LatticeMico32 Dual-Port On-Chip Memory Controller

The LatticeMico32 dual-port on-chip memory controller provides two slave interfaces to the WISHBONE bus master ports that allow the ports to access the Lattice Semiconductor FPGA embedded block RAMs (EBRs). The presence of two slave interfaces on the memory controller allows two simultaneous accesses from the bus master ports. The dual-port on-chip memory controller automatically instantiates the EBR using the parameterized module interface (PMI).

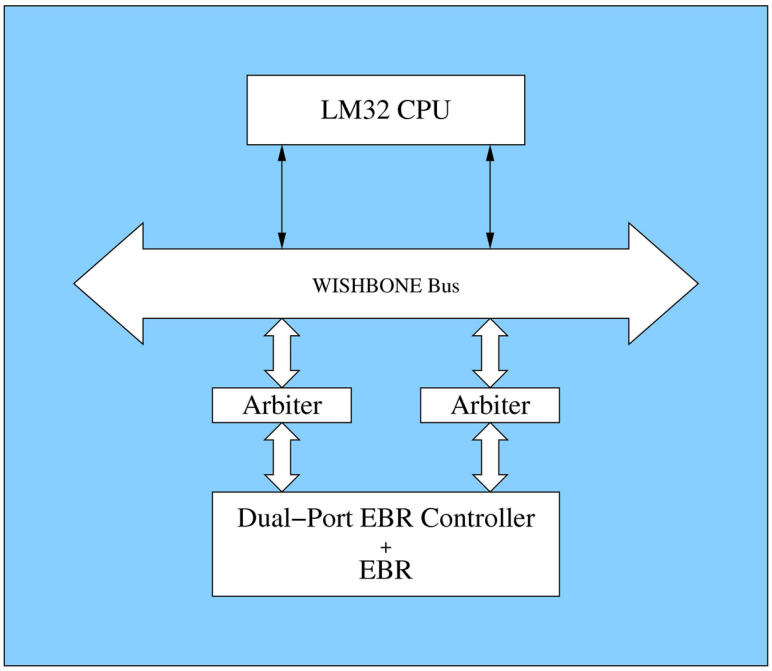
Features

The LatticeMico32 dual-port on-chip memory controller includes the following features:

- WISHBONE B.3 interface
- Two connectors between WISHBONE master ports and Lattice Semiconductor EBR memory
- Classic WISHBONE write and read bus operations
- Support for single transfer cycle with 8, 16, or 32-bit data transfers
- Support for linear increment burst cycle with 32-bit data transfers
- Automatic resolution of read/write conflicts when both slave interfaces access the same location in Lattice semiconductor EBR memory

As shown in Figure 1, the dual-port on-chip memory controller acts as a connector between the WISHBONE master ports and the Lattice Semiconductor EBR memory. For additional details about the WISHBONE bus, refer to the *LatticeMico32 Processor Reference Manual*.

Figure 1: LatticeMico32 Dual-Port On-Chip Memory Controller



Functional Description

The LatticeMico32 dual-port on-chip memory controller has two read/write slave interfaces: Port A and Port B. This allows two WISHBONE bus master devices to read from (or write to) the memory in the same cycle. The EBRs required to construct the memory are included with the memory controller. The maximum memory size or address range is limited by the available FPGA's EBR blocks.

The memory controller supports two types of WISHBONE read and write bus operations:

- 8, 16, or 32-bit classic cycles
- 32-bit linear increment burst cycle.

The memory controller is designed to handle the scenario in which both slave interfaces simultaneously access a given memory location. Table 1 shows the

outcome for such a scenario for different combinations of read/write operations on both the slave interfaces.

Table 1: Read/Write Operations on both Slave Interfaces

Port A	Port B	Internal Operation
Read	Read	Read operations proceed normally. Data is available on both ports in the same cycle.
Read	Write	Read and Write operations proceed normally. Data on Port A is the current data in location that is being simultaneously written to by Port B.
Write	Read	Read and Write operations proceed normally. Data on Port B is the current data in location that is being simultaneously written to by Port A.
Write	Write	A Write operation from Port B is stalled until the Write operation from Port A to the memory location is complete. Final data in memory is that of Port B.

Figure 2 on page 4 and Figure 3 on page 5 show the timing diagrams of the dual-port on-chip memory controller for classic read and write operations respectively. The width of the data transaction can be 8, 16, or 32 bits.

Figure 4 on page 6 shows the timing diagram of the dual-port on-chip memory controller for classic read and write operations in the scenario where accesses from both slave interfaces occur to the same location in memory. The memory controller obeys the rules shown in Table 1 on page 3. The width of the data transaction can be 8, 16, or 32 bits. In the scenario where classic write cycles are initiated to an identical memory lcoation from both slave interfaces in the same cycle, the write operation on slave interface B is delayed by a single cycle. Note that the WISHBONE ACK signal is delayed by one cycle. In the scenario where classic read cycles are initiated to an identical memory location from both slave interfaces in the same cycle, no such delay is introduced.

Figure 5 on page 7 and Figure 6 on page 8 show the timing diagram of the dual-port on-chip memory controller for linear incrementing burst read and write cycles respectively. The width of the data transaction can only be 32 bits.

Figure 7 on page 9 shows the timing diagram of the dual-port on-chip memory controller for linear incrementing burst read and write cycles in the scenario where accesses from both slave interfaces occur to the same location in memory. The width of the data transaction can only be 32 bits. The memory controller obeys the rules shown in Table 1 on page 3 in the scenario where linear incrementing burst cycles are initiated to an identical memory location from both slave interfaces in the same cycle. In the case of simultaneous write operations to the same memory location, a write from slave interface B is delayed by a single cycle. Note that the WISHBONE ACK signal is delayed by one cycle. In the case of simultaneous read operations to the same memory location, no such delay is introduced.

Figure 2: Dual-Port On-Chip Memory Controller Timing Diagram for Classic Read

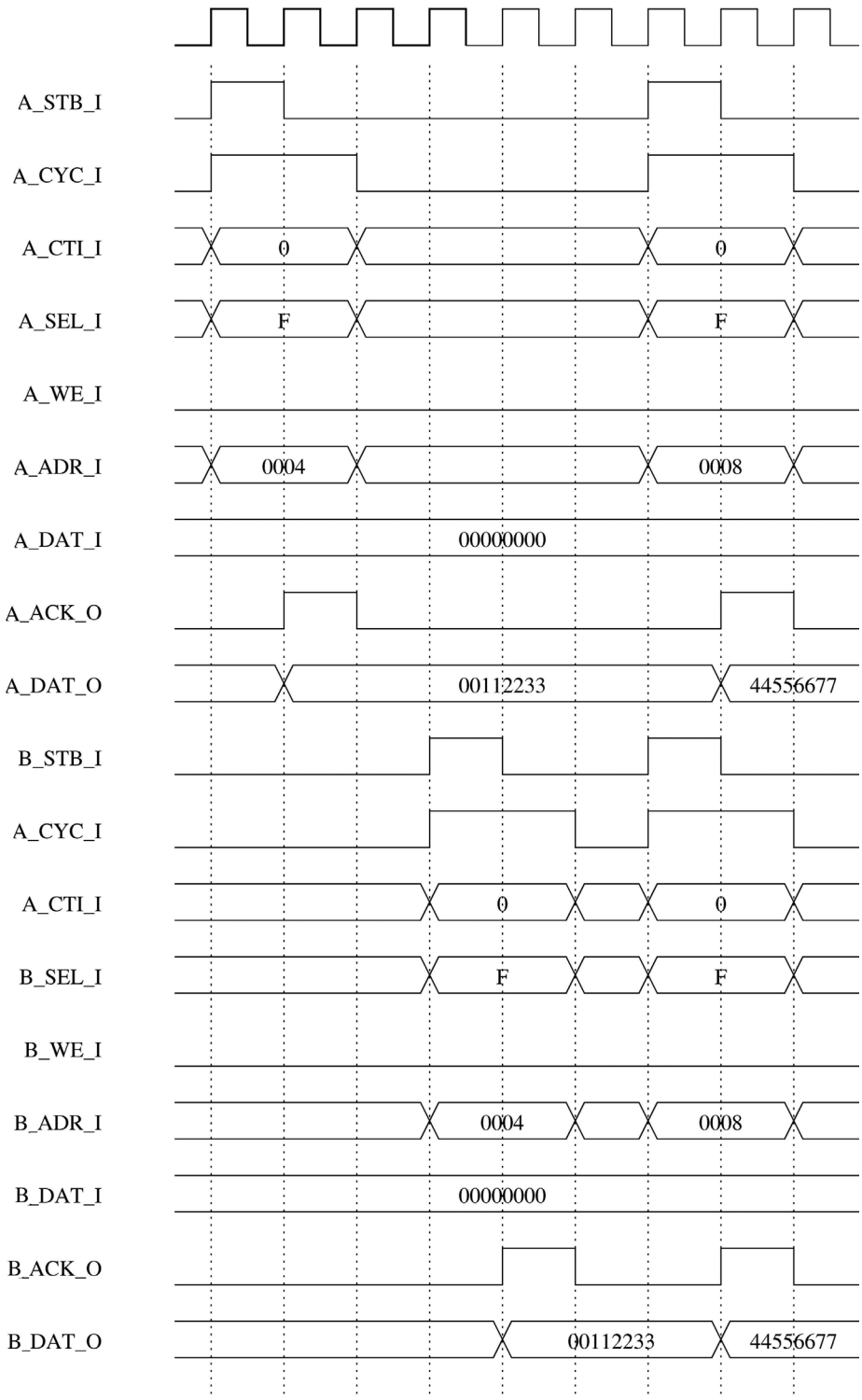


Figure 3: Dual-Port On-Chip Memory Controller Timing Diagram for Classic Write

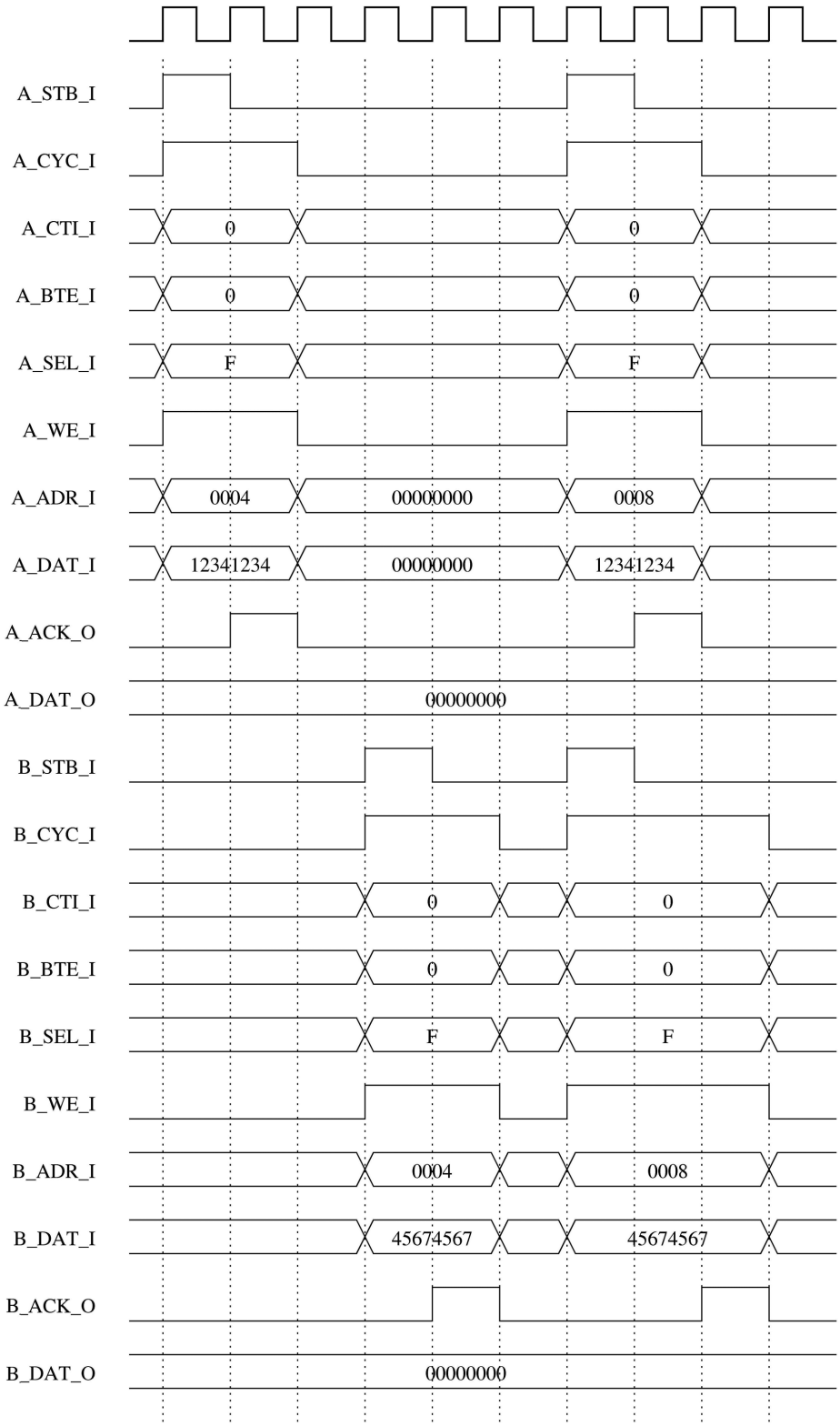


Figure 4: Dual Port On-chip Memory Controller Timing Diagram for Classic Read/Write to same Memory Addresses from both Slave Interfaces

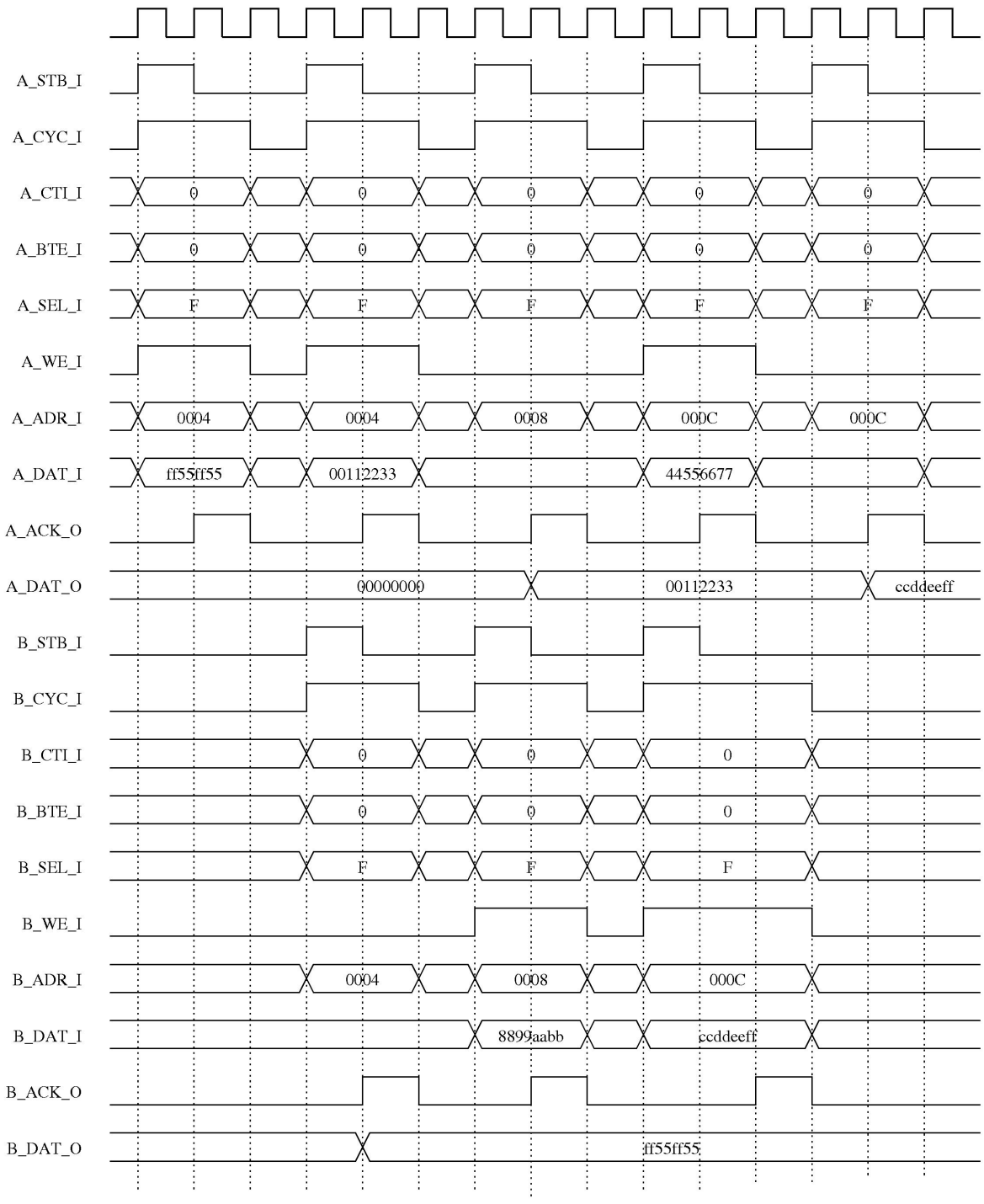


Figure 5: Dual-Port On-Chip Memory Controller Timing Diagram for Linear Incrementing Read Burst

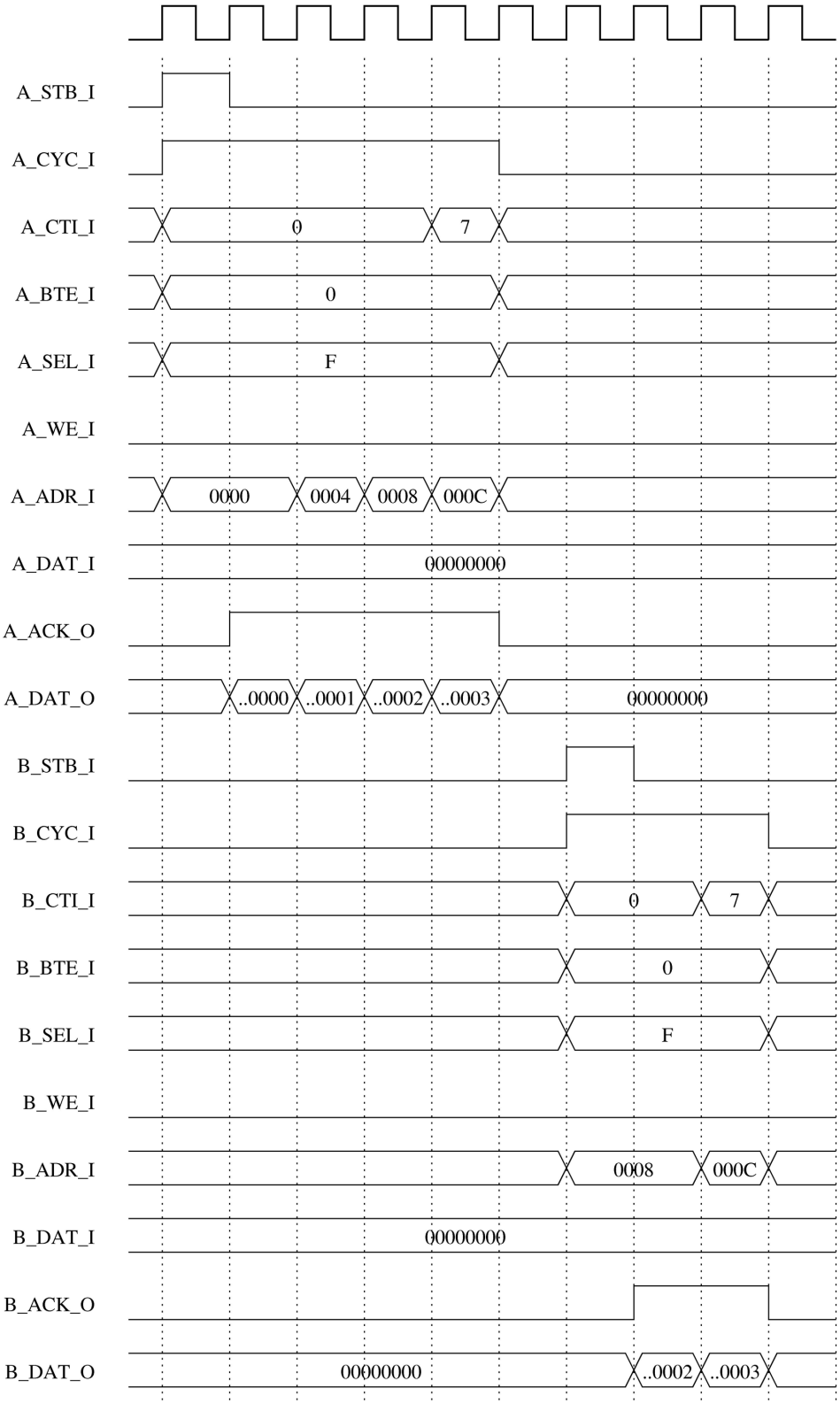


Figure 6: Dual-Port On-Chip Memory Controller Timing Diagram for Linear Incrementing Write Burst

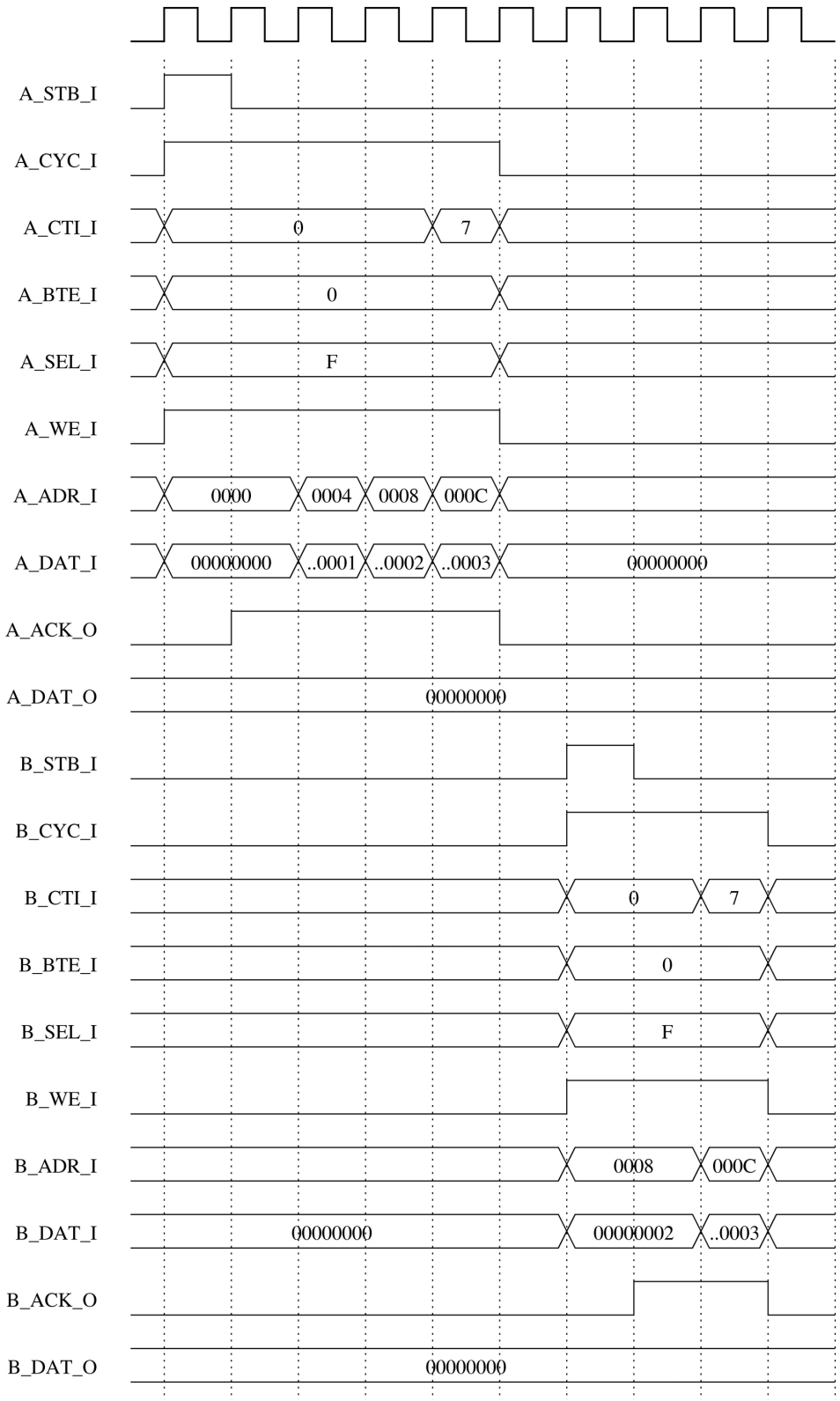
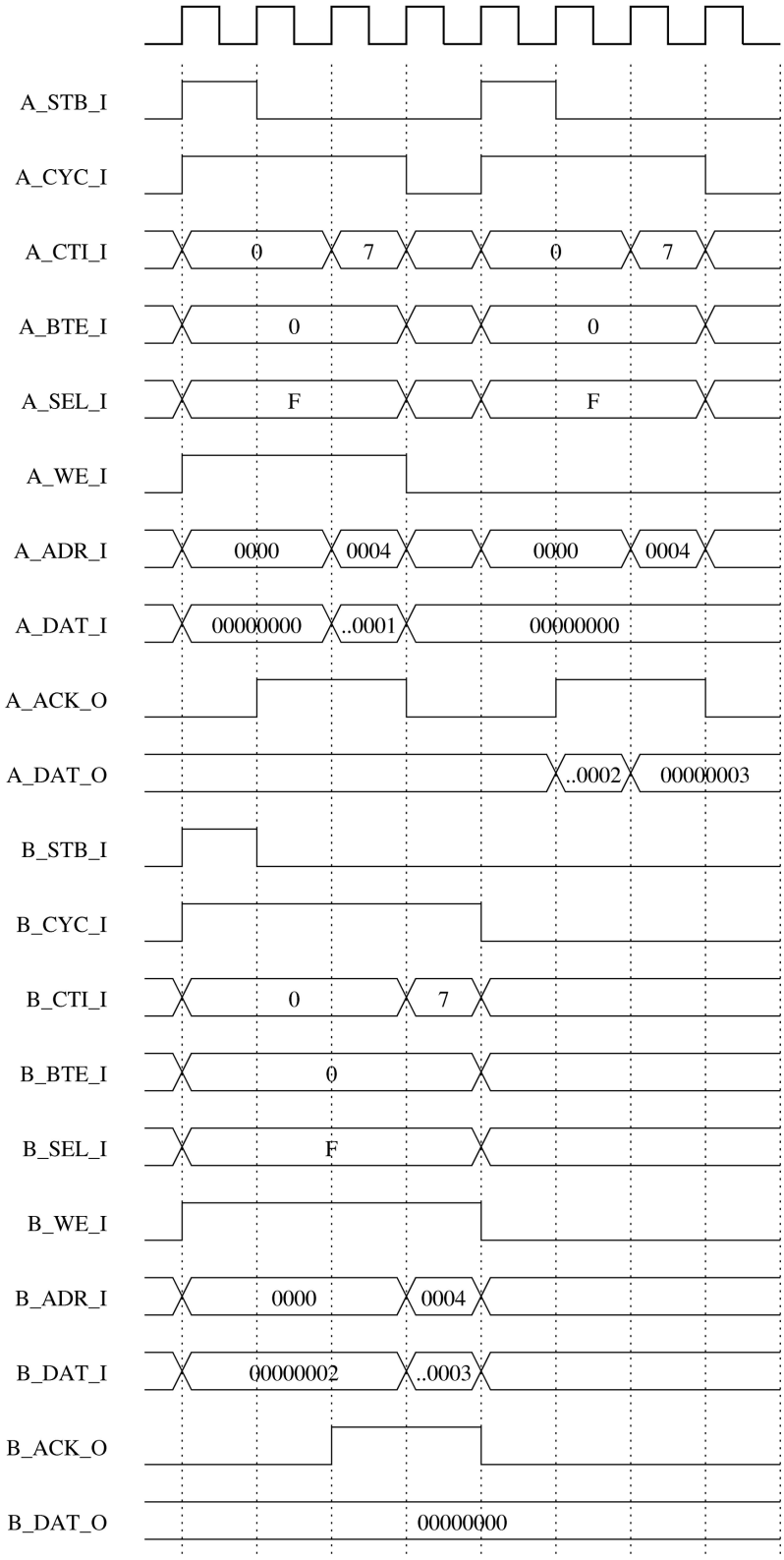


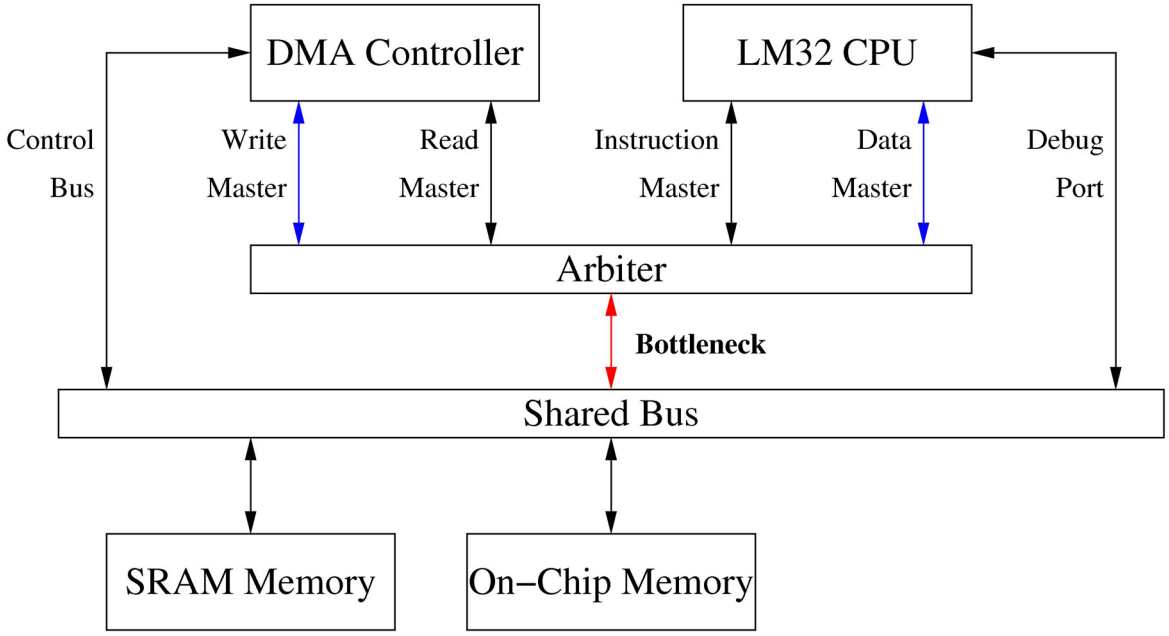
Figure 7: Dual-Port On-Chip Memory Controller Timing Diagram for Linear Incrementing Read/Write Burst to Same Memory Address from both Slave Interfaces



Usage Scenario

Consider the design in Figure 8. It employs shared-bus arbitration in which the arbiter grants control of the Wishbone bus to only one bus master in any given cycle. For example, if the DMA Controller's write master and the LM32 CPU bus both request access to the on-chip memory, only one will be granted access (i.e., the bottleneck is the arbiter). In such a design, the dual-port memory will not provide any performance benefits.

Figure 8: Platform with Single-Port On-Chip Memory and Shared-Bus Arbitration Policy



Now consider the design in Figure 9 on page 11. It employs slave-side arbitration in which an arbiter exists for each Wishbone slave in the design, and its purpose is to grant control of the slave to one of the bus masters in the design. For example, the arbiter dedicated to on-chip memory will arbitrate between all bus masters requesting access to the on-chip memory. When compared to the design in Figure 8, the bottleneck is now due to the limitation that the on-chip memory has only one read/write port. In such a design, the dual-port memory will provide performance benefits by allowing two bus masters to access the on-chip memory as shown in Figure 10 on page 11.

Figure 9: Platform with Single-Port On-Chip Memory and Slave-Side Arbitration Policy

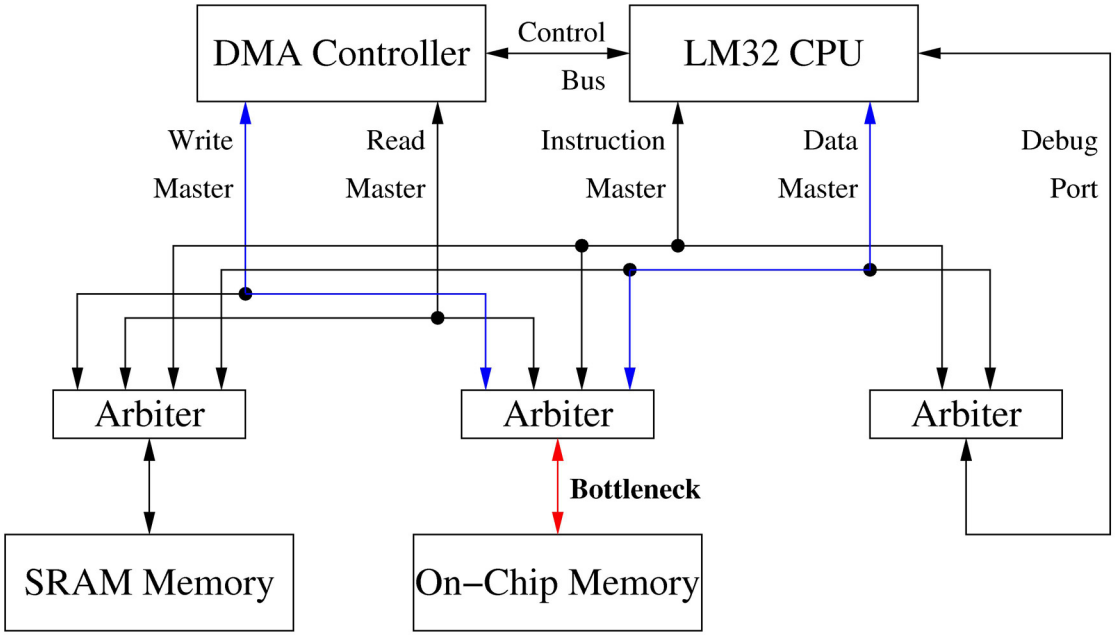
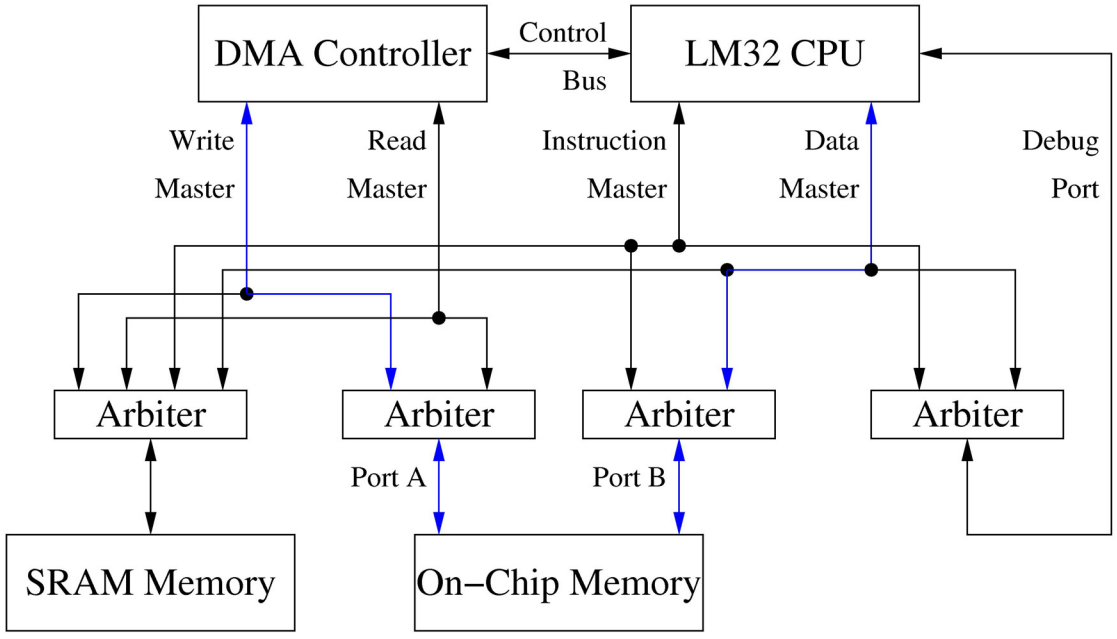


Figure 10: Platform with Dual-Port On-Chip Memory and Slave-Side Arbitration Policy



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 dual-port on-chip memory controller.

UI Parameters

Table 2 shows the UI parameters that are available for configuring the LatticeMico32 dual-port on-chip memory controller through the Mico System Builder (MSB) interface.

Table 2: Dual-Port On-Chip Memory Controller UI Parameters

Dialog Box Option	Description	Allowable Values	Default Value
Instance Name	Specifies the name of the dual-port on-chip memory controller instance.	Alphanumeric and underscores	ebr
Base Address	Specifies the base address for the device. The minimum boundary alignment is 0X00. The base address can be configured automatically in the LatticeMico32 Mico System builder.	0X00000000–0XFFFFFFFF If other components are included in the platform, the range of allowable values will vary.	0X00000000
Size of Memory (in Bytes)	Specifies the total size (depth x width) of the memory implemented using embedded block RAMs, in bytes (EBRs, in bits). The size is the limit of the EBR block in the selected FPGA.	0 – 72K	2048
Initialization File Name	Specifies the name of the memory initialization file. **Note**: The memory initialization file can be created from the software development environment. For a detailed description of the steps, refer to the “Deploying to On-Chip Memory” section of the *LatticeMico32 Software Developer User Guide*.	Valid file name	None
File Format	Specifies the format of the memory initialization file.	Hexadecimal, binary	Hexadecimal

Note: For LatticeEC/ECP devices, 1 EBR = 1 kb

Note

You can use the ispLEVER Memory Initialization Tool to change the content of initialized RAM (EBR) after synthesis, placement, and routing. For more information on the ispLEVER Memory Initialization Tool, choose **Help > Project Navigator > User Interface > Memory Initialization Dialog Box**.

HDL Parameters

Table 3 lists the parameters that appear in the HDL.

Table 3: Dual-Port On-Chip Memory Controller HDL Parameters

Parameter Name	Description	Allowable Values
BASE_ADDRESS	Specifies the base address for the device.	0X00000000–0XFFFFFFFF
SIZE	Specifies the total size of the memory, in bits (depth x width), implemented using embedded block RAMs (EBRs). The size is the limit of the EBR block in the selected FPGA.	0X00000000–0XFFFFFFFF
INIT_FILE_NAME	Specifies the name of the memory initialization file.	Valid file name
INIT_FILE_FORMAT	Specifies the format of the memory initialization file.	Hexadecimal, binary

I/O Ports

Table 4 describes the input and output ports of the LatticeMico32 dual-port on-chip memory controller.

Table 4: Dual-Port On-Chip Memory Controller I/O Ports

Signal Name	Active	Direction	Initial State	Description
CLK_I	HIGH	I	0	Input clock signal. The clock is rising-edge active.
RST_I	HIGH	I	0	System reset signal
WISHBONE Slave Interface Signals for Port A				
A_ADDR_I [31:0]	—	I	0	Address input array. Address is generated by the master.
A_DAT_I[31:0]	—	I	0	Data input array, which is valid for a write request.
A_WE_I	—	I	0	Write enable signal. Value is 1 for a write and 0 for a read.
A_CYC_I	HIGH	I	0	Cycle input. When asserted, it indicates that a bus cycle is in progress.
A_STB_I	HIGH	I	0	Strobe input. When asserted, it indicates that the slave is selected.
A_SEL_I[3:0]	HIGH	I	0	Select input array, which indicates where the valid data is expected on the data bus. The read operation reads the bytes that are EBR_SEL high. The write operation writes only the selected bytes, leaving the others unchanged.
A_CTI_I[2:0]	—	I	0	Cycle-type indication signal
A_BTE_I[1:0]	—	I	0	Burst-type extension signal

Table 4: Dual-Port On-Chip Memory Controller I/O Ports (Continued)

Signal Name	Active	Direction	Initial State	Description
A_LOCK_I	HIGH	I	0	Slave bus locked
A_ACK_O	HIGH	O	0	Acknowledge output. When asserted, it indicates normal cycle termination.
A_DAT_O[31:0]	—	O	0	Data output array
A_ERR_O	HIGH	O	0	Slave error, which is always 0.
A_RTY_O	HIGH	O	0	Slave retry, which is always 0.
WISHBONE Slave Interface Signals for Port B				
B_ADDR_I [31:0]	—	I	0	Address input array. Address is generated by the master.
B_DAT_I[31:0]	—	I	0	Data input array, which is valid for a write request.
B_WE_I	—	I	0	Write enable signal. Value is 1 for a write and 0 for a read.
B_CYC_I	HIGH	I	0	Cycle input. When asserted, it indicates that a bus cycle is in progress.
B_STB_I	HIGH	I	0	Strobe input. When asserted, it indicates that the slave is selected.
B_SEL_I[3:0]	HIGH	I	0	Select input array, which indicates where the valid data is expected on the data bus. The read operation reads the bytes that are EBR_SEL high. The write operation writes only the selected bytes, leaving the others unchanged.
B_CTI_I[2:0]	—	I	0	Cycle-type indication signal
B_BTE_I[1:0]	—	I	0	Burst-type extension signal
B_LOCK_I	HIGH	I	0	Slave bus locked
B_ACK_O	HIGH	O	0	Acknowledge output. When asserted, it indicates normal cycle termination.
B_DAT_O[31:0]	—	O	0	Data output array
B_ERR_O	HIGH	O	0	Slave error, which is always 0.
B_RTY_O	HIGH	O	0	Slave retry, which is always 0.

EBR Resource Utilization

The number of EBRs in the LatticeMico32 dual-port on-chip memory controller depends on the user-selected memory size.

- For LatticeECP2, LatticeXP2, LatticeECP2M, LatticeSC, and LatticeSCM devices, the number of EBRs is equal to:

  ceil (size in bytes/2048)

- For LatticeEC, LatticeECP, and LatticeXP devices, the number of EBRs is equal to:

  ceil (size in bytes/1024)

Software Support

The LatticeMico32 dual-port on-chip memory controller does not require associated software support. It can access a memory's location by treating it as a general-purpose read/write memory.