LatticeMico32 Asynchronous SRAM Controller

The LatticeMico32 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 3.0 version (formerly the 7.0 SP2 version) of the LatticeMico32 asynchronous SRAM controller.

Features

The LatticeMico32 asynchronous SRAM controller includes the following features:

- WISHBONE B.3 interface
- Configurable SRAM data bus width up to 32 bits
- Configurable SRAM address bus width up to 32 bits
- Configurable read latency
- Configurable write latency
- Support for burst-mode transfers according to the WISHBONE protocol

For additional details about the WISHBONE bus, refer to the *LatticeMico32 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. The controller interacts with the WISHBONE master port, using classic-mode registered-feedback bus cycle control strobes. For further information on the WISHBONE registered feedback bus cycle, refer to *WISHBONE Specifications, Version B3,* Chapter 4.

The memory controller has a configurable address bus and data bus width. The address bus can be up to 32 bits long. The data bus can be configured for 8-, 16-, or 32-bit widths. You can instantiate multiple SRAM controllers to permit access to memories of varying address and data bus sizes.

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.

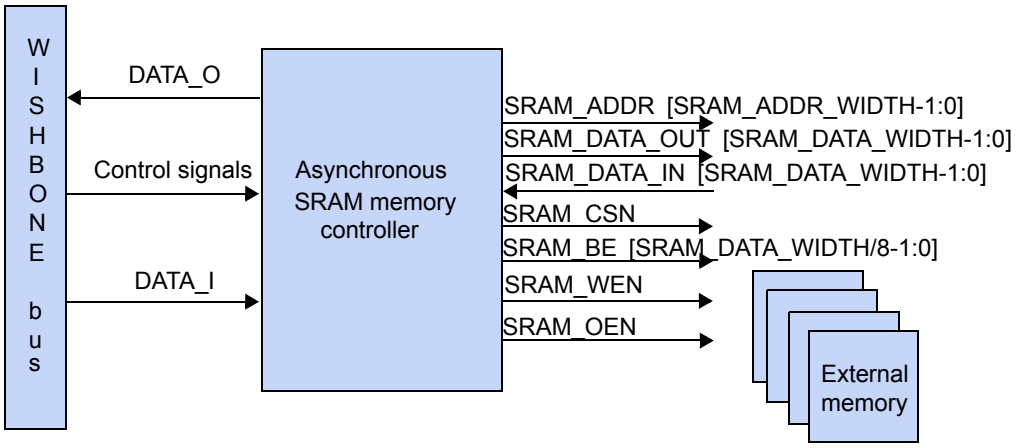
Since asynchronous RAM devices do not have a cycle acknowledge signal, the memory controller provides one. The ACK_O signal is controlled with a fixed read/write latency value. The read latency is independent of the write latency. Each increment in latency value represents an increase in the length of 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 LatticeMico32 asynchronous SRAM controller.

UI Parameters

Table 1 shows the UI parameters available for configuring the LatticeMico32 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			
Read Latency	Specifies the latency for reading the SRAM (measured in CLK_I cycles).	1 – 15	1

Table 1: Asynchronous SRAM Controller UI Parameters (Continued)

Dialog Box Option	Description	Allowable Values	Default Values
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 LatticeMico32 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

[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

I/O Ports

Table 3 describes the input and output ports of the LatticeMico32 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 [31:0]	—	I	0	WISHBONE data bus input (for write)
SEL_I [3:0]	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
LOCK_I	High	I	X	WISHBONE lock signal
ACK_O	High	O	0	Indicates the normal termination of a bus
DAT_O [31:0]	—	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

Table 3: Asynchronous SRAM Controller I/O Ports

I/O Port	Active	Direction	Initial State	Description
SRAM_DATA_IN [SRAM_DATA_WIDTH-1:0]	—	I	X	SRAM data input
SRAM_CSN	Low	O	1	SRAM chip select
SRAM_BE [3:0]]	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 and Figure 3 show the asynchronous SRAM controller's WISHBONE and external signals during a "latency = 1" read/write cycle.

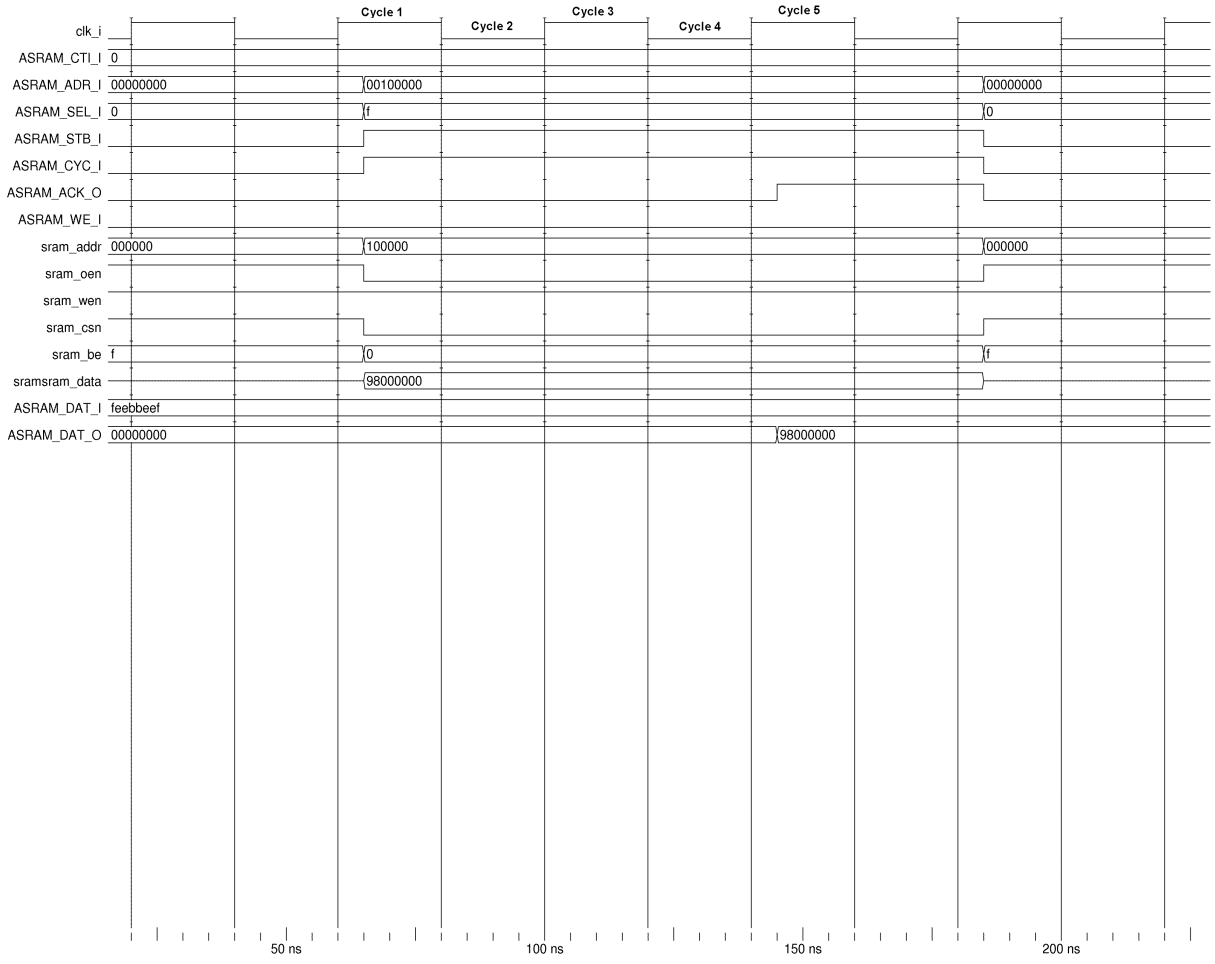
SRAM Read Logic

A read memory transaction begins with CYC_I and STB_I being asserted following a CLK_I rising edge, as shown in CLK_I cycle 1 in Figure 2. The memory controller passes the address asserted on ADR_I to the SRAM_ADDR bus when CYC_I and STB_I are asserted.

The memory controller counts CLK_I cycles until the read latency counter reaches its terminal count, when the ACK_O strobe is asserted, as shown in clock cycle 5 in Figure 2.

The memory controller latches the data driven by the asynchronous SRAM and drives it onto the DAT_O bus at the rising edge of ACK_O. DAT_O is therefore valid in CLK_I cycle 5, as shown in Figure 2.

Figure 2: Asynchronous SRAM Read



SRAM Write Logic

A write memory transaction begins with CYC_I and STB_I being asserted following a CLK_I rising edge, as shown in CLK_I cycle 1 in Figure 3. The memory controller passes the address asserted on ADR_I to the SRAM_ADDR bus when CYC_I and STB_I are asserted. The SRAM_DATA bus follows the DAT_I bus.

Once the WISHBONE cycle has started (that is, when CYC_I and STB_I are asserted) and the WE_I signal indicates a write cycle is in progress, the controller can assert the SRAM_WEN strobe. The soonest SRAM_WEN can be asserted is in cycle 3.

The memory controller counts CLK_I cycles until the write latency counter reaches its terminal count, when the ACK_O strobe is asserted. The

SRAM_WEN strobe is deasserted at the same time that ACK_O is asserted, as shown in clock cycle 7 in Figure 3.

The SRAM memory latches the data driven by the LatticeMico32 controller at the rising edge of the SRAM_WEN strobe. The rising edge of SRAM_WEN signals the completion of a single SRAM memory write transaction.

The asynchronous SRAM controller never uses the chip select to terminate a memory transaction.

Figure 3: Asynchronous SRAM Write

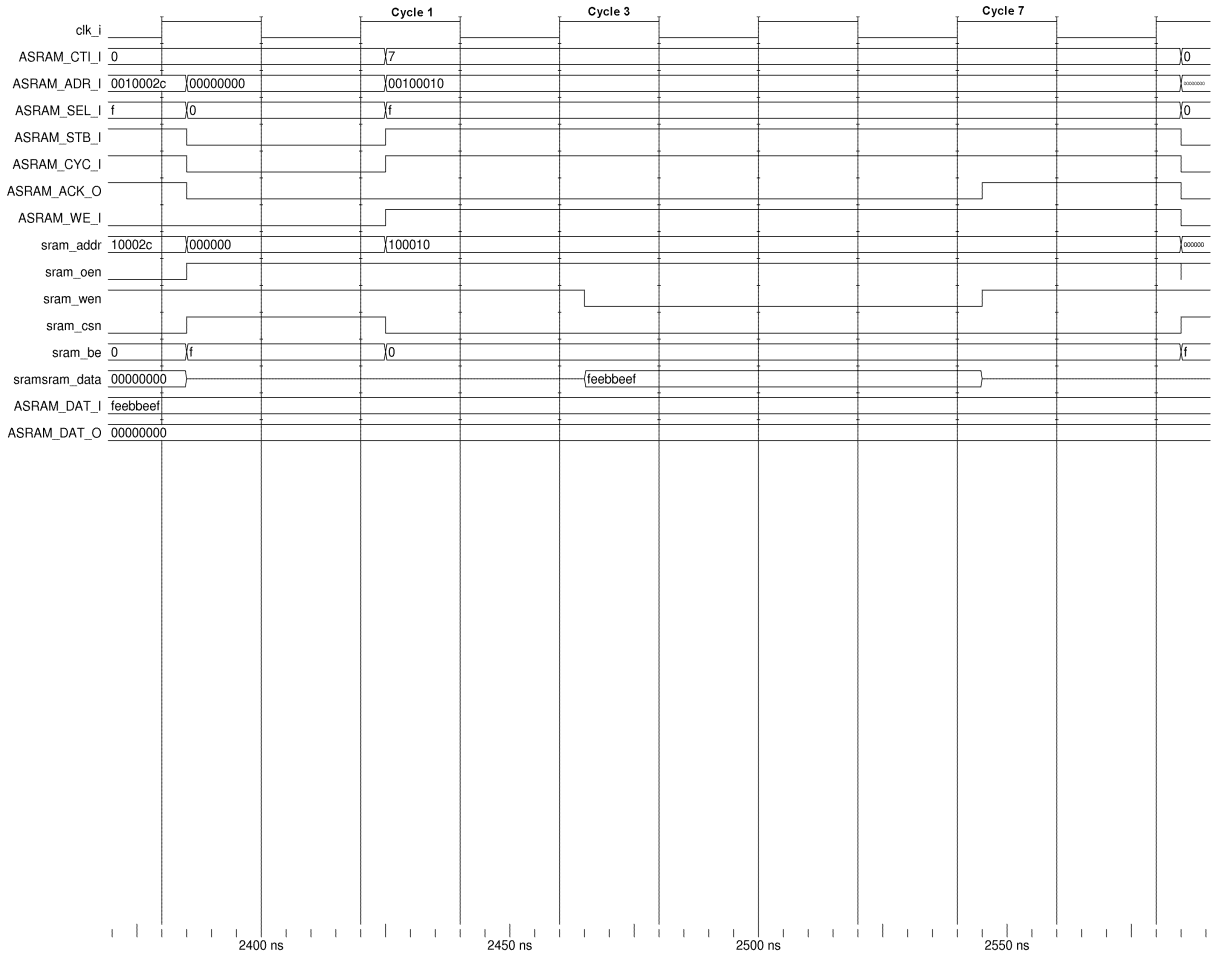


Figure 4 shows the port names and timing diagram of the asynchronous SRAM controller in burst 4 write mode.

Figure 4: Asynchronous SRAM Controller Burst 4 Write Timing Diagram

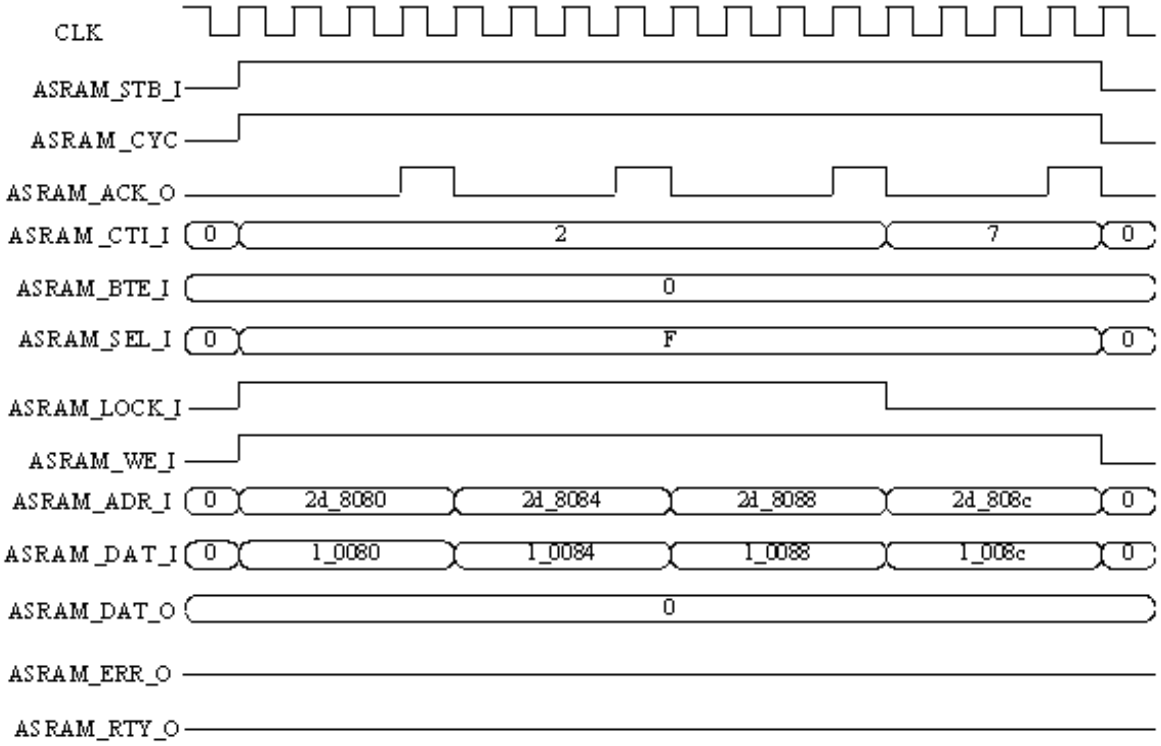
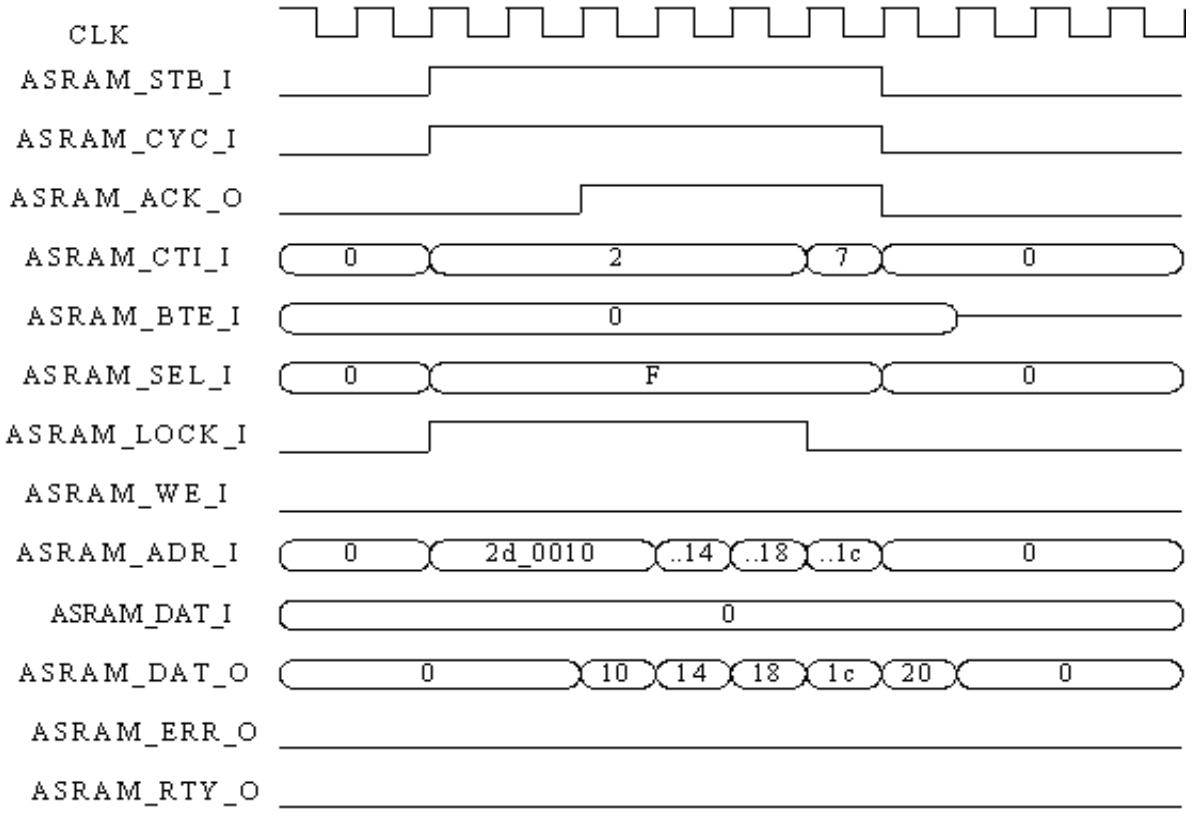


Figure 5 shows the port names and timing diagram of the asynchronous SRAM controller in burst 4 write mode.

Figure 5: Asynchronous SRAM Controller Burst 4 Read Timing Diagram



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.