Table 1-12: **Traffic Generator Signal Descriptions**

Signal Name	Direction	Description
addr_mode_i[1:0]	Input	Valid settings for this signal are: `00`: Block RAM address mode. The address comes from the bram_cmd_i input bus. `01`: FIXED address mode. The address comes from the fixed_addr_i input bus. `10`: PRBS address mode (Default). The address is generated from the internal 32-bit LFSR circuit. The seed can be changed through the cmd_seed input bus. `11`: SEQUENTIAL address mode. The address is generated from the internal address counter. The increment is determined by the User Interface port width.
bl_mode_i[1:0]	Input	Valid settings for this signal are: `00`: Block RAM burst mode. The burst length comes from the bram_cmd_i input bus. `01`: FIXED burst mode. The burst length comes from the fixed_instr_i input bus. `10`: PRBS burst mode (Default). The burst length is generated from the internal 16-bit LFSR circuit. The seed can only be changed through the parameter section.
bram_cmd_i[38:0]	Input	This bus contains the block RAM interface ports: {BL, INSTR, ADDRESS}.
bram_rdy_o	Output	This block RAM interface output indicates when the traffic generator is ready to accept input from bram_cmd_i bus.
bram_valid_i	Input	For the block RAM interface, the bram_cmd_i bus is accepted when both bram_valid_i and bram_rdy_o are asserted.
clk_i	Input	This signal is the clock input.
cmd_seed_i[31:0]	Input	This bus is the seed for the command PRBS generator.
counts_rst	Input	When counts_rst is asserted, wr_data_counts and rd_data_counts are reset to zero.

Table 1-12: **Traffic Generator Signal Descriptions *(Cont'd)***

Signal Name	Direction	Description
data_mode_i[3:0]	Input	Valid settings for this signal are: `0000`: Reserved. `0001`: FIXED data mode. Data comes from the fixed_data_i input bus. `0010`: DGEN_ADDR (Default). The address is used as the data pattern. `0011`: DGEN_HAMMER. All 1s are on the DQ pins during the rising edge of DQS, and all 0s are on the DQ pins during the falling edge of DQS. This option is only valid if parameter DATA_PATTERN = "DGEN_HAMMER" or "DGEN_ALL". `0100`: DGEN_NEIGHBOR. All 1s are on the DQ pins during the rising edge of DQS except one pin. The address determines the exception pin location. This option is only valid if parameter DATA_PATTERN = "DGEN_ADDR" or "DGEN_ALL". `0101`: DGEN_WALKING1. Walking 1s are on the DQ pins. The starting position of 1 depends on the address value. This option is only valid if parameter DATA_PATTERN = "DGEN_WALKING" or "DGEN_ALL". `0110`: DGEN_WALKING0. Walking 0s are on the DQ pins. The starting position of 0 depends on the address value. This option is only valid if parameter DATA_PATTERN = "DGEN_WALKING0" or "DGEN_ALL". `0111`: DGEN_PRBS. A 32-stage LFSR generates random data and is seeded by the starting address. This option is only valid if parameter DATA_PATTERN = "DGEN_PRBS" or "DGEN_ALL".
data_seed_i[31:0]	Input	This bus is the seed for the data PRBS generator.
end_addr_i[31:0]	Input	This bus defines the end-address boundary for the port address space. The least-significant bits [3:0] are ignored.
error	Output	This signal is asserted when the readback data is not equal to the expected value.
error_status[n:0]	Output	This signal latches these values when the error signal is asserted: [31:0]: Read start address [37:32]: Read burst length [39:38]: Reserved [40]: mcb_cmd_full [41]: mcb_wr_full [42]: mcb_rd_empty [64 + (DWIDTH - 1):64]: expected_cmp_data [64 + (2*DWIDTH - 1):64 + DWIDTH]: read_data
fixed_addr_i[31:0]	Input	This 32-bit input is the fixed address input bus.
fixed_bl_i[5:0]	Input	This 6-bit input is the fixed burst length input bus.
fixed_data_i[31:0]	Input	This 32-bit input is the fixed data input bus.
fixed_instr_i[2:0]	Input	This 3-bit input is the fixed instruction input bus.

Table 1-12: **Traffic Generator Signal Descriptions *(Cont'd)***

Signal Name	Direction	Description
instr_mode_i[3:0]	Input	Valid settings for this signal are: `0000`: Block RAM instruction mode. The instruction comes from the bram_cmd_i input bus. `0001`: FIXED instruction mode. The instruction comes from the fixed_instr_i input bus. `0010`: W/R instruction mode (Default). This mode generates pseudo-random WRITE and READ instruction sequences. `0011`: WP/RP instruction mode. This mode generates pseudo-random WRITE precharge and READ precharge instruction sequences. `0100`: W/WP/R/RP. This mode generates pseudo-random WRITE, WRITE precharge, READ, and READ precharge instruction sequences. `0101`: W/WP/R/RP/RF. This mode generates pseudo-random WRITE, WRITE precharge, READ, READ precharge, and REFRESH instruction sequences.
mcb_cmd_addr_o[29:0]	Output	MCB's Command port interface.
mcb_cmd_bl_o[5:0]	Output	MCB's Command port interface.
mcb_cmd_en_o	Output	MCB's Command port interface.
mcb_cmd_full_i	Input	MCB's Command port interface.
mcb_cmd_instr_[2:0]	Output	MCB's Command port interface.
mcb_rd_data_i[DWIDTH-1:0]	Input	MCB's Data port interface.
mcb_rd_empty_i	Input	MCB's Data port interface.
mcb_rd_en_o	Input	MCB's Data port interface.
mcb_wr_data_o[DWIDTH-1:0]	Output	MCB's Data port interface.
mcb_wr_en_o	Output	MCB's Data port interface.
mcb_wr_full_i	Input	MCB's Data port interface.
mode_load_i	Input	When this signal is asserted (High), the values in addr_mode_i, instr_mode_i, bl_mode_i, and data_mode_i are latched and the next traffic pattern is based on the new settings.
rd_data_counts[47:0]	Output	The value of this bus is incremented when data is read from the MCB's read data port.
rst_i	Input	This signal is the Reset input.
run_traffic_i	Input	When this signal is asserted (High), the traffic generator starts generating command and data patterns. This signal should be only be asserted when mode_load_i is *not* asserted.
start_addr_i[31:0]	Input	This input defines the start address boundary for the port address space. The least-significant bits [3:0] are ignored.
wr_data_counts[47:0]	Output	The value of this output is incremented when data is sent to the MCB's write data port.

Modifying Port Address Space

The address space for a port can be easily modified by changing the BEGIN_ADDRESS and END_ADDRESS parameters found in the top-level test bench file. These two values must be set to align to the port data width. The two additional parameters, PRBS_SADDR_MASK_POS and PRBS_EADDR_MASK_POS, are used in the default PRBS address mode to ensure that out-of-range addresses are not sent to the port.

The PRBS_SADDR_MASK_POS parameter creates an OR mask that shifts PRBS generated addresses with values below BEGIN_ADDRESS up into the valid address space of the port. It should be set to a 32-bit value equal to the BEGIN_ADDRESS parameter. The PRBS_EADDR_MASK_POS parameter creates an AND mask that shifts PRBS generated addresses with values above END_ADDRESS down into the valid address space of the port. It should be set to a 32-bit value, where all bits above the most-significant address bit of END_ADDRESS are set to 1 and all remaining bits are set to 0. Table 1-13 shows some examples of setting the two mask parameters.

Table 1-13: **Example Settings for Address Space and PRBS Masks**

SADDR	EADDR	PRBS_SADDR_MASK_POS	PRBS_EADDR_MASK_POS
0x1000	0xFFFF	0x00001000	0xFFFF0000
0x2000	0xFFFF	0x00002000	0xFFFF0000
0x3000	0xFFFF	0x00003000	0xFFFF0000
0x4000	0xFFFF	0x00004000	0xFFFF0000
0x5000	0xFFFF	0x00005000	0xFFFF0000
0x2000	0x1FFF	0x00002000	0xFFFFE000
0x2000	0x2FFF	0x00002000	0xFFFFD000
0x2000	0x3FFF	0x00002000	0xFFFFC000
0x2000	0x4FFF	0x00002000	0xFFFF8000
0x2000	0x5FFF	0x00002000	0xFFFF8000
0x2000	0x6FFF	0x00002000	0xFFFF8000
0x2000	0x7FFF	0x00002000	0xFFFF8000
0x2000	0x8FFF	0x00002000	0xFFFF0000
0x2000	0x9FFF	0x00002000	0xFFFF0000
0x2000	0xAFFF	0x00002000	0xFFFF0000
0x2000	0xBFFF	0x00002000	0xFFFF0000
0x2000	0xCFFF	0x00002000	0xFFFF0000
0x2000	0xDFFF	0x00002000	0xFFFF0000
0x2000	0xEFFF	0x00002000	0xFFFF0000
0x2000	0xFFFF	0x00002000	0xFFFF0000

Custom Command Sequences

The traffic generator can send a custom command sequence to the User Interface port by reading address, instruction, and burst length values directly from a block RAM via the bram_cmd_i input bus. The CMD_PATTERN parameter in the Traffic Generator module must be set to "CGEN_ALL" (default) or "CGEN_BRAM" for this mode of operation. In the CGEN_ALL case, the addr_mode_i, instr_mode_i, and bl_mode_i inputs must be set to their respective block RAM mode values.

The bram_cmd_i input bus is a combination of the burst length, instruction, and address values as follows:

bram_cmd_i[38:0] = {BL[5:0], INSTR[2:0], ADDRESS[29:2]}

Address bits [1:0] and [31:30] are padded with 0s. The traffic generator accepts the bram_cmd_i value when both bram_valid_i and bram_rdy_o are asserted (High).

The command patterns instr_mode_i, addr_mode_i and bl_mode_i of the `traffic_gen` module can each be set independently. The provided `init_mem_pattern_ctr` module has interface signals to allow the command pattern to be modified in real time using the ChipScope tool's VIO. To change command pattern:

1. Set vio_modify_enable to "1".
2. Set vio_addr_mode_value to:
   - 0: bram address input.
   - 1: fixed_address.
   - 2: prbs address.
   - 3: sequential address.
3. Set vio_bl_mode_value to:
   - 0: bram bl input.
   - 1: fixed bl.
   - 2: prbs bl. If bl_mode value is set to 2, the addr_mode value is forced to 2.
4. Set vio_fixed_bl_value to: 1 — 64.

Memory Initialization and Traffic Test Flow

After power up, the Init Memory Control block directs the traffic generator to initialize the memory with the selected data pattern through the memory initialization procedure.

Memory Initialization

1. The data_mode_i input is set to select the data pattern (for example, data_mode_i[3:0] = `0010` for the address as the data pattern).
2. The start_addr_i input is set to define the lower address boundary.
3. The end_addr_i input is set to define the upper address boundary.
4. bl_mode_i is set to `01` to get the burst length from the fixed_bl_i input.
5. The fixed_bl_i input is set to either 16 or 32.
6. instr_mode_i is set to `0001` to get the instruction from the fixed_instr_i input.
7. The fixed_instr_i input is set to the "WR" command value of the memory device.
8. addr_mode_i is set to `11` for the sequential address mode to fill up the memory space.
9. mode_load_i is asserted for one clock cycle.

When the memory space has been initialized with the selected data pattern, the Init Memory Control block instructs the traffic generator to begin running traffic through the traffic test flow procedure (by default, the addr_mode_i, instr_mode_i, and bl_mode_i inputs are set to select PRBS mode).

Traffic Test Flow

1. The addr_mode_i input is set to the desired mode (PRBS is the default).
2. The cmd_seed_i and data_seed_i input values are set for the internal PRBS generator. This step is not required for other patterns.
3. The instr_mode_i input is set to the desired mode (PRBS is the default).
4. The bl_mode_i input is set to the desired mode (PRBS is the default).
5. The data_mode_i input should have the same value as in the memory pattern initialization stage detailed in Memory Initialization.
6. The run_traffic_i input is asserted to start running traffic.
7. If an error occurs during testing (that is, the read data does not match the expected data), the error bit is set until reset is applied.
8. Upon an error, the error_status bus latches the values defined in Table 1-12, page 51.

With some modifications, the example design can be changed to allow addr_mode_i, instr_mode_i, and bl_mode_i to be changed dynamically when run_traffic_i is deasserted. However, after changing the setting, the memory initialization steps need to be repeated to ensure the proper pattern is loaded into the memory space.

EDK Flow Details

This chapter describes how to use the MIG tool available in Xilinx® Platform Studio (XPS). It contains these sections:

- EDK Overview
- AXI Spartan-6 FPGA DDRx Memory Controller

EDK Overview

The Embedded Development Kit (EDK) provides an alternative package to the RTL than that of the MIG tool flow in the CORE Generator™ interface. The XPS IP Catalog contains the IP core axi_s6_ddrx with the same RTL that is provided by the MIG tool. The difference is that the RTL is packaged as an EDK pcore suitable for use in embedded processor based systems. The axi_s6_ddrx pcore only provides an AXI4 slave interface for each of the ports that are enabled. If a native MCB port is needed, refer to the Multi-Port Memory Controller (MPMC) IP provided by EDK as an alternative.

The axi_s6_ddrx IP is configured using the same MIG tool that is used in the CORE Generator tool. The GUI flow is the same as described in the MIG Overview of Chapter 1. However, instead of generating the UCF/RTL, the MIG tool sets the parameters for the RTL in the XPS MHS file. From the parameters, the pcore can generate the correct constraints for itself during platgen. Because the pcore is only a component in the system, the clock/reset structure must also be configured in XPS as it is not automatically generated as is done in the CORE Generator tool RTL. After the IP is configured and the ports are connected, the XPS tool is relied on to perform all other aspects of IP management including generating a bitstream and running simulations. For more information about EDK and XPS, see *EDK Concepts, Tools, and Techniques* [Ref 2] and *Embedded System Tools Reference Guide* [Ref 3].

The simplest way to get started with the axi_s6_ddrx memory controller is to use the base system builder (BSB) wizard in XPS. The BSB guides the user through a series of options to provide an entire embedded project with an optional axi_s6_ddrx memory controller. If the memory controller is selected, an already configured, connected, and tested axi_s6_ddrx controller is provided for a particular reference board, such as the SP601 and SP605 boards.

AXI Spartan-6 FPGA DDRx Memory Controller

The Advanced eXtensible Interface (AXI) Spartan®-6 FPGA DDRx Memory Controller core provides a high-performance multi-ported AXI4 slave front-end connection to LPDDR SDRAM/DDR/DDR2/DDR3 external memory. This core use the Memory Control Block (MCB) primitive and adapts the MCB native interface to use the AXI4 slave interface. This provides full functionality of all the features present on the Spartan-6 FPGA MCB core.

Feature Overview

In addition to the MCB feature set, the AXI features include:

- Supports read-only and write-only modes.
- Supports AXI4 INCR/WRAP transactions.
- Supports a mode to guarantee write coherency between ports.
- Does not reorder transactions.
- Round-Robin Read/Write arbitration.
- Little-endian AXI4 slave interface.
- One up to six AXI4 slave compliant memory interface(s).
- AXI4 slave interface running 1:1 clock rate to the Spartan-6 FPGA MCB controller port interface (can be asynchronous to memory).
- AXI4 slave interface data width of 32, 64, or 128 bits. AXI4 data width cannot be greater than the MCB native data width.
- Support for all MCB-supported memories (LPDDR, DDR, DDR2, and DDR3).
- Support for AXI4 long bursts up to 256 data beats.

Feature Description and AXI Protocol Support

This section describes how the AXI Spartan-6 FPGA DDRx Memory Controller interprets and supports the AXI4 specification. These interpretations of the AXI4 specification as it relates to a memory controller follow the Xilinx design conventions that balance performance, size, and complexity.

Interface Width

The AXI Read and Write data width can be 32, 64, or 128. It must be equal to the MCB data width. The MCB data width can be 32, 64, or 128 bits, depending on MCB configuration.

Interface Clock

Each AXI4 slave interface can run with a completely independent clock from each other and from the memory clock. All AXI channels and interface logic within a specific AXI4 slave interface use the same clock, with no additional clock conversion before passing into the associated MCB port.

Address Width

The address width must be parameterized to support the desired system address bus width. If the system address bus is defined wider than the memory size, it is acceptable to alias/wrap the memory across the address space. The MCB interface supports a maximum of 30 bits for the address bus. The MSB of the AXI address is cut off, if necessary. A 32-bit constant address width is used for compatibility with EDK. The address also wraps if the address range specified by the base and high address is smaller than the memory size.

Read-Only or Write-Only AXI Ports

Each AXI4 interface can be configured as Read-only or Write-only even when connected to a bidirectional MCB port. This permits logic optimization when bidirectional data flow is not required. The Read-only or Write-only AXI port is required when connected to a unidirectional MCB port. When placed in Read-only or Write-only mode, unnecessary Read/Write arbitration logic and datapath logic are removed. If the MCB port is natively a bidirectional port, the MIG GUI and source RTL allow the user to choose a Read only or Write only AXI4 interface for FPGA resource savings.

Reset

The AXI4 interface has a single synchronous reset, active Low signaling, that resets the entire core and brings it to a known initialized state. A reset event causes a full reset including recalibration of the controller.

Bursts

These rules apply:

- The AXI Spartan-6 FPGA DDRx Memory Controller supports `INCR` and `WRAP` bursts including AXI4 extensions of `INCR` burst up to 256 data beats.
- Attempting `FIXED` bursts does not hang the AXI4 interface, but a `FIXED` burst does not have a logical meaning for a memory controller. For simplicity, `FIXED` burst commands result in an `INCR` command. No errors are flagged.
- Supports burst size down to 1 byte wide burst. Burst sizes below the native data width of the MCB port controller datapath is called a subsize burst or "narrow" transfer. Subsize burst is supported, but the AXI protocol defines a subsize burst to have data rotate through the correct byte lanes. Narrow burst support is conditional. If the system has no masters that produce narrow bursts, then significant logic can be reduced by removing support for the narrow bursts. This is controlled by the `C_S`*<Port_Num>*`_AXI_SUPPORTS_NARROW_BURST` parameter.
- The AXI Spartan-6 FPGA DDRx Memory Controller can assume that bursts do not cross a 4 KB address boundary as defined in the AXI4 specification. However, a burst that crosses a 4 KB boundary does not hang the interface, but it can cause that transaction to have undefined behavior on memory contents.

Cache Bits

These cache bit rules apply:

- The AXI Spartan-6 FPGA DDRx Memory Controller does not implement bridging, speculative pre-fetching, or L2 caching functions so it can ignore all CACHE bits and treat them as `00000`.
- The AXI Spartan-6 FPGA DDRx Memory Controller attempts to return B Responses as soon as possible without violating AXI ordering rules to reduce latency to master waiting for B Responses.
- Because the AXI Spartan-6 FPGA DDRx Memory Controller is connected to a multi-ported hard memory controller, it must not issue a B Response until the Write has completed to memory. The B response must guarantee that another Write or Read on

another MCB port that accesses the same memory location could not complete ahead of the current Write transaction. The parameter C_S<Port_Num>_AXI_STRICT_COHERENCY can be set to 0 to relax write coherency checking so that the B Response is returned earlier when the transaction is known to have completed relative to that port instead of being delayed to ensure the write completes across all ports.

Protection Bits

The AXI Spartan-6 FPGA DDRx Memory Controller ignores the AXI `PROT` bits and assume all transactions are normal, non-secure accesses.

Exclusive Access

This IP does not currently support exclusive access.

Response Signaling

The AXI Spartan-6 FPGA DDRx Memory Controller always generates an OKAY response.

IDs, Threads, and Reordering

The MCB interface is strictly linear; therefore no reordering or threads is implemented in the bridge. Transactions are returned in the exact order they are received.

Read/Write Acceptance Depth

The read acceptance depth is five outstanding transactions. The Write acceptance depth is four outstanding transactions.

Read/Write Arbitration

AXI has separate Read and Write channels. An external memory has only a single address bus. Therefore the AXI Spartan-6 FPGA DDRx Memory Controller must arbitrate between coincident Read and Write requests to determine which one to execute to memory. The arbitration algorithm for Read and Write requests is Round-Robin.

Endianess

The AXI Spartan-6 FPGA DDRx Memory Controller is little-endian only.

Region Bits

The AXI Spartan-6 FPGA DDRx Memory Controller does not have to make use of REGION bits and can ignore this signal.

Low Power Interface

The AXI Spartan-6 FPGA DDRx Memory Controller does not support low power interface.

Limitations

The AXI Spartan-6 FPGA DDRx Memory Controller does not support QoS.