LatticeMico Asynchronous SRAM Controller

The LatticeMico asynchronous SRAM controller is a slave device for the WISHBONE architecture. It interfaces to an industry-standard asynchronous SRAM device.

Version

This document describes the 3.3 version of the LatticeMico asynchronous SRAM controller.

Features

The LatticeMico asynchronous SRAM controller includes the following features:

- WISHBONE B.3 interface:
  - Data bus width configurable as 8 or 32 bits
  - Support for Classic and Linear Incrementing Burst WISHBONE cycles
  - Support for byte, halfword, and word transfers on 32-bit WISHBONE data bus
- Configurable SRAM data bus width up to 32 bits
- Configurable SRAM address bus width up to 32 bits
- Configurable read latency
- Configurable write latency
- Optional capability for registering data from asynchronous SRAM before sending to WISHBONE master on reads

Copyright © December 2012 Lattice Semiconductor Corporation.

For additional details about the WISHBONE bus, refer to the *LatticeMico32 Processor Reference Manual* or *LatticeMico8 Processor Reference Manual*.

Functional Description

The asynchronous SRAM controller translates the synchronous WISHBONE bus signals into control strobes used to access an asynchronous SRAM. The controller decodes the WISHBONE cycle type and generates asynchronous chip selects, byte enables, read enables, and write enables, as required. For further information on the WISHBONE registered feedback bus cycle, refer to *WISHBONE Specifications, Version B3,* Chapter 4.

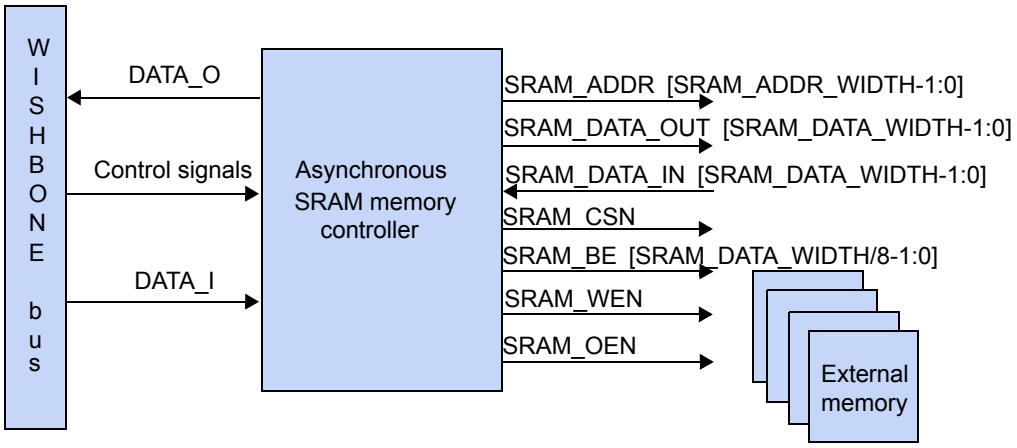
The memory controller can be configured for any combination of address and data bus widths of the asynchronous SRAM. The address bus can be up to 32 bits. The data bus can be configured to be 8, 16, or 32 bits.

When in operation, the controller monitors the address bus and STB_I and CYC_I to determine when an asynchronous memory transaction is in progress. The address, STB_I, and CYC_I control signals are asserted or deasserted at the CLK_I rising edge. CLK_I may be transitioning at a rate much too high for the asynchronous RAM to accept, so the memory controller must control and hold off the ACK_O control signal that indicates that the WISHBONE bus transaction is complete. Since asynchronous RAM devices do not have a cycle acknowledge signal, the memory controller provides one. The ACK_O signal is controlled with a fixed read/write latency value. The read latency is independent of the write latency. Each increment in latency value represents an increase in the length of the bus transaction by one CLK_I time period. The read/write latency permits the controller to work with slower SRAM devices. The controller counts CLK_I cycles until the read/write latency value has been reached. At this point, ACK_O is asserted, and the WISHBONE cycle is terminated.

The memory controller does not implement dynamic or region-based latency. The read and write latency values are fixed during module generation. If different regions of the asynchronous memory space require different read/write latency values, you must generate one asynchronous RAM controller for each region. If the latency values cannot be set high enough to permit the SRAM to operate, it is necessary to obtain faster SRAM, reduce the CLK_I time period, or modify the HDL source for the controller.

Figure 1 shows how an application uses the memory controller.

Figure 1: Asynchronous SRAM Usage



Configuration

The following sections describe the graphical user interface (UI) parameters, the hardware description language (HDL) parameters, and the I/O ports that you can use to configure and operate the LatticeMico asynchronous SRAM controller.

UI Parameters

Table 1 shows the UI parameters available for configuring the LatticeMico asynchronous SRAM controller through the Mico System Builder (MSB) interface.

Table 1: Asynchronous SRAM Controller UI Parameters

Dialog Box Option	Description	Allowable Values	Default Values
Instance Name	Specifies the name of the asynchronous SRAM controller instance.	Alphanumeric and underscores	sram
Base Address	Specifies the base address for the device. The minimum boundary alignment is 0x4.	0X00000000–0XFFFFFFFF	0X00000000
Size	Specifies the size of the external memory, in bytes.	0 – 4294967296	1048576
Share External Ports (for HPE_mini Board)	Enables a common address and data bus for flash and SRAM components created for the platform. When this option is selected, each flash and SRAM component adds its instance name to the address or data bus port name.	1, 0	1
Settings			

Table 1: Asynchronous SRAM Controller UI Parameters (Continued)

Dialog Box Option	Description	Allowable Values	Default Values
Read Latency	Specifies the latency for reading the SRAM (measured in CLK_I cycles).	1 – 15	1
Write Latency	Specifies the latency for writing the SRAM (measured in CLK_I cycles).	1 – 15	1
SRAM Address Width[2]	Specifies the width of the address, in bits.	1 – 32	23*
SRAM Data Width[1]	Specifies the width of the memory's data bus, in bits.	8, 16, 32	32
SRAM Byte Enable Width	Specifies the width of the byte enable, or control strobe, for each of the 8-bit pieces of logic that constitute the data bus in the asynchronous RAM. The byte enable indicates that the LatticeMico32 microprocessor is accessing the 8-bit sub-element of the larger combined data bus. The SRAM BE width must be assigned a value that is the data bus width modulo 8. For example, if the default value of the data bus width is 32, the SRAM BE width should be 4. Legal values for this field are 4, 2, and 1. The byte enables (BE[n], n= 3..0) are activated as follows: 32-bit bus: D31-24/BE3 D23-16/BE2 D15-8/BE1 D7-0/BE0 16-bit bus: D15-8/BE1 D7-0/BE0 8-bit bus: D7-0/BE0 Therefore, you use SRAM_BE_WIDTH to define the upper limit of the Verilog bus: output [SRAM_BE_WIDTH-1:0] sram_be;	4, 2, 1	4
Performance Settings			
Registered Data Output	Specifies whether the data from asynchronous SRAM must be latched before sending on WISHBONE data bus.	0, 1	0
WISHBONE Configuration			
WISHBONE Data Bus Width	Specifies the size of the WISHBONE data bus.	8, 32	32

[1,2] On the LatticeMico32/DSP development board, the address and data bus of the SRAM is shared with that of the parallel flash.

* The default value is 23 because the two lower-order bits are not used, since the SRAM on the board is configured as 32 bits wide.

HDL Parameters

Table 2 lists the parameters that appear in the HDL.

Table 2: Asynchronous SRAM Controller HDL Parameters

Parameter Name	Description	Allowable Values
SRAM_DATA_WIDTH	Defines the width of the memory's data bus.	8, 16, 32
SRAM_ADDR_WIDTH	Defines the width of the address.	1-32
READ_LATENCY	Defines the latency for reading the SRAM.	1-15
WRITE_LATENCY	Defines the latency for writing the SRAM.	1-15
DATA_OUTPUT_REG	Determines whether data from asynchronous SRAM is registered before sending on WISHBONE data bus.	0, 1
ASRAM_WB_DAT_WIDTH	Specifies the width of the WISHBONE data bus.	8, 32

I/O Ports

Table 3 describes the input and output ports of the LatticeMico asynchronous SRAM controller.

Table 3: Asynchronous SRAM Controller I/O Ports

I/O Port	Active	Direction	Initial State	Description
WISHBONE Side Ports				
CLK_I	—	I	0	System clock
RST_I	High	I	0	System reset
CTI_I	—	I	0	Cycle-type identifier. Only "000" is valid.
BTE_I	—	I	0	Burst-type extension.
ADR_I [31:0]	—	I	0	WISHBONE address bus
DAT_I [ASRAM_WB_DAT_WIDTH-1:0]	—	I	0	WISHBONE data bus input (for write)
SEL_I [ASRAM_WB_DAT_WIDTH/8-1:0]	High	I	0	Select output array, one bit for every byte
WE_I	High	I	0	Write enable
STB_I	High	I	0	Strobe indicating a valid data transfer
CYC_I	High	I	X	A valid bus cycle in progress

Table 3: Asynchronous SRAM Controller I/O Ports

I/O Port	Active	Direction	Initial State	Description
LOCK_I	High	I	X	WISHBONE lock signal
ACK_O	High	O	0	Indicates the normal termination of a bus
DAT_O [ASRAM_WB_DAT_WIDTH-1:0]	—	O	0	WISHBONE data bus output (for read)
ERR_0	High	O	0	WISHBONE error signal
RTY_O	High	O	0	WISHBONE retry signal
Asynchronous SRAM Interface Ports				
SRAM_ADDR [SRAM_ADDR_WIDTH-1:0]	—	O	0	SRAM address output
SRAM_DATA_OUT [SRAM_DATA_WIDTH-1:0]	—	O	0	SRAM data output
SRAM_DATA_IN [SRAM_DATA_WIDTH-1:0]	—	I	X	SRAM data input
SRAM_CSN	Low	O	1	SRAM chip select
SRAM_BE [SRAM_BE_WIDTH-1:0]	Low	O	0	SRAM byte enable **Note**: SRAM_BE_WIDTH is equivalent to SRAM_DATA_WIDTH/8.
SRAM_WEN	Low	O	0	SRAM write enable
SRAM_OEN	Low	O	0	SRAM output enable

Timing Diagrams

Figure 2 shows the asynchronous SRAM controller's timing for a WISHBONE 32-bit Classic Read and a asynchronous SRAM with a read latency of 1 and a data bus width of 32 bits. The read transaction begins with SRAM_CYC_I and SRAM_STB_I being asserted following a clock rising edge. The memory controller passes the address asserted on the SRAM_ADR_I bus on the next

clock rising edge. Since the latency is 1, data is expected from the asynchronous SRAM in the same cycle. The memory controller drives the data on to the SRAM_DAT_O bus.

Figure 2: WISHBONE 32-Bit Classic Read (non-registered read)

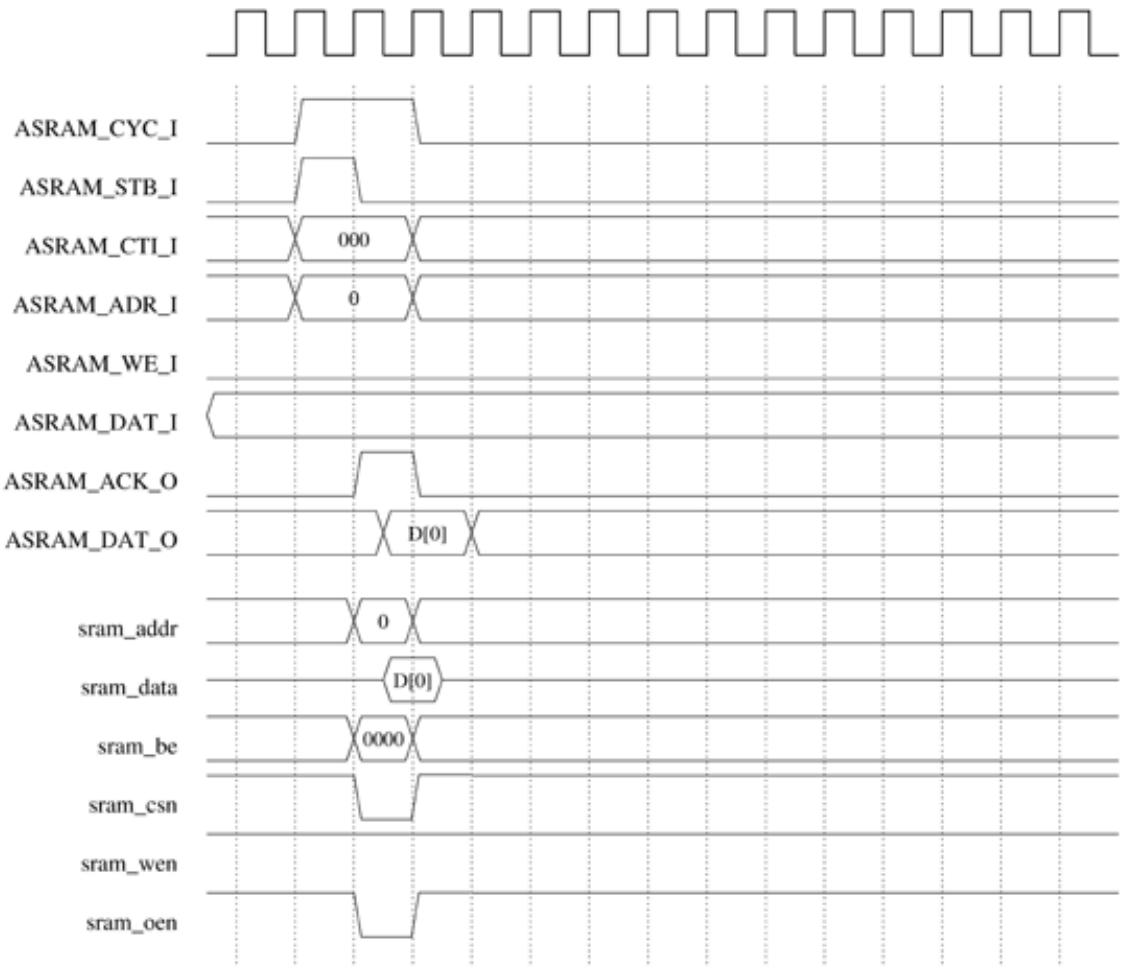
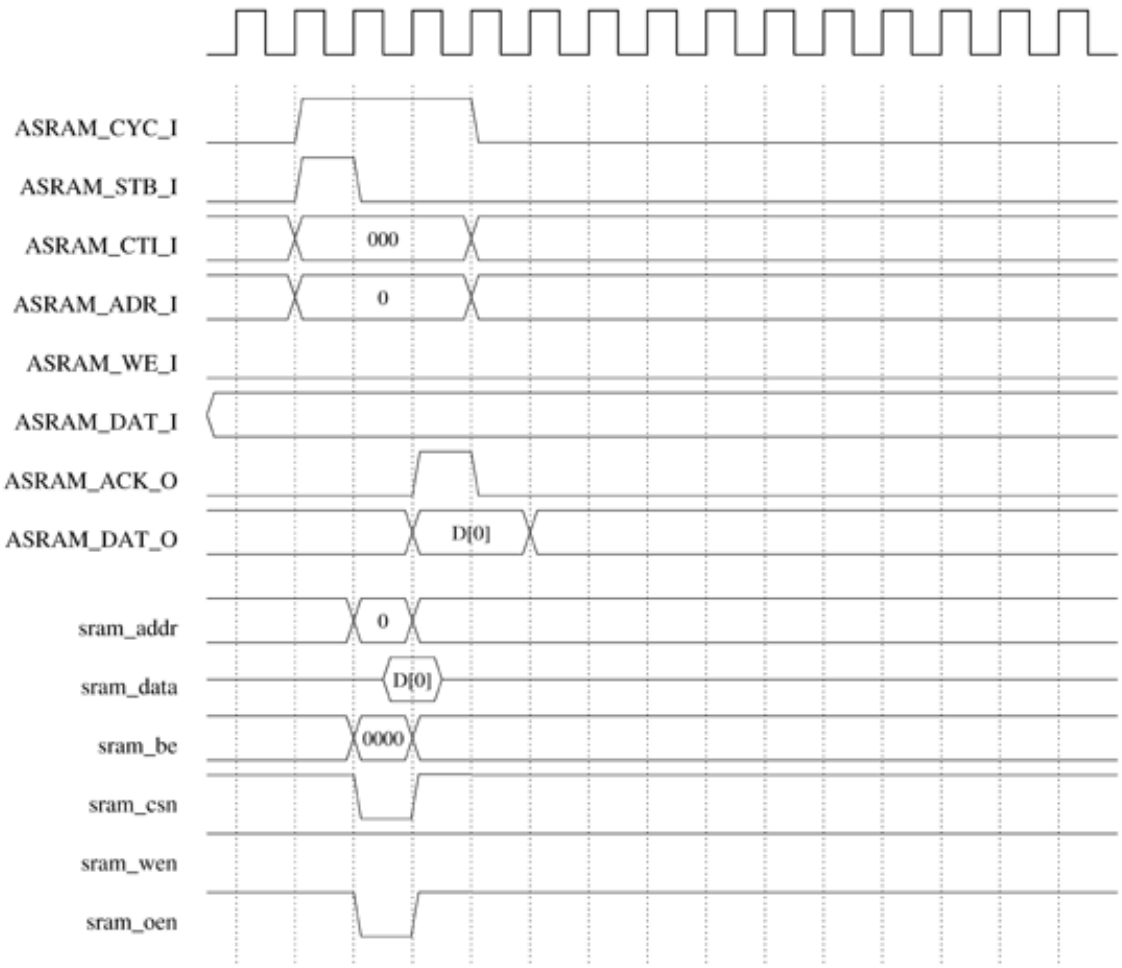


Figure 3 shows the asynchronous SRAM controller's timing for a WISHBONE 32-bit Classic Read and a asynchronous SRAM with a read latency of 1 and a data bus width of 32 bits. The read transaction begins with SRAM_CYC_I and SRAM_STB_I being asserted following a clock rising edge. The memory controller passes the address asserted on SRAM_ADR_I bus on the next clock rising edge. Since the latency is 1, data is expected from the asynchronous SRAM in the same cycle. Since the memory controller is configured to register the data from asynchronous SRAM controller, it drives the data on to the SRAM_DAT_O bus on the next clock rising edge.

Figure 4 shows the asynchronous SRAM controller's timing for a WISHBONE 32-bit Classic Write and an asynchronous SRAM with a write latency of 1 and a data bus width of 32 bits. The write transaction begins with SRAM_CYC_I

Figure 3: WISHBONE 32-Bit Classic Read (registered read)



and SRAM_STB_I being asserted following a clock rising edge. The memory controller passes the address, asserted on SRAM_ADR_I bus, and data, asserted on SRAM_DAT_I bus, on the next clock rising edge. Since the latency is 1, the asynchronous SRAM is expected to complete the write in the same cycle; therefore, the memory controller asserts SRAM_ACK_O in the same cycle.

Figure 4: WISHBONE 32-Bit Classic Write

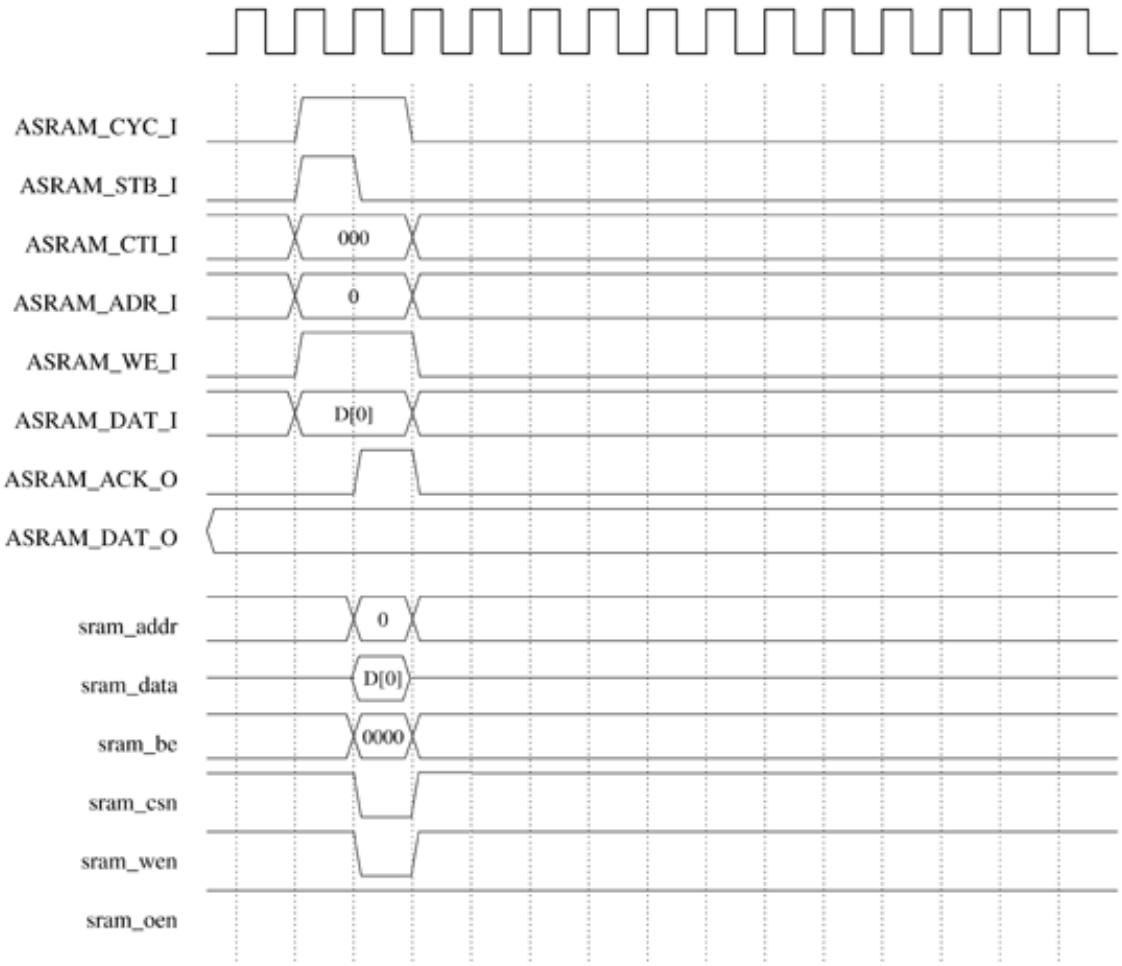


Figure 5 shows the asynchronous SRAM controller's timing for a WISHBONE 32-bit Burst Read and an asynchronous SRAM with a read latency of 1 and a data bus width of 32 bits. The read transaction begins with SRAM_CYC_I and SRAM_STB_I being asserted following a clock rising edge. The memory controller passes the address asserted on SRAM_ADR_I bus on the next clock rising edge. Since the latency is 1, data is expected from the asynchronous SRAM in the same cycle. The memory controller drives the data on to the SRAM_DAT_O bus and asserts the SRAM_ACK_O signal. The memory controller waits for the SRAM_STB_I to be asserted following the next clock rising edge before commencing another read from the asynchronous SRAM. This process is repeated until it detects the end of the burst read.

Figure 5: WISHBONE 32-Bit Burst Read (non-registered read)

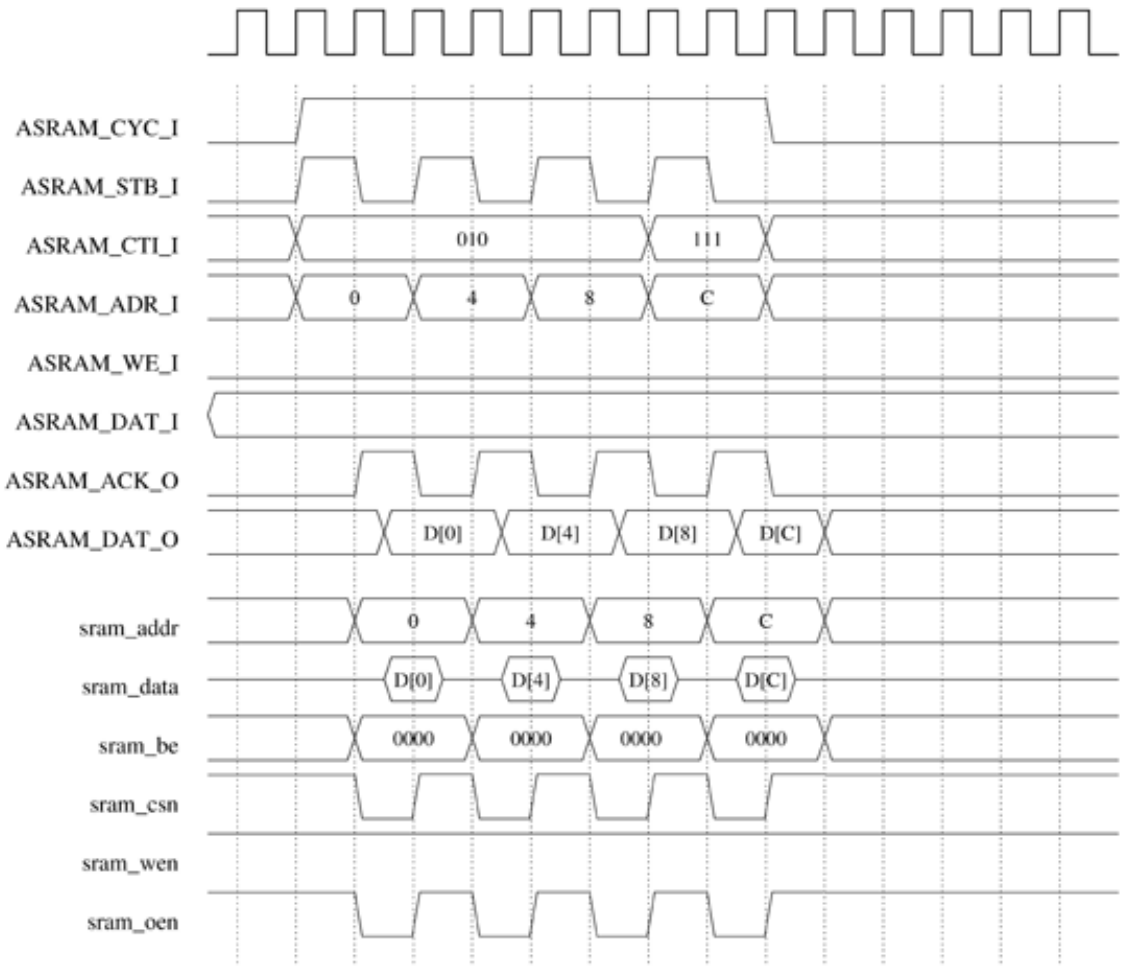


Figure 6 shows the asynchronous SRAM controller's timing for a WISHBONE 32-bit Burst Write and an asynchronous SRAM with a write latency of 1 and a data bus width of 32 bits. The write transaction begins with SRAM_CYC_I and SRAM_STB_I being asserted following a clock rising edge. The memory controller passes the address, asserted on SRAM_ADR_I bus, and data,

asserted on SRAM_DAT_I bus, on the next clock rising edge. Since the latency is 1, the asynchronous SRAM is expected to complete the write in the same cycle; therefore, the memory controller asserts SRAM_ACK_O in the same cycle. The memory controller waits for the SRAM_STB_I to be asserted following the next clock rising edge before commencing another write to the asynchronous SRAM. This process is repeated until it detects the end of the burst write.

Figure 6: WISHBONE 32-Bit Burst Write

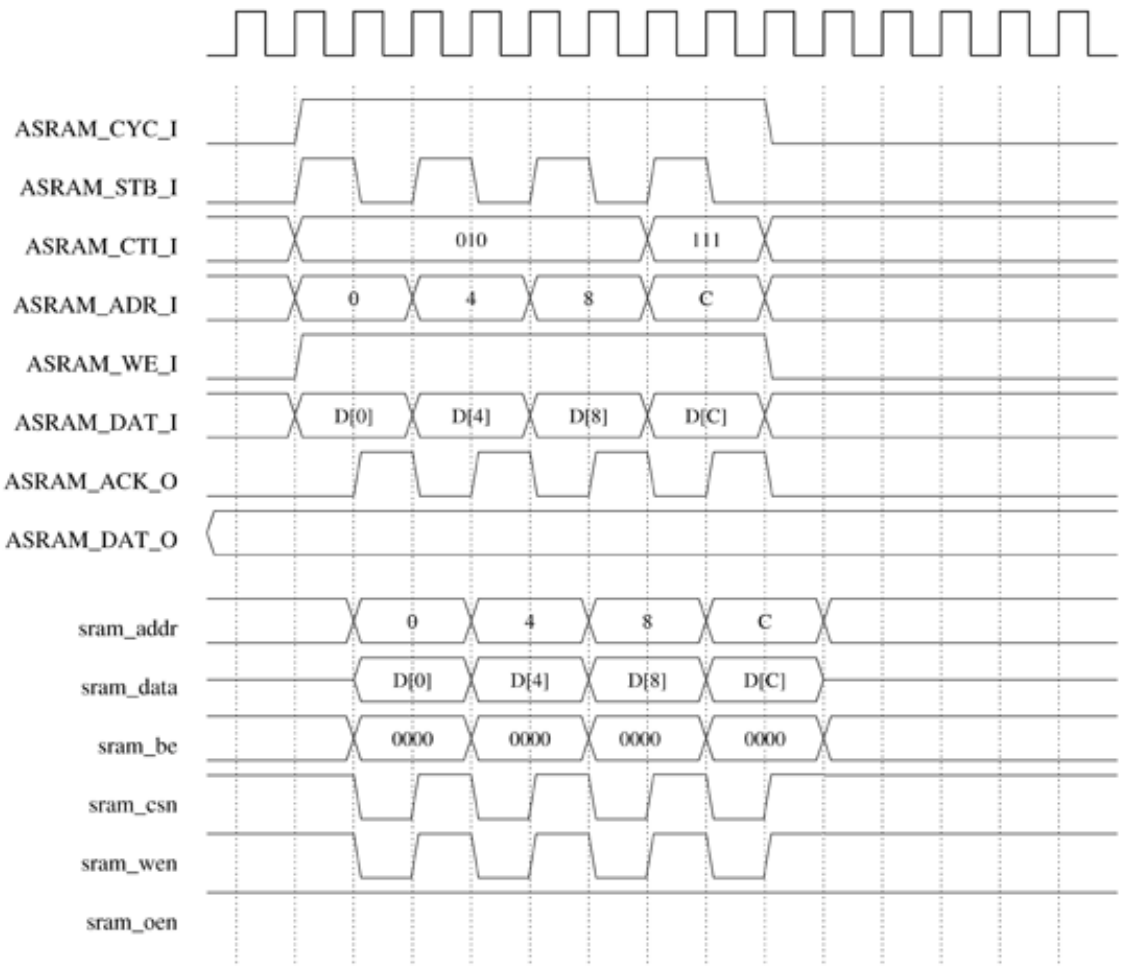
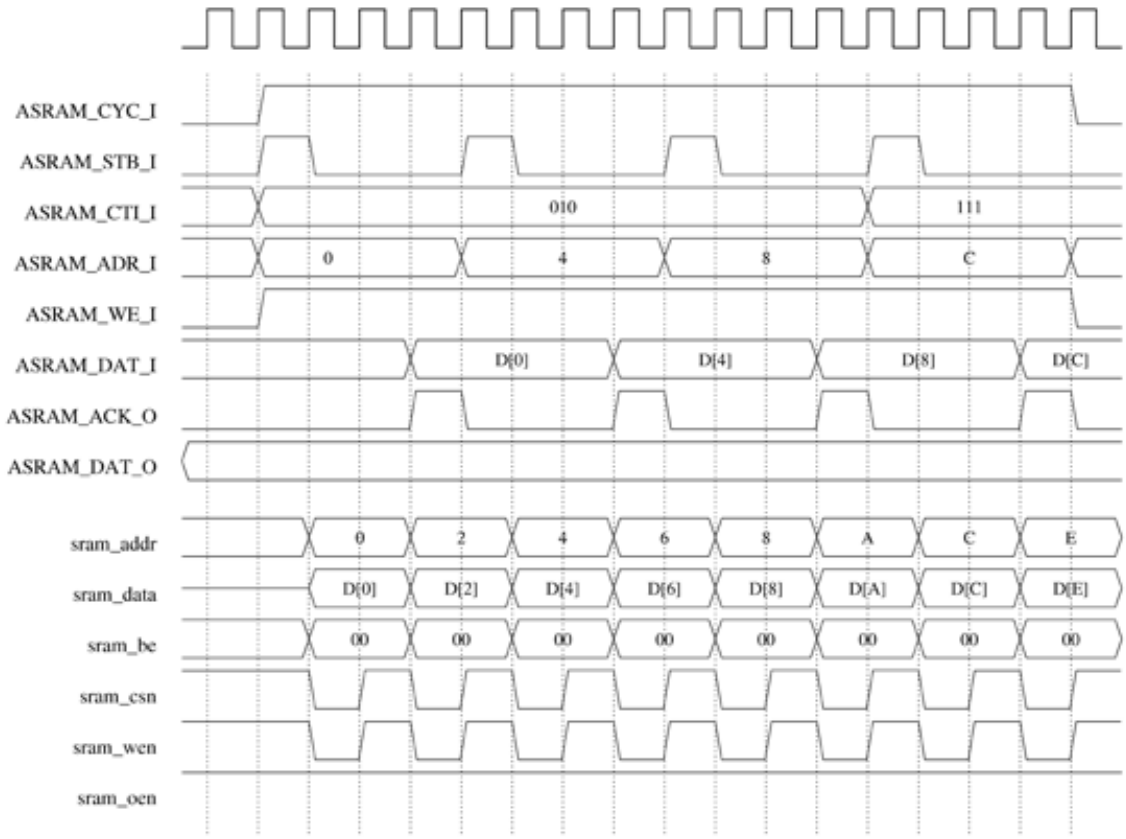


Figure 7 shows the asynchronous SRAM controller's timing for a WISHBONE 32-bit Burst Write and an asynchronous SRAM with a write latency of 1 and a data bus width of 16 bits. Each 32-bit WISHBONE write is translated into two 16-bit write transactions to the asynchronous SRAM. The write transaction begins with SRAM_CYC_I and SRAM_STB_I being asserted following a clock rising edge. The memory controller passes the address, asserted on SRAM_ADR_I bus, and data bits 31-16, asserted on SRAM_DAT_I bus, to the asynchronous SRAM on the next clock rising edge. Since the latency is 1, the asynchronous SRAM is expected to complete the write in the same cycle. The memory controller deasserts SRAM_CSN and SRAM_WEN on the next

clock rising edge. On the following clock edge, the memory controller asserts SRAM_CSN and SRAM_WEN again and increments the address to the next memory location. Since the latency is 1, the memory controller asserts SRAM_ACK_O in the same cycle indicating, that it can accept another write in the following cycle.

Figure 7: WISHBONE 32-Bit Burst Write (Asynchronous SRAM with 16-Bit Data Bus)



EBR Resource Utilization

The asynchronous SRAM controller uses no EBRs.

Software Support

The asynchronous SRAM controller does not require associated software support. It can access the memory's location by treating it as a general-purpose read/write memory.

Revision History

Component Version	Description
1.0	Initial release.
3.0 (7.0 SP2)	Corrected endianness. Added support for 8- and 16-bit operational modes.
3.1	Support added for 8 and 32-bit WISHBONE Data Bus. Corrected burst writes.
3.2	Fixed bug that incorrectly set up the data bus width. Updated document with new corporate logo.
3.3	▶ Added support for LatticeMico8-based designs in addition to LatticeMico32-based designs. ▶ Component can be used in designs that do not include a processor.

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, CleanClock, Custom Movile Device, DiePlus, E^2CMOS, Extreme Performance, FlashBAK, FlexiClock, flexiFLASH, flexiMAC, flexiPCS, FreedomChip, GAL, GDX, Generic Array Logic, HDL Explorer, iCE Dice, iCE40, iCE65, iCEblink, iCEcable, iCEchip, iCEcube, iCEcube2, iCEman, iCEprog, iCEsab, iCEsocket, IPexpress, ISP, ispATE, ispClock, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDX2, ispGDXV, ispGENERATOR, ispJTAG, ispLEVER, ispLeverCORE, ispLSI, ispMACH, ispPAC, ispTRACY, ispTURBO, ispVIRTUAL MACHINE, ispVM, ispXP, ispXPGA, ispXPLD, Lattice Diamond, LatticeCORE, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeECP2, LatticeECP2M, LatticeECP3, LatticeECP4, LatticeMico, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, LatticeXP2, MACH, MachXO, MachXO2, MACO, mobileFPGA, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, Platform Manager, ProcessorPM, PURESPEED, Reveal, SiliconBlue, Silicon Forest, Speedlocked, Speed Locking, SuperBIG, SuperCOOL, SuperFAST, SuperWIDE, sysCLOCK, sysCONFIG, sysDSP, sysHSI, sysI/O, sysMEM, The Simple Machine for Complex Design, TraceID, TransFR, UltraMOS, and specific product designations are either registered trademarks or trademarks of Lattice Semiconductor Corporation or its subsidiaries in the United States and/or other countries. ISP, Bringing the Best Together, and More of the Best are service marks of Lattice Semiconductor Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.