Read undefined. Write as zero. |
| [4:0] | WSTRD | Read wait state. Defaults to 11111 at reset.<br>For SRAM and ROM, the WSTRD field controls the number of wait states for read accesses, and the external wait assertion timing for reads.<br>For burst ROM, the WSTRD field controls the number of wait states for the first read access only.<br>Wait state time = WSTRD x $t_{HCLK}$[a] |

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

### 3.3.3 Bank Write Wait State Control Registers 0-7

The eight read/write SMBWSTWRRx Registers are the bank 0-7 write wait state control registers. Each register is identical in structure. Figure 3-3 shows the bit assignments of an SMBWSTWRRx Register.

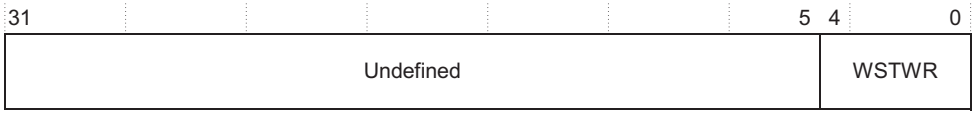



**Figure 3-3 SMBWSTWRRx Register bit assignments**

Table 3-4 lists the bit assignments of an SMBWSTWRRx Register.

**Table 3-4 SMBWSTWRRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:5] | - | Read undefined. Write as zero. |
| [4:0] | WSTWR | Write wait state. Defaults to 11111 at reset.<br>For SRAM, the WSTWR field controls the number of wait states for write accesses, and the external wait assertion timing for writes.<br>Wait state time = WSTWR x $t_{HCLK}$[a]<br>WSTWR does not apply to read-only devices such as ROM. |

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

### 3.3.4 Bank Output Enable Assertion Delay Control Registers 0-7

The eight read/write SMBWSTOENRx Registers are the bank 0-7 output enable assertion delay control registers. Each register is identical in structure. Figure 3-4 shows the bit assignments of an SMBWSTOENRx Register.

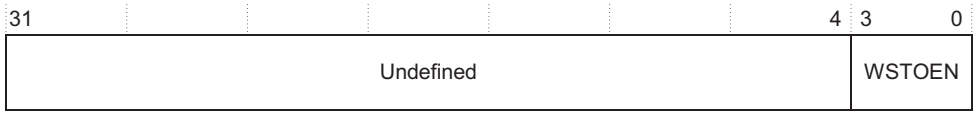



**Figure 3-4 SMBWSTOENRx Register bit assignments**

Table 3-5 lists the bit assignments of an SMBWSTOENRx Register.

**Table 3-5 SMBWSTOENRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:4] | Reserved | Read undefined. Write as zero. |
| [3:0] | WSTOEN | 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 read/write SMBWSTWENRx Registers are the bank 0-7 write enable assertion delay control registers. Each register is identical in structure. Figure 3-5 shows the bit assignments of an SMBWSTWENRx Register.

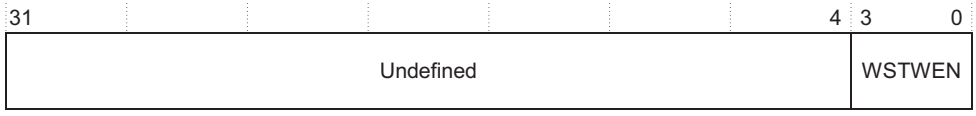



**Figure 3-5 SMBWSTWENRx Register bit assignments**

Table 3-6 lists the bit assignments of an SMBWSTWENRx Register.

**Table 3-6 SMBWSTWENRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:4] | - | Read undefined. Write as zero. |
| [3:0] | WSTWEN | Write enable assertion delay from chip select assertion. Defaults to 0001 at reset. |

### 3.3.6 Bank Control Registers 0-7

The eight read/write SMBCRx Registers are the bank 0-7 control registers. Each register is identical in structure. Table 3-7 lists the memory bank default external memory width at reset.

**Table 3-7 PrimeCell SSMC reset default memory width**

| PrimeCell SSMC 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 |

**Note**

If the system boots from a Bank 7 external ROM device, then you must configure the memory width for that bank using the external control pins **SMMWCS7[1:0]** to provide the correct default external memory width at reset.

Figure 3-6 on page 3-11 shows the bit assignments of an SMBCRx Register.

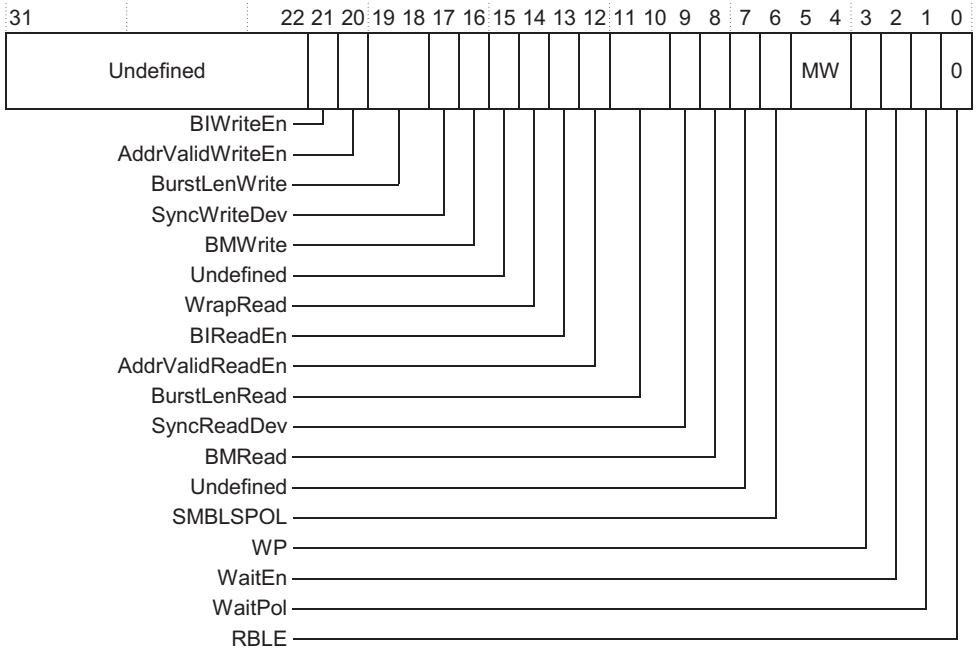



**Figure 3-6 SMBCRx Register bit assignments**

Table 3-8 lists the bit assignments of an SMBCRx Register.

**Table 3-8 SMBCRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:22] | - | Read undefined. Write as zero. |
| [21] | BIWriteEn | Controls the behavior of the **SMBAA** signal during write operations:<br>0 **SMBAA**=1 at all times<br>1 Signal active for synchronous read accesses (default). |
| [20] | AddrValidWriteEn | Controls the behavior of the signal **SMADDRVALID** during write operations:<br>0 Signal always HIGH<br>1 Signal active for asynchronous and synchronous write accesses (default). |

**Table 3-8 SMBCRx Register bit assignments (continued)**

| Bits | Name | Description |
|---|---|---|
| [19:18] | BurstLenWrite | Burst transfer length. Sets the number of sequential transfers that the burst device supports for a write:<br>00 4-transfer burst (default)<br>01 8-transfer burst<br>10 Reserved<br>11 Continuous burst (synchronous only). |
| [17] | SyncWriteDev | Synchronous access capable device connected. Access the device using synchronous accesses for writes:<br>0 Asynchronous device (default).<br>1 Synchronous device. |
| [16] | BMWrite | Burst mode write:<br>0 Nonburst writes to memory devices (default at reset)<br>1 Burst mode writes to memory devices. |
| [15] | - | Read undefined. Write as zero. |
| [14] | WrapRead | Enables the wrapping burst feature from external memory. This is to support eight word wrapping bursts only. Only valid for burst 16 from 16-bit external memory and burst 8 from 32-bit external memory.<br>0 Disabled (default).<br>1 Enabled.<br>**Note**<br>The PL093 SMC supports wrapping reads, but does not support wrapping writes. |
| [13] | BIReadEn | Controls the behavior of the **SMBAA** signal during read operations.<br>0 **SMBAA**=1 at all times.<br>1 Signals active for synchronous read accesses (default). |
| [12] | AddrValidReadEn | Controls the behavior of the signal **SMADDRVALID** during read operations:<br>0 Signal always HIGH.<br>1 Signal active for asynchronous and synchronous read accesses (default). |
| [11:10] | BurstLenRead | Burst transfer length. Sets the number of sequential transfers that the burst device supports for a read:<br>00 4-transfer burst.<br>01 8-transfer burst.<br>10 16-transfer burst.<br>11 Continuous burst (synchronous only). |

**Table 3-8 SMBCRx Register bit assignments (continued)**

| Bits | Name | Description |
|---|---|---|
| [9] | SyncReadDev | Synchronous access capable device connected. Access the device using synchronous accesses for reads:<br>0 Asynchronous device (default).<br>1 Synchronous device. |
| [8] | BMRead | Burst mode read and asynchronous page mode:<br>0 Nonburst reads from memory devices (default at reset).<br>1 Burst mode reads from memory devices. |
| [7] | - | Read undefined. Write as zero. |
| [6] | SMBLSPOL | Polarity of signal **nSMBLS**:<br>0 Signal is active LOW (default).<br>1 Signal is active HIGH. |
| [5:4] | MW | Memory width:<br>00 8-bit.<br>01 16-bit.<br>10 32-bit.<br>11 Reserved.<br>Defaults to different values at reset for each bank, see Table 3-8 on page 3-12. |
| [3] | WP | Write protect:<br>0 No write protection, for example, SRAM or write enabled Flash (default at reset).<br>1 Device is write protected, for example, ROM, burst ROM, read-only Flash, or SRAM. |

**Table 3-8 SMBCRx Register bit assignments (continued)**

| Bits | Name | Description |
|---|---|---|
| [2] | WaitEn | External memory controller wait signal enable:<br>0 The PrimeCell SSMC is not controlled by the external wait signal (default at reset).<br>1 The PrimeCell SSMC looks for the external wait input signal, **SMWAIT**. |
| [1] | WaitPol | Polarity of the external wait input for activation:<br>0 The **SMWAIT** signal is active LOW (default at reset).<br>1 The **SMWAIT** signal is active HIGH. |
| [0] | RBLE | Read byte lane enable:<br>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 you must deassert it during a read (default at reset). The **nSMBLS** signals act as write enables in this configuration.<br>1 **nSMBLS[3:0]** all asserted LOW during system reads from external memory. This is for 16 or 32-bit devices where you use the separate write enable signal, and you must hold the byte lane selects asserted during a read. The **nSMWEN** signal acts as the write enable in this configuration. |

### SMBCRx example configurations

Table 3-9 lists some example register configurations for the SMBCRx Registers.

**Table 3-9 SMBCRx example configurations**

| Bits | Name | Asynchronous 4-beat burst SRAM 4x8-bit devices | Synchronous memory in asynchronous mode (reset), nonburst mode | Synchronous memory in synchronous read mode, asynchronous writes, 4-beat burst | Synchronous memory in synchronous read mode, asynchronous writes, 4-beat burst read |
|---|---|---|---|---|---|
| [31:22] | Reserved | 0 | 0 | 0 | 0 |
| [21] | BIWriteEn | 0 | 0 | 0 | 0 |
| [20] | AddrValidWriteEn | 0 | 1 | 1 | 1 |
| [19:18] | BurstLenWrite | 00 | 00 | 00 | 00 |
| [17] | SyncWriteDev | 0 | 0 | 0 | 1 |

**Table 3-9 SMBCRx example configurations (continued)**

| Bits | Name | Asynchronous 4-beat burst SRAM 4x8-bit devices | Synchronous memory in asynchronous mode (reset), nonburst mode | Synchronous memory in synchronous read mode, asynchronous writes, 4-beat burst | Synchronous memory in synchronous read mode, asynchronous writes, 4-beat burst read |
|---|---|---|---|---|---|
| [16] | BMWrite | 1 | 0 | 0 | 0 |
| [15:13] | Reserved | 000 | 000 | 000 | 000 |
| [12] | AddrValidReadEn | 0 | 1 | 1 | 1 |
| [11:10] | BurstLenRead | 00 | 00 | 00 | 00 |
| [9] | SyncReadDev | 0 | 0 | 1 | 1 |
| [8] | BMRead | 1 | 0 | 1 | 1 |
| [7] | Reserved | 00 | 00 | 00 | 00 |
| [6] | SMBLSPOL | 0 | 0 | 0 | 0 |
| [5:4] | MW | 10 | 10 | 10 | 10 |
| [3] | WP | 0 | 0 | 0 | 0 |
| 2 | WaitEn | 0 | 0 | 0 | 0 |
| 1 | WaitPol | 0 | 0 | 0 | 0 |
| 0 | RBLE | 0 | 0 | 0 | 0 |

### 3.3.7 Bank Status Registers 0-7

The eight read/write SMBSRx Registers show the status of each memory banks 0-7. Figure 3-7 shows the bit assignments of an SMBSRx Register.

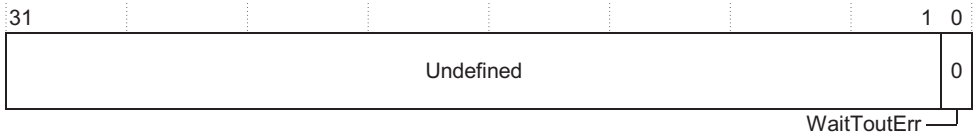



**Figure 3-7 SMBSRx Register bit assignments**

Table 3-10 lists the bit assignments of a SMBSRx Register.

**Table 3-10 SMBSRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:1] | - | Read undefined. Write as zero. |
| [0] | WaitToutErr | External wait timeout error flag.<br>Reading this bit:<br>0 No Error (default at reset).<br>1 External wait timeout error.<br>Writing this bit:<br>0 Has no effect.<br>1 Clears the write protect error status flag. |

### 3.3.8 Bank Burst Read Wait Delay Control Registers 0-7

The eight read/write SMBWSTBRDRx Registers are the bank 0-7 burst read wait state control registers. Each register is identical in structure. Figure 3-8 shows the bit assignments of an SMBWSTBRDRx Register.

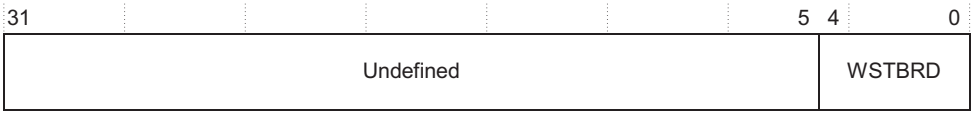



**Figure 3-8 SMBWSTBRDRx Register bit assignments**

Table 3-11 lists the bit assignments of an SMBWSTBRDRx Register.

**Table 3-11 SMBWSTBRDRx Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:5] | - | Read undefined. Write as zero. |
| [4:0] | WSTBRD | Burst read wait state. Defaults to 1111 at reset.<br>For burst devices, the WSTBRD field controls the number of wait states for the burst read accesses after the first read.<br>Wait state time = WSTBRD x $t_{HCLK}$[a]<br>WSTBRD does not apply to nonburst devices. |

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

### 3.3.9 SSMC Status Register

The read-only SSMCSR Register shows the current status of the external wait during an externally waited transfer. Figure 3-9 shows the bit assignments of the SSMCSR Register.

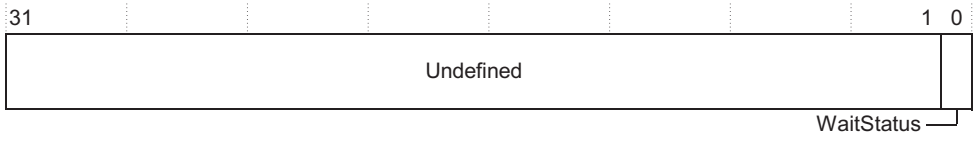



**Figure 3-9 SSMCSR Register bit assignments**

Table 3-12 lists the bit assignments of the SSMCSR Register.

**Table 3-12 SSMCSR Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:1] | - | Read undefined. |
| [0] | WaitStatus | External wait status, read:<br>0 **SMWAIT** deasserted.<br>1 **SMWAIT** asserted.<br>After an externally waited transfer that was terminated early, this bit value can detect when **SMWAIT** is deasserted. At all other times, this bit reads zero. |

### 3.3.10 SSMC Control Register

The read/write SSMCCR Register controls the SSMC. Settings in this register affect all banks. Figure 3-10 shows the bit assignments of the SSMCCR Register.

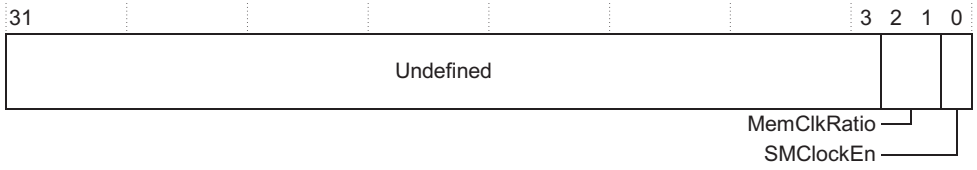



**Figure 3-10 SSMCCR Register bit assignments**

Table 3-13 lists the bit assignment of the SSMCCR Register.

**Table 3-13 SSMCCR Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:3] | - | Read undefined. Write as zero. |
| [2:1] | MemClkRatio | Defines the ratio of **SMMEMClk** to **HCLK**:<br>`00` **SMMEMCLK** = **HCLK** (default).<br>`01` **SMMEMCLK** = **HCLK**/2.<br>`10` **SMMEMCLK** = **HCLK**/3.<br>`11` Reserved. |
| [0] | SMClockEn | **SMCLK** enable:<br>`0` Clock only active during memory accesses.<br>`1` Clock always running.<br>Clock stopping saves power by stopping **SMCLK** when it is not required. If clock stopping is enabled before the memory access, SSMC controller stops **SMCLK** on the following conditions:<br>• asynchronous read access to asynchronous memory<br>• asynchronous write access to asynchronous memory<br>• asynchronous read access to synchronous memory<br>• asynchronous write access to synchronous memory. |

**Note**

You must set the SSMCCR register to match the externally defined clock ratio.

### 3.3.11 Peripheral Identification Registers 0-3

The read-only SSMCPeriphID0-3 Registers are four 8-bit registers, that span address locations `0xFE0-0xFEC`. You can treat the registers conceptually as a single 32-bit register. The read-only registers provide the following options of the peripheral:

**PartNumber[11:0]**

This identifies the peripheral. The product code `0x93` is used for the PrimeCell SSMC.

**DesignerID[19:12]**

This is the identification of the designer. ARM Limited is `0x41` (ASCII A).

**Revision[23:20]**

This is the revision number of the peripheral. The revision number starts from 0 and is revision dependent.

**Configuration[31:24]**

This is the configuration option of the peripheral. The configuration value is 0.

Figure 3-11 shows the bit assignment for the SSMCPeriphID0-3 Registers.

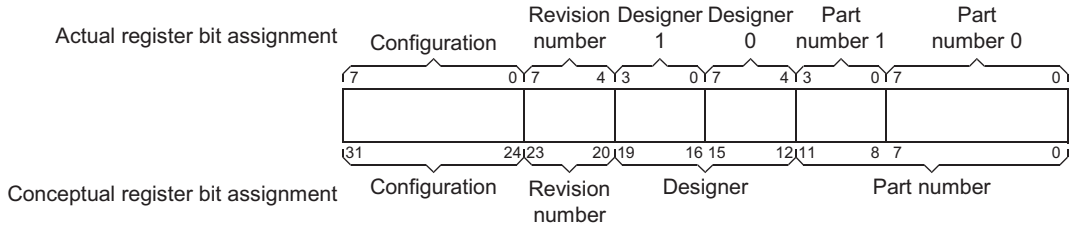



**Figure 3-11 Peripheral identification Register bit assignments**

The four, 8-bit Peripheral Identification Registers are described in the following sections:

- *Peripheral Identification Register 0*
- *Peripheral Identification Register 1* on page 3-20
- *Peripheral Identification Register 2* on page 3-20
- *Peripheral Identification Register 3* on page 3-20.

### Peripheral Identification Register 0

The read-only SSMCPeriphID0 Register is hard-coded and the fields within the registers determine the reset value. Table 3-14 lists the bit assignment of the SSMCPeriphID0 Register.

**Table 3-14 SSMCPeriphID0 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:0] | PartNumber0 | These bits read back as `0x93` |

### Peripheral Identification Register 1

The read-only SSMCPeriphID1 Register is hard-coded and the fields within the registers determine the reset value. Table 3-15 lists the bit assignment of the SSMCPeriphID1 Register.

**Table 3-15 SSMCPeriphID1 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:4] | Designer0 | These bits read back as `0x1` |
| [3:0] | PartNumber1 | These bits read back as `0x0` |

### Peripheral Identification Register 2

The read-only SSMCPeriphID2 Register is hard-coded and the fields within the registers determine the reset value. Table 3-16 lists the bit assignment of the SSMCPeriphID2 Register.

**Table 3-16 SSMCPeriphID2 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:4] | Revision | These bits read back as `0x1` |
| [3:0] | Designer1 | These bits read back as `0x4` |

### Peripheral Identification Register 3

The read-only SSMCPeriphID3 Register is hard-coded and the fields within the registers determine the reset value. Table 3-17 lists the bit assignment of the SSMCPeriphID3 Register.

**Table 3-17 SSMCPeriphID3 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:0] | Configuration | These bits read back as `0x00` |

### 3.3.12 PrimeCell Identification Registers 0-3

The read-only SSMCPCellID0-3 Registers are four 8-bit registers, that span address locations `0xFF0-0xFFC`. You can treat the registers conceptually as a 32-bit register. The register acts as a standard cross-peripheral identification system. Figure 3-12 shows the bit assignment for the SSMCPCellID0-3 Registers.

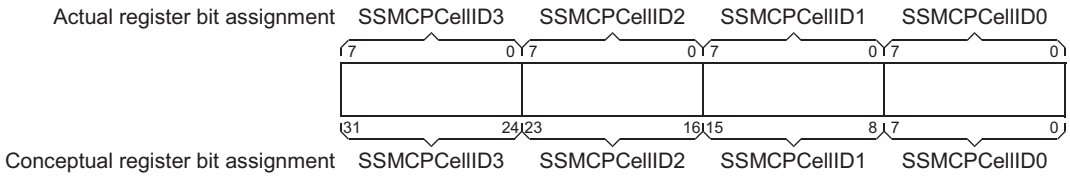



**Figure 3-12 PrimeCell identification Register bit assignments**

The four 8-bit registers are described in the following subsections:

- *PrimeCell Identification Register 0*
- *PrimeCell Identification Register 1* on page 3-22
- *PrimeCell Identification Register 2* on page 3-22
- *PrimeCell Identification Register 3* on page 3-22.

#### PrimeCell Identification Register 0

The read-only SSMCPCellID0 Register is hard-coded and the fields within the registers determine the reset value. Table 3-18 lists the bit assignment of the SMCPCellID0 Register.

**Table 3-18 SMCPCellID0 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:0] | SSMCPCellID0 | These bits read back as `0x0D` |

### PrimeCell Identification Register 1

The read-only SSMCPCellID1 Register is hard-coded and the fields within the registers determine the reset value. Table 3-19 lists the bit assignment of the SSMCPCellID1 Register.

**Table 3-19 SSMCPCellID1 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:0] | SSMCPCellID1 | These bits read back as `0xF0` |

### PrimeCell Identification Register 2

The read-only SSMCPCellID2 Register is hard-coded and the fields within the registers determine the reset value. Table 3-20 lists the bit assignment of the SMCPCellID2 Register.

**Table 3-20 SMCPCellID2 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:0] | SSMCPCellID2 | These bits read back as `0x05` |

### PrimeCell Identification Register 3

The read-only SSMCPCellID3 Register is hard-coded and the fields within the registers determine the reset value. Table 3-21 lists the bit assignment of the SSMCPCellID3 Register.

**Table 3-21 SSMCPCellID3 Register bit assignments**

| Bits | Name | Function |
|---|---|---|
| [15:8] | - | Read undefined |
| [7:0] | SSMCPCellID3 | These bits read back as `0xB1` |

# Chapter 4
# **Programmer's Model for Test**

This chapter describes the additional logic for functional verification and production testing. It contains the following sections:

## 4.1 Scan testing

The PrimeCell SSMC 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.

## 4.2 Test registers

Test registers enable testing of the SSMC using the AMBA TIC block-level tests methodology and for integration testing. See Table 3-1 on page 3-3 for the base offsets and reset values of these registers. The registers are described in:

- *SSMC Test Control Register*
- *SSMC Test Output Register*
- *SSMC Test Input Register* on page 4-4.

### 4.2.1 SSMC Test Control Register

This read/write register controls the test mode of the SSMC. Figure 4-1 shows the bit assignments of the SSMCITCR Register.

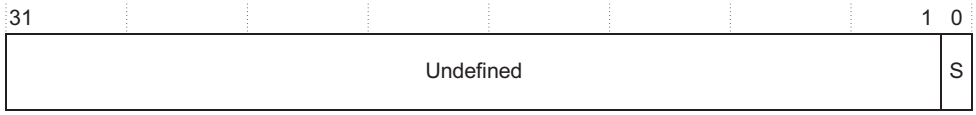



**Figure 4-1 SSMCITCR Register bit assignments**

Table 4-1 lists the bit assignments of the SSMCITCR Register.

**Table 4-1 SSMCITCR Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:1] | - | Read undefined. Write as zero. |
| [0] | S | Test mode enable. Multiplex the test registers to control the input and output lines:<br>0 = normal operation<br>1 = test registers multiplexed onto input and output lines. |

### 4.2.2 SSMC Test Output Register

This read/write register controls and reads the outputs of the SSMC. Figure 4-2 on page 4-4 shows the bit assignments of the SSMCITOP Register.

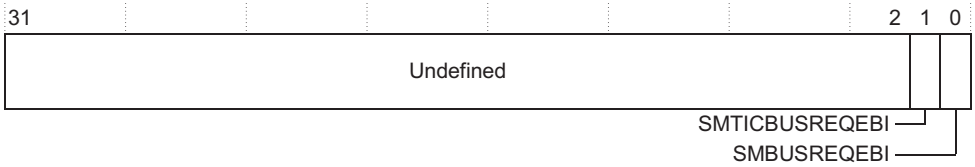



**Figure 4-2 SSMCITOP Register bit assignments**

Table 4-2 lists the bit assignments of the SSMCITOP Register.

**Table 4-2 SSMCITOP Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:2] | - | Read undefined. Write as zero. |
| [1] | SMTICBUSREQEBI | Read returns current value of signal. Write sets signal to value written. |
| [0] | SMBUSREQEBI | Read returns current value of signal. Write sets signal to value written. |

### 4.2.3 SSMC Test Input Register

This read/write register controls and reads the inputs of the SSMC. Figure 4-3 shows the bit assignments of the SSMCITIP Register.

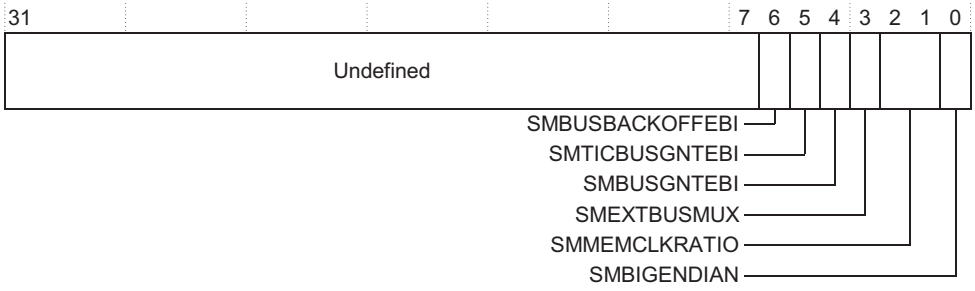



**Figure 4-3 SSMCITIP Register bit assignments**

Table 4-3 lists the bit assignments of the SSMCITIP Register.

**Table 4-3 SSMCITIP Register bit assignments**

| Bits | Name | Description |
|---|---|---|
| [31:7] | - | Read undefined. Write as zero. |
| [6] | SMBUSBACKOFFEBI | Read returns current value of signal.<br>Write sets signal to value written. |
| [5] | SMTICBUSGNTEBI | Read returns current value of signal.<br>Write sets signal to value written. |
| [4] | SMBUSGNTEBI | Read returns current value of signal.<br>Write sets signal to value written. |
| [3] | SMEXTBUSMUX | Read returns current value of signal.<br>Write sets signal to value written. |
| [2:1] | SMMEMCLKRATIO | Read returns current value of signal.<br>Write sets signal to value written. |
| [0] | SMBIGENDIAN | Read returns current value of signal.<br>Write sets signal to value written. |

# Appendix A
# **Signal Descriptions**

This chapter describes the AMBA AHB and module-specific non-AMBA AHB signals that you use with the SSMC. It contains the following sections:

## A.1 AMBA AHB interface signals

Table A-1 lists the common AMBA AHB signals.

**Table A-1 Common AMBA AHB signals**

| Signal name | Type | Source/ destination | Description |
|---|---|---|---|
| **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. Resets the system and the bus when asserted LOW. |

## A.2 AMBA AHB slave interface signals

There are two AHB slave interfaces on this device. One interface accesses the Control Registers and the other accesses the memory. Table A-2 lists the AMBA AHB slave interface signals.

**Table A-2 AMBA AHB slave interface signals**

| Signal name | Type | Source/destination | Description |
|---|---|---|---|
| **HADDRREG[11:2]** | Input | AMBA AHB master | System address bus, least significant 29 bits, driven by the active bus master. |
| **HRDATAREG[31:0]** | Output | AMBA AHB slave | Read data bus. Reads data from the PrimeCell AHB SSMC. |
| **HREADYINREG** | Input | Other AHB slave | Transfer completed input. When HIGH the **HREADYIN** signal indicates that a transfer has finished on the bus. You can drive this signal LOW to extend a transfer. |
| **HREADYOUTREG** | Output | AMBA AHB slave | Transfer done output. When HIGH indicates that a transfer has finished on the bus. You can drive this signal LOW to extend a transfer. |
| **HRESPREG[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. |
| **HSELREG** | Input | AMBA AHB decoder | Slave select signal for PrimeCell AHB SSMC Control and Status Registers. |
| **HSIZEREG[2:0]** | Input | AMBA AHB master | Transfer size signal. This signal indicates the size of the current transfer, this can be byte (8-bit), halfword (16-bit), or word (32-bit). |
| **HTRANSREG[1:0]** | Input | AMBA AHB master | Indicates the type of the current transfer, this can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY. |
| **HWDATAREG[31:0]** | Input | AMBA AHB master | Write data bus. Transfers data to the PrimeCell AHB SSMC. |
| **HWRITEREG** | Input | AMBA AHB master | Transfer direction signal. When HIGH, this signal indicates a write to the PrimeCell AHB SSMC and when LOW, a read from the PrimeCell AHB SSMC. |

Table A-3 on page A-4 describes the AMBA AHB slave memory interface signals.

**Table A-3 AMBA AHB slave memory interface signals**

| Signal name | Type | Source/destination | Description |
|---|---|---|---|
| **HADDRSMC[11:0]** | Input | AMBA AHB master | System address bus, least significant 29 bits, driven by the active bus master. |
| **HBURSTSMC[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. |
| **HRDATASMC[31:0]** | Output | AMBA AHB slave | Read data bus. Reads data from the PrimeCell AHB SSMC. |
| **HREADYINSMC** | Input | Other AHB slave | Transfer completed input. When HIGH the **HREADYIN** signal indicates that a transfer has finished on the bus. You can drive this signal LOW to extend a transfer. |
| **HREADYOUTSMC** | Output | AMBA AHB slave | Transfer done output. When HIGH indicates that a transfer has finished on the bus. You can drive this signal LOW to extend a transfer. |
| **HRESPSMC[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. |
| **HSELSMC[7:0]** | Input | AMBA AHB decoder | Slave select signal for PrimeCell AHB memory interface. One select line is provided for each memory bank. |
| **HSIZESMC[2:0]** | Input | AMBA AHB master | Transfer size signal. This signal indicates the size of the current transfer, this can be byte (8-bit), halfword (16-bit), or word (32-bit). |
| **HTRANSSMC[1:0]** | Input | AMBA AHB master | Indicates the type of the current transfer, this can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY. |
| **HWDATASMC[31:0]** | Input | AMBA AHB master | Write data bus. Transfers data to the PrimeCell AHB SSMC. |
| **HWRITESMC** | Input | AMBA AHB master | Transfer direction signal. When HIGH, this signal indicates a write to the PrimeCell AHB SSMC and when LOW, a read from the PrimeCell AHB SSMC. |

## A.3 AMBA AHB master interface signals

Table A-4 lists the AMBA AHB master interface signals.

**Table A-4 AMBA AHB master interface signals**

| Signal name | Type | Source/destination | Description |
|---|---|---|---|
| **HADDRTIC[31:0]** | Output | AMBA AHB slave | The 32-bit system address bus. |
| **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. |
| **HBUSREQTIC** | Output | AMBA AHB arbiter | A signal from the TIC to the bus arbiter that indicates that it requires the bus. |
| **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. |
| **HLOCKTIC** | Output | AMBA AHB arbiter | When HIGH, this signal indicates that the TIC requires locked access to the bus and you must not grant any other master the bus until this signal is LOW. |
| **HPROTTIC[3:0]** | Output | AMBA AHB slave | The protection control signals indicate if the transfer is an opcode fetch or data access, in addition to 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. |
| **HRDATATIC[31:0]** | Input | AMBA AHB slave | The read data bus transfers data from bus slaves to the bus master during test read operations. |
| **HREADYINTIC** | Input | AMBA AHB slave | Transfer completed input. When HIGH the **HREADYIN** signal indicates that a transfer has finished on the bus. You can driver this signal LOW to extend a transfer. |
| **HRESPTIC[1:0]** | Input | AMBA AHB slave | Transfer response. The transfer response provides additional information on the status of a transfer. The TIC supports both SPLIT and RETRY responses. |
| **HSIZETIC[2:0]** | Output | AMBA AHB slave | Transfer size signal. This signal indicates the size of the current transfer, this can be byte (8-bit), halfword (16-bit), or word (32-bit). The TIC does not support larger transfer sizes. |

**Table A-4 AMBA AHB master interface signals (continued)**

| Signal name | Type | Source/destination | Description |
|---|---|---|---|
| **HTRANSTIC[1:0]** | Output | AMBA AHB slave | Indicates the type of the current transfer, this can be NONSEQUENTIAL, SEQUENTIAL, or IDLE. The TIC does not use the BUSY transfer type. |
| **HWDATATIC[31:0]** | Output | AMBA AHB slave | The write data bus transfers data from the master to bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However you can easily extend this to permit for higher bandwidth operation. |
| **HWRITETIC** | Output | AMBA AHB slave | Transfer direction signal. When HIGH, this signal indicates a write transfer and when LOW a read transfer. |

## A.4 Non-AMBA signals

Table A-5 lists the internal signals.

**Table A-5 Internal signal descriptions**

| Signal name | Type | Source/ destination | Description |
|---|---|---|---|
| **SCANENABLE** | Input | System | Dummy pin for use as a dedicated scan enable input. |
| **SCANINCLKDELAY** | Input | System | Scan chain input with respect to **SMMEMCLKDELAY**. |
| **SCANINFBCLK[3:0]** | Input | System | Scan chain input with respect to **FBCLK**. |
| **SCANINHCLK** | Input | System | Dummy pin for use as a dedicated **HCLK** scan chain input. |
| **SCANINSMMEMCLK** | Input | System | Scan chain input for **SMMEMCLK** domain. |
| **SCANINnSMMEMCLK** | Input | System | Scan chain input for **nSMMEMCLK** domain. |
| **SCANOUTCLKDELAY** | Output | System | Scan chain output with respect to **SMMEMCLKDELAY**. |
| **SCANOUTFBCLK[3:0]** | Output | System | Scan chain output with respect to **FBCLK**. |
| **SCANOUTHCLK** | Output | System | Dummy pin for use as a dedicated **HCLK** scan chain input. |
| **SCANOUTSMMEMCLK** | Output | System | Scan chain output for **SMMEMCLK** domain. |
| **SCANOUTnSMMEMCLK** | Output | System | Scan chain output for **nSMMEMCLK** domain. |
| **SMBIGENDIAN** | Input | System | This static configuration bit indicates the type of endianness of the memory system:<br>`0` Little-endian.<br>`1` Big-endian. |
| **SMBUSBACKOFFEBI** | Input | External bus multiplexor | Release bus signal. Tells SSMC to release bus. Only used when **EXTBUSMUX** is tied to 1. |
| **SMBUSGNTEBI** | Input | External bus multiplexor | Bus grant signal. Indicates that the SSMC is granted control of the external bus. Only used when **EXTBUSMUX** is tied to 1. |
| **SMBUSREQEBI** | Output | External bus multiplexor | Bus request signal. Indicates that the SSMC has requested use of the external bus. Only used when **EXTBUSMUX** is tied to 1. |

**Table A-5 Internal signal descriptions (continued)**

| Signal name | Type | Source/ destination | Description |
|---|---|---|---|
| **SMEXTBUSMUX** | Input | System | This static configuration bit indicates whether the internal bus multiplexor (DBI) is used, or an external bus multiplexor (for example EBI) is used:<br>`0` Internal bus multiplexor.<br>`1` External bus multiplexor. |
| **SMMEMCLK** | Input | System | Memory clock. |
| **nSMMEMCLK** | Input | System | Inverted memory clock. |
| **SMMEMCLKDELAY** | Input | Input pad | Delayed version of **SMMEMCLK** that clocks external control signals. |
| **SMMEMCLKRATIO[1:0]** | Input | System | Defines ratio of **SMMEMCLK** to **HCLK**:<br>`00` **SMMEMCLK = HCLK**.<br>`01` **SMMEMCLK = HCLK**/2.<br>`10` **SMMEMCLK = HCLK**/3.<br>`11` Reserved. |
| **SMTICBUSGNTEBI** | Input | External bus multiplexor | Bus grant signal, indicates that the TIC is granted control of the external bus. Only used when **EXTBUSMUX** tied to 1. |
| **SMTICBUSREQEBI** | Output | External bus multiplexor | Bus request signal, indicates that the TIC has requested use of the external bus. Only used when **EXTBUSMUX** tied to 1. |

## A.5 Input/output pad signals

Table A-6 lists the signals to the input/output pads.

**Table A-6 Input/output pad signals**

| 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], and [7:0] on the data bus. |
| **nSMBURSTWAIT[7:0]** | Input | Input pad | Synchronous burst wait input that the external device uses to delay a synchronous burst transfer. |
| **nSMCS0** | Output | Output pad | Chip select for bank 0 of external memory. Default active LOW. |
| **nSMCS1** | Output | Output pad | Chip select for bank 1 of external memory. Default active LOW. |
| **nSMCS2** | Output | Output pad | Chip select for bank 2 of external memory. Default active LOW. |
| **nSMCS3** | Output | Output pad | Chip select for bank 3 of external memory. Default active LOW. |
| **nSMCS4** | Output | Output pad | Chip select for bank 4 of external memory. Default active LOW. |
| **nSMCS5** | Output | Output pad | Chip select for bank 5 of external memory. Default active LOW. |
| **nSMCS6** | Output | Output pad | Chip select for bank 6 of external memory. Default active LOW. |
| **nSMCS7** | Output | Output pad | Chip select for bank 7 of external memory. 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. |
| **nSMOEN** | Output | Output pad | Output enable for external memory banks, active LOW. |
| **nSMWEN** | Output | Output pad | Write enable for the external memory banks, active LOW. |

**Table A-6 Input/output pad signals (continued)**

| Signal name | Type | Source/destination | Description |
|---|---|---|---|
| **SMADDR[25:0]** | Output | Output pad | External memory address bus, to external memory banks. |
| **SMADDRVALID** | Output | Output pad | External address valid output, that indicates when the address output is stable during synchronous burst transfers. |
| **SMBAA** | Output | Output pad | External burst address advance signal. Advances the address count in the external memory device. |
| **SMBLS7POL** | Input | Input pad | This defines the reset value of bit 6 in SMBCR7 (**nSMBLS**):<br>`0` **nSMBLS** is active LOW (default).<br>`1` **nSMBLS** is active HIGH. |
| **SMCANCELWAIT** | Input | Input pad | This signal enables the system to recover from an externally waited transfer that takes longer than expected to finish. Active HIGH. |
| **SMCLK[3:0]** | Output | Output pad | The clocks output to synchronous memory devices. |
| **SMCS0** | Output | Output pad | Chip select for bank 0 of external memory.<br>Default active HIGH. |
| **SMCS1** | Output | Output pad | Chip select for bank 1 of external memory.<br>Default active HIGH. |
| **SMCS2** | Output | Output pad | Chip select for bank 2 of external memory.<br>Default active HIGH. |
| **SMCS3** | Output | Output pad | Chip select for bank 3 of external memory.<br>Default active HIGH. |
| **SMCS4** | Output | Output pad | Chip select for bank 4 of external memory.<br>Default active HIGH. |
| **SMCS5** | Output | Output pad | Chip select for bank 5 of external memory.<br>Default active HIGH. |
| **SMCS6** | Output | Output pad | Chip select for bank 6 of external memory.<br>Default active HIGH. |
| **SMCS7** | Output | Output pad | Chip select for bank 7 of external memory.<br>Default active HIGH. |

**Table A-6 Input/output pad signals (continued)**

| Signal name | Type | Source/destination | Description |
|---|---|---|---|
| **SMDATAIN[31:0]** | Input | Input pad | External input data bus that reads data from memory bank. |
| **SMDATAOUT[31:0]** | Input | Input pad | External output data that writes data from SSMC to memory bank. |
| **SMFBCLK[3:0]** | Input | Input pad | The fedback clocks from the memory devices. |
| **SMMWCS7[1:0]** | Input | Input pad | These static configuration bits indicate the memory width used for boot memory bank 1:<br>00 8-bit.<br>01 16-bit.<br>10 32-bit.<br>11 Reserved. |
| **SMTESTACK** | 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 **SMTESTACK** is LOW, you must extend the current test vector until **SMTESTACK** becomes HIGH. |
| **SMTESTREQA** | Input | Input pad | This is the Test Bus Request. A input signal and is required as a dedicated device pin.<br>During normal system operation, the **SMTESTREQA** signal requests entry into the test mode. During test, **SMTESTREQA**, in combination with **SMTESTREQB**, indicates the type of test vector that is applied in the following cycle. |
| **SMTESTREQB** | Input | Input pad | During test, this signal, in combination with **SMTESTREQA**, indicates the type of test vector to apply in the following cycle. |
| **SMWAIT** | Input | Input pad | Wait mode input from external memory controller. Active HIGH or active LOW (default), as programmed in the SSMC Control Registers for each bank. |

# Glossary

This glossary describes some of the terms used in ARM manuals. 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)**

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, APB, and AXI conform 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.

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

**Aligned** A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. 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.

**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.

**Architecture** The organization of hardware and/or software that characterizes a processor and its attached components, and enables devices with similar characteristics to be grouped together when describing their behavior, for example, Harvard architecture, instruction set architecture, ARMv6 architecture.

**ASIC** *See* Application Specific Integrated Circuit.

**ATPG** *See* Automatic Test Pattern Generation.

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

**AXI** *See* Advanced eXstensible Interface.

**Beat** Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats.

*See also* Burst.

**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, four-beat, eight-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.

**Data Abort** An indication from a memory system to the core of an attempt to access an illegal data memory location. An exception must be taken if the processor attempts to use the data that caused the abort.

*See also* Abort, External Abort, and Prefetch Abort.

**Direct Memory Access (DMA)**
An operation that accesses main memory directly, without the processor performing any accesses to the data concerned.

**DMA** *See* Direct Memory Access.

**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.

**Multi-layered** An AMBA scheme to break a bus into segments that are controlled in access. This enables local masters to reduce lock overhead.

**Power-on reset** *See* Cold reset.

**Region** A partition of instruction or data memory space.

**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.

**Undefined** Indicates an instruction that generates an Undefined instruction trap. See the *ARM Architecture Reference Manual* for more details on ARM exceptions.

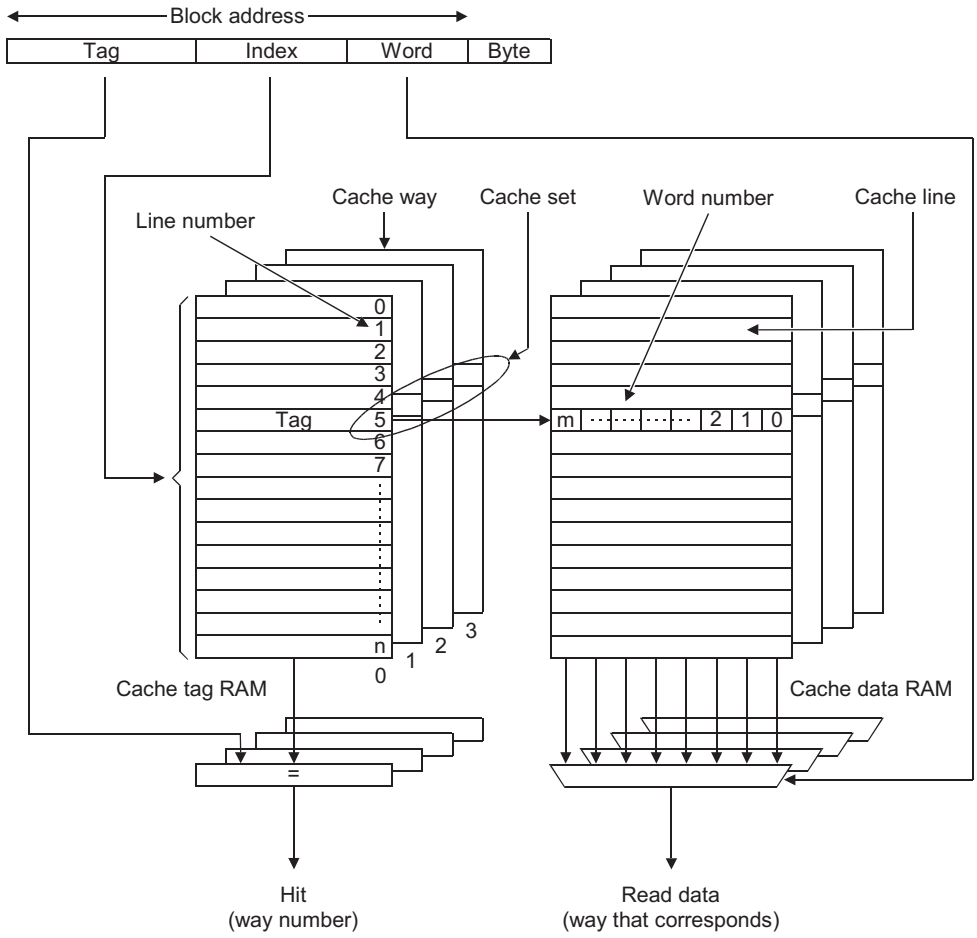
**Unpredictable** For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor,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

**Word** A 32-bit data item.

**Cache terminology diagram**

The diagram below illustrates the following cache terminology:

- block address
- cache line
- cache set
- cache way
- index
- tag.



# Index

## D



## E



## F



## H



## I



## L



## M



## N



## O



## P

## R



## S

## T



## W

