MIG Example Design with Traffic Generator (CORE Generator Tool Native Interface Only)

This section explains how to simulate and implement the MIG generated example design. This design includes a traffic generator for demonstrating and testing the MCB based memory interface. The bitstream created from implementation of the example design can be targeted to a Spartan-6 FPGA SP601 or SP605 hardware evaluation board to demonstrate DDR2 or DDR3 interfaces, respectively.

The example design includes these modules as shown in Figure 1-38:

- Spartan-6 FPGA MIG Wrapper: top-level wrapper file produced by the MIG tool, containing an MCB and other FPGA resources necessary to create the desired memory interface.
- TB_top: test bench stimulus module with the Init Memory Control block and the Traffic Pattern Generator.
- Clock Infrastructure: Spartan-6 FPGA PLL and clock network resources required for the memory design.

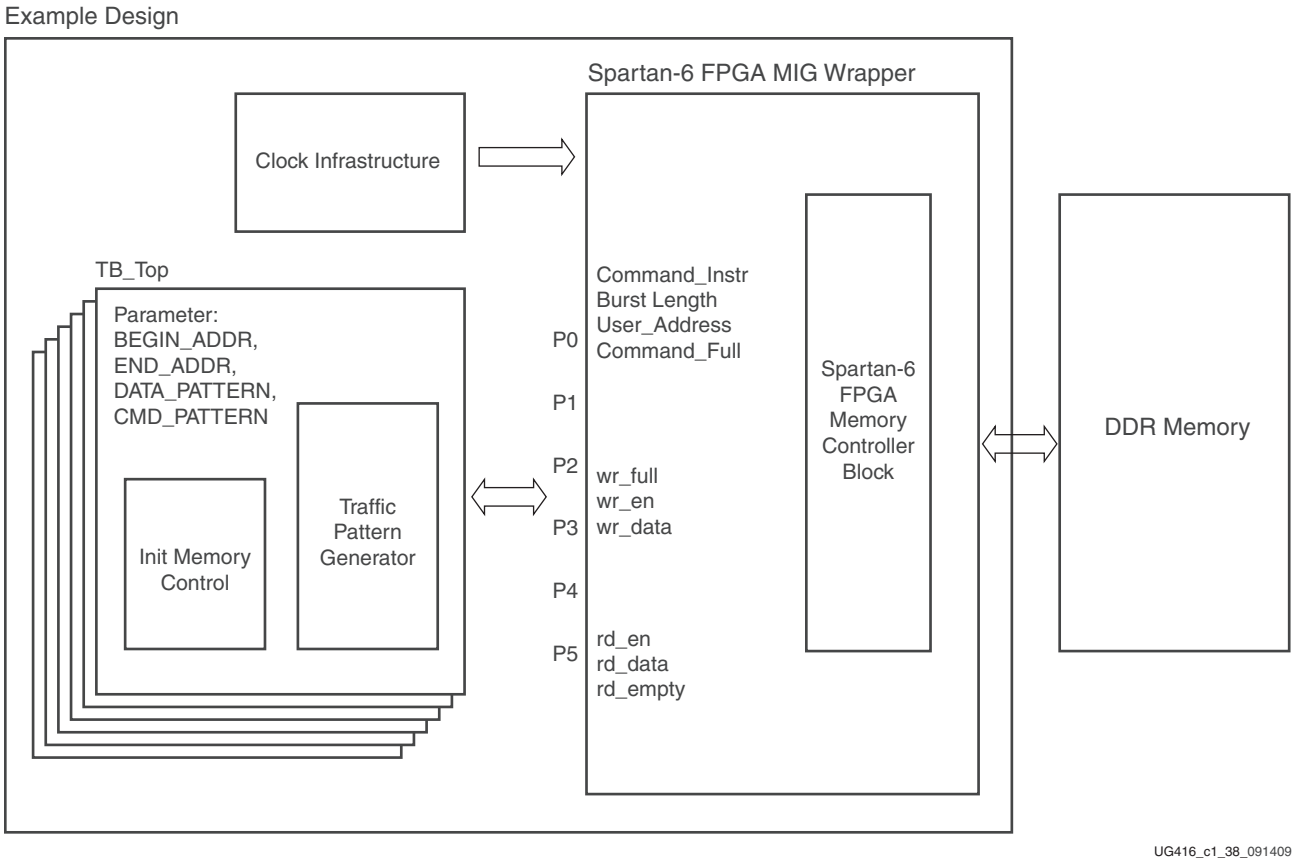

Figure 1-38: **MIG Example Design with Synthesizable Traffic Generator**

Traffic Generator Operation

The Traffic Generator module contained within the synthesizable test bench can be parameterized to create various stimulus patterns for the memory design. It can produce

repetitive test patterns for verifying design integrity as well as pseudo-random data streams that model "real world" traffic.

The MIG tool creates a separate traffic generator for each enabled port of the User Interface. Each traffic generator can create traffic patterns for the entire address space of its associated port. A default address space for each port is assigned by the MIG tool using the BEGIN_ADDRESS and END_ADDRESS parameters found in the top-level test bench file (`tb_top.v`). See Modifying the Example Design, page 48 for information on using these parameters to change the port address space.

The test bench first initializes the entire address space of the port with the requested data pattern (data pattern options are discussed in the following subsections). The Init Memory Control block directs the traffic generator to step sequentially through all addresses in the port address space, writing the appropriate data value to each location in the memory device as determined by the selected data pattern. By default, the test bench uses the address as the Data pattern.

When the memory has been initialized, the traffic generator begins stimulating the User Interface ports to create traffic to and from the memory device. By default, the traffic generator sends pseudo-randomized commands to the port, meaning that the instruction sequences (R/W, R, W, etc.), addresses, and burst lengths are determined by pseudo-random bitstream (PRBS) generator logic in the test bench. As with the address space and data pattern, the default PRBS command pattern can be changed as described in Modifying the Example Design, page 48.

The read data returning from the memory device is accessed by the traffic generator through the User Interface read data port and compared against internally generated "expect" data. If an error is detected (for example, there is a mismatch between read data and expect data), an error signal is asserted and the readback address, readback data, and expect data are latched into the error_status outputs.

Each stimulus data pattern is described in the following subsections.

Address as Data Pattern (Default)

This pattern writes each memory location with its own address, a simple test for finding address bus related issues (see Figure 1-39).

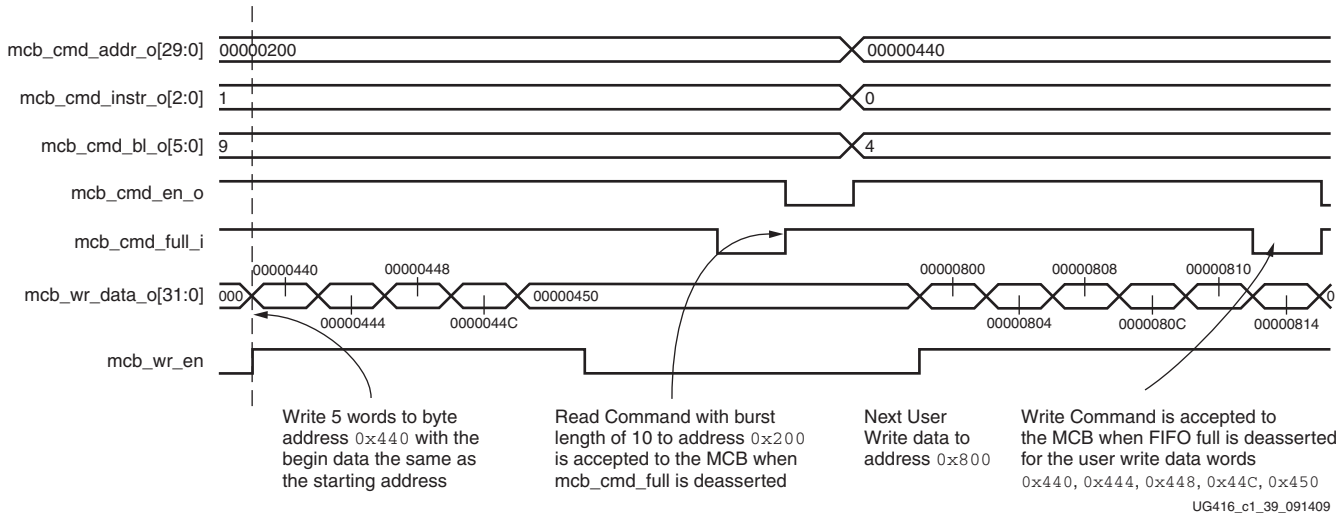

Figure 1-39: **Address as Data Pattern on DQ Bus**

Hammer Data Pattern

This pattern stresses the memory interface with simultaneous switching output (SSO) noise (see Figure 1-40). When multiple output drivers switch simultaneously, they can cause a voltage drop or ground bounce on the power planes of the PCB or inside the device package.

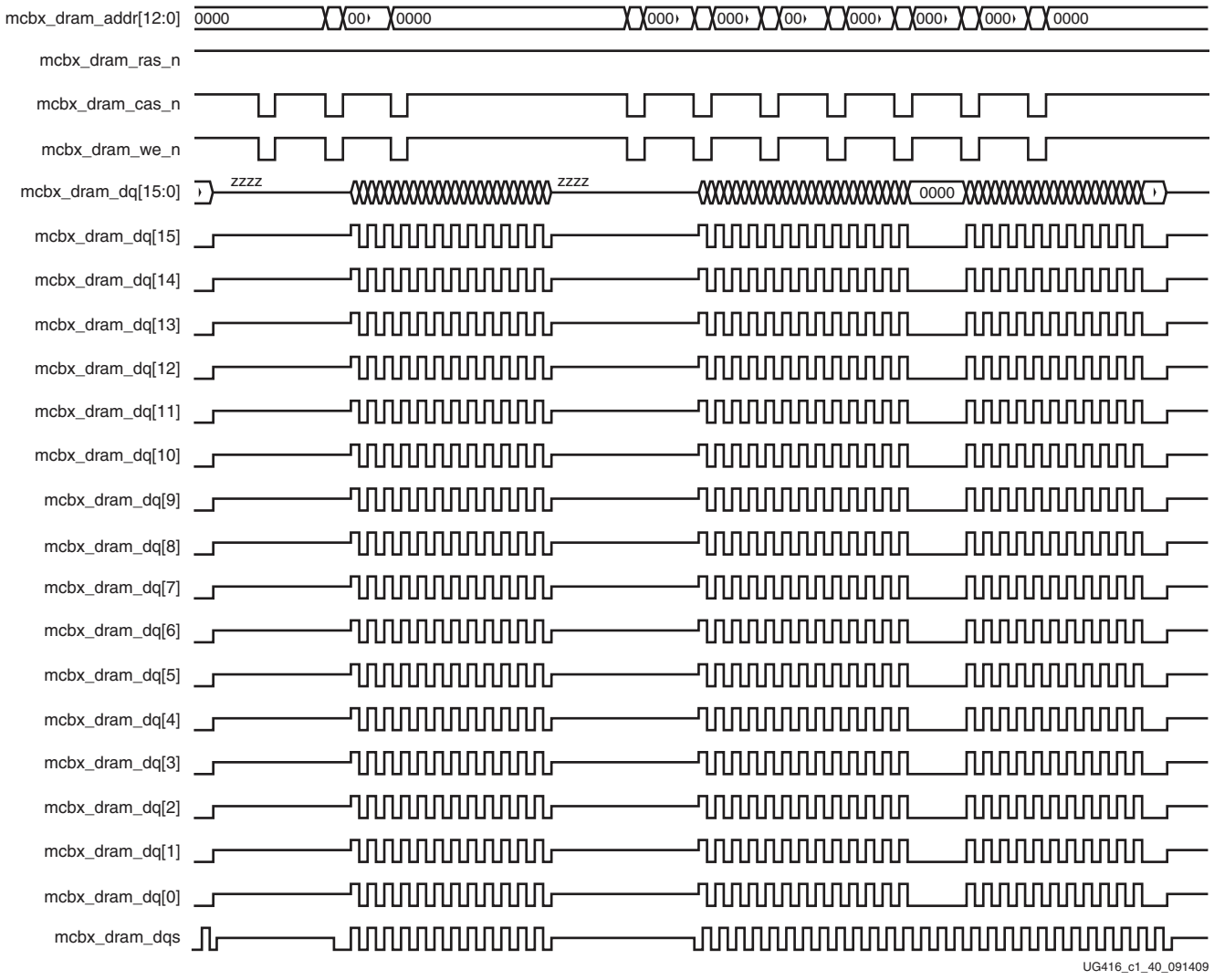

Figure 1-40: **Hammer Data Pattern on DQ Bus**

Neighbor Data Pattern

This pattern is similar to the Hammer pattern with the exception that one DQ pin remains Low on any given cycle (see Figure 1-41). This pattern can be used to measure the degree of noise coupling on a static I/O pin due to SSO noise created by other pins.

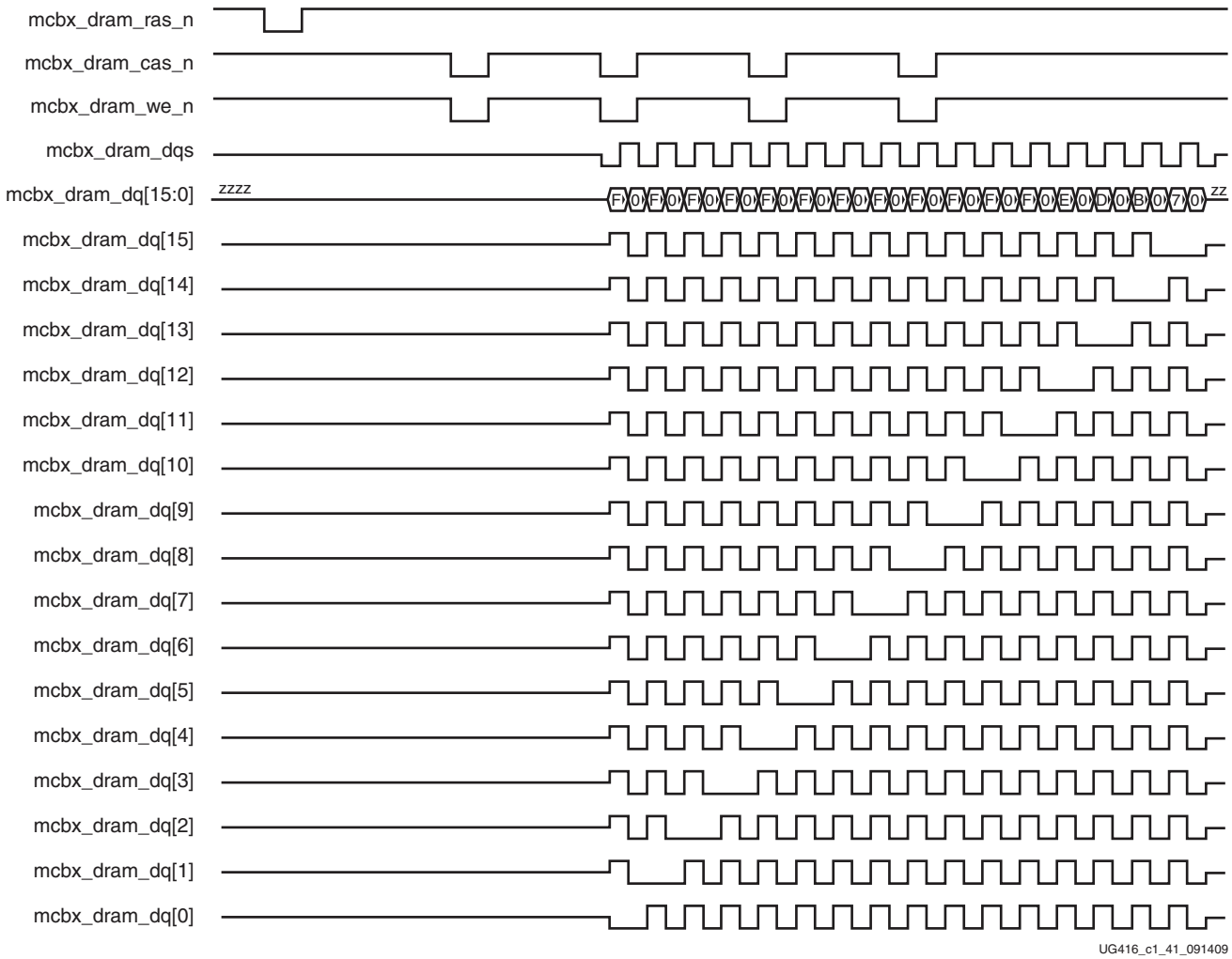

Figure 1-41: **Neighbor Data Pattern on DQ Bus**

Walking 1s and Walking 0s Data Pattern

The Walking 1s and Walking 0s patterns (see Figure 1-42 and Figure 1-43, respectively) ensure that each memory bit location can be set to both 1 and 0, independently from other bits. The DQ bus connectivity on the PCB can also be verified with these tests.

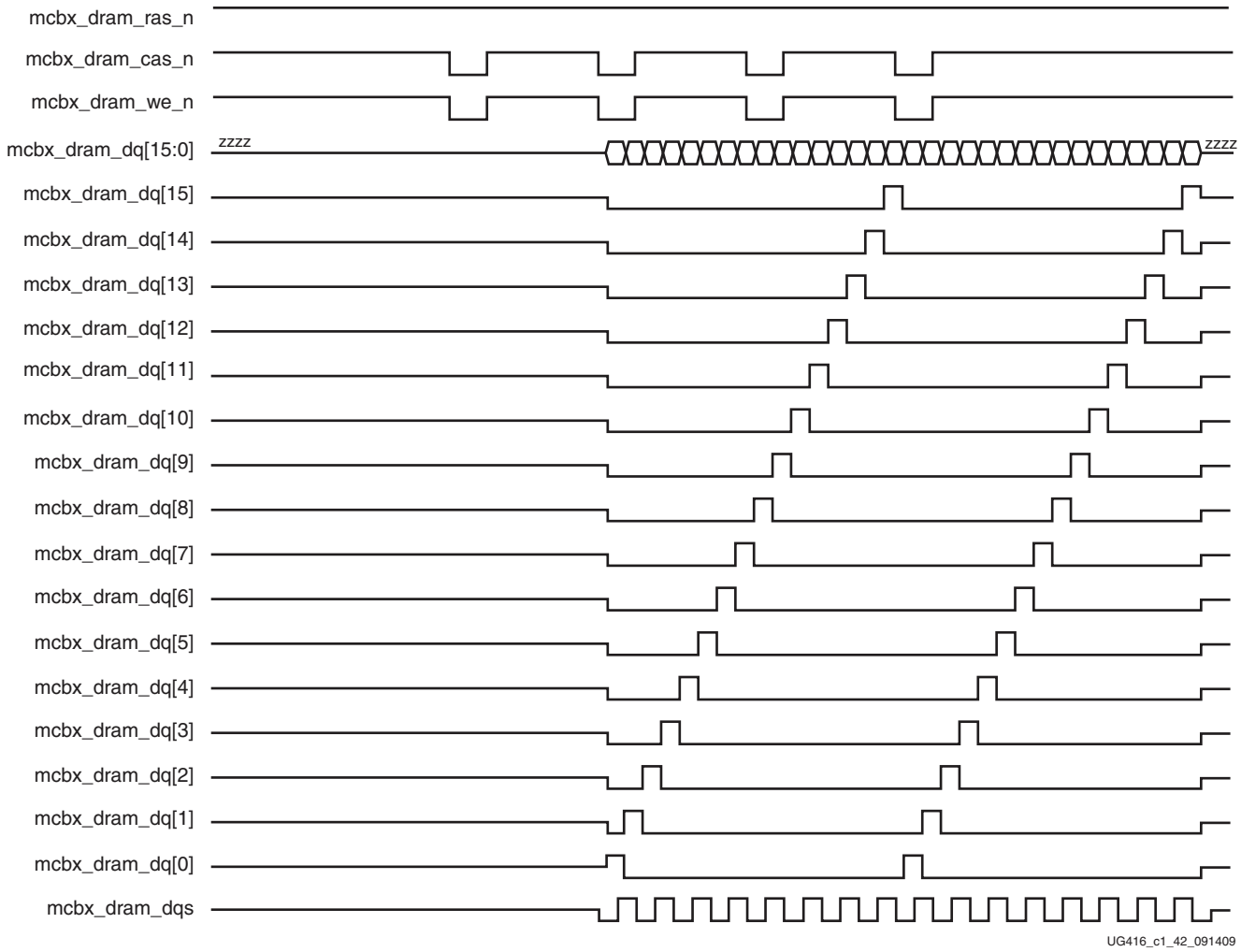

Figure 1-42: **Walking 1s Data Pattern on DQ Bus**

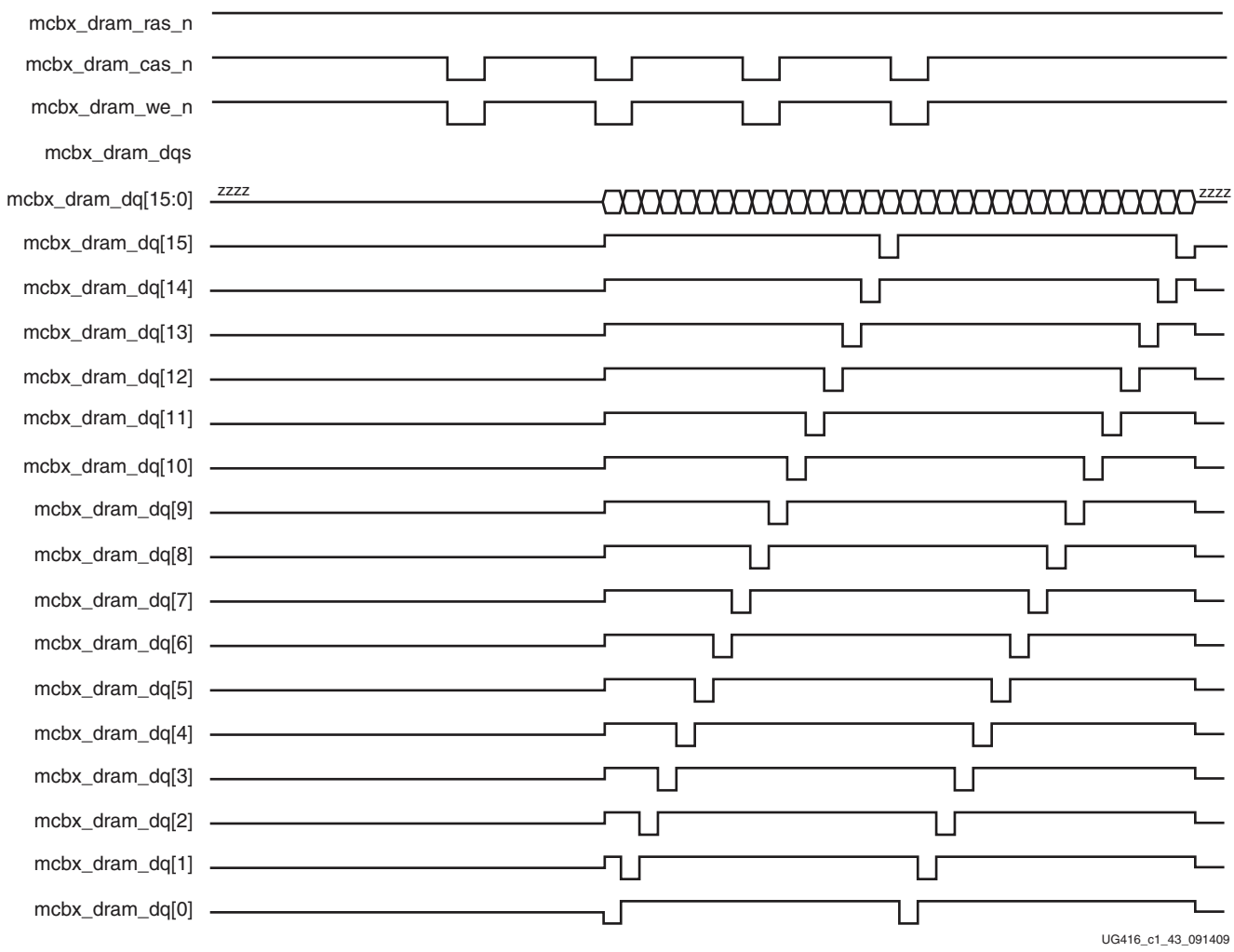

Figure 1-43: **Walking 0s Data Pattern on DQ Bus**

PRBS Data Pattern

This pattern creates PRBS data. The starting address of each data burst is used as a seed to a 32-bit LFSR circuit to generate bursts with randomized data, approximating a "real world" application test.

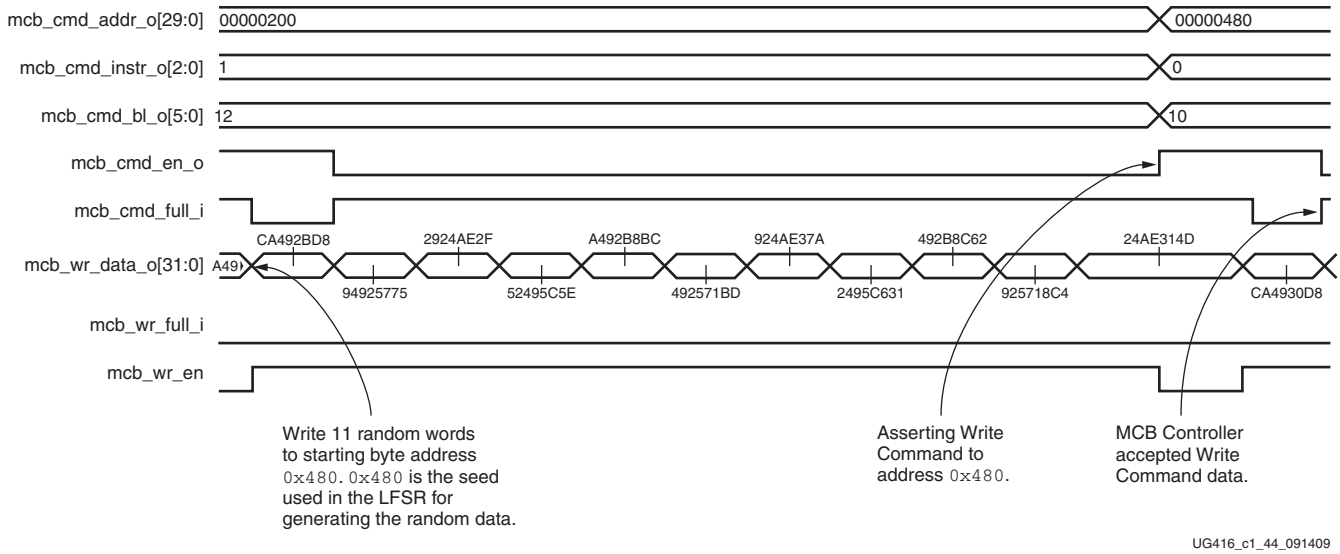

Figure 1-44: **PRBS Data Pattern on DQ Bus**

Setting Up for Simulation

In simulation, the user ports in the traffic generator are assigned with a small address range to avoid memory overflow if the system has limited physical memory installed. For hardware testing, the user can manually modify the HWTESTING parameter in `example_top` for a larger address space range.

See the "Simulation" section in *Spartan-6 FPGA Memory Controller User Guide* [Ref 1] for more details on simulating designs with the MCB.

Functional Simulation

To simulate the MIG example design or the MIG user design, the Xilinx® UNISIM library must be compiled and mapped to the simulator. Currently, MIG generated designs are supported only for Xilinx ISim and ModelSim version 6.4b or above. However, the encrypted model of the Spartan-6 FPGA MCB is provided for ISim, ModelSim, and Cadence Incisive Enterprise Simulator (IES). EDK generated designs using the MCB are supported on all three of these simulators.

The Traffic Generator test bench provided with the example design allows pre-implementation functional simulations to be performed on the generated memory interface solution.

Memory Devices Supported for Functional Simulation

The MIG tool supports Micron DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM, and LPDDR memory devices. It also supports Elpida DDR2 SDRAM memory devices for simulation. ModelSim and ISim are the simulation tools supported. ModelSim supports all of the listed memory devices, while ISim supports only the Micron devices.

To run the simulation:

1. Go to this directory:

   `<project_dir>/<component_name>/example_design/sim/functional`

2. Run the script command that corresponds to the chosen simulation tool and operating system:
   - Windows
     - For ModelSim, type at the prompt: **`sim.do`**
     - For ISim, type at the prompt: **`isim`**
   - Linux
     - For ModelSim, type at the prompt: **`source sim.do`**
     - For ISim, type at the prompt: **`source isim.do`**

Implementing the Example Design

The MIG tool automatically generates the `ise_flow.bat` script file found in the `par` folder of the example design. This script runs the design through the synthesis, translate, map, and par operations. Refer to this file to see all recommended build options for the design.

Modifying the Example Design

The test bench in the MIG generated example design can be modified to implement different data and command patterns. This section defines the test bench parameters and signal names that should be understood when making changes to the example design.

Top-Level Parameters

The top-level test bench file (`tb_top.v`) contains several parameters that can be modified to change the behavior of the traffic generator. Table 1-10 describes these parameters and identifies any default values. In general, the data pattern and address space parameters are the most likely to be modified, because the other parameters are normally fixed characteristics of the memory and MCB configuration.

The easiest way to change the data pattern implemented by the traffic generator is to open the `example_top.v` file in the `rtl` directory and edit the local parameter for Data Mode (for example, C3_p0_DATA_MODE). The four-bit code for this parameter can be changed using the binary values defined for the data_mode_i[3:0] signals in Table 1-12, page 51.

Table 1-10: **Parameters for the TB_TOP Module**

Parameter	Parameter Description	Parameter Value
BEGIN_ADDRESS	Sets the memory start address boundary	This parameter defines the start boundary for the port address space. The least-significant bits [3:0] of this value are ignored.
DATA_PATTERN	Sets the data pattern to be generated	Valid settings for this parameter are: ADDR (Default): The address is used as a data pattern. 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. WALKING1: Walking 1s are on the DQ pins and the starting position of 1 depends on the address value. 0: Walking 0s are on the DQ pins and the starting position of 1 depends on the address value. NEIGHBOR: The Hammer pattern is on all DQ pins except one. The address determines the exception pin location. PRBS: A 32-stage LFSR generates random data and is seeded by the starting address.
DWIDTH	The MIG tool sets the default based on the User Data port width	Valid settings for this parameter are 32, 64, and 128 bits.
END_ADDRESS	Sets the memory-end address boundary	This parameter defines the end boundary for the port address space. The least-significant bits [3:0] of this value are ignored.
FAMILY	Indicates the Family type	The value of this parameter is “SPARTAN6”.
NUM_DQ_PINS	The MIG tool sets the default based on the number of data (DQ) pins for the selected memory	Valid settings for this parameter are “4”, “8”, and “16”.
PORT_MODE	The MIG tool sets the default based on the port configuration (bidirectional, W only, or R only)	Valid settings for this parameter are: BI_MODE: Generate a WRITE data pattern and monitor the READ data for comparison. WR_MODE: Generate only WRITE data patterns. No comparison logic is generated for the port. RD_MODE: Generate only READ control logic for the port.
PRBS_EADDR_MASK_POS	Sets the 32-bit AND MASK position	This parameter is used with the PRBS address generator to shift random addresses down into the port address space. The END_ADDRESS value is ANDed with the PRBS address for bit positions that have a “1” in this mask.
PRBS_SADDR_MASK_POS	Sets the 32-bit OR MASK position	This parameter is used with the PRBS address generator to shift random addresses up into the port address space. The BEGIN_ADDRESS value is ORed with the PRBS address for bit positions that have a “1” in this mask.

Traffic Generator Parameter

The CMD_PATTERN parameter can be modified within the Traffic Generator module (see Table 1-11). This parameter is not brought to the top-level test bench because it should not be modified under normal circumstances. However, certain situations might require a change to the default value, such as when address, burst length, and instruction values are provided from a block RAM (see Custom Command Sequences, page 54).

Table 1-11: **Parameter for the Traffic Generator Module**

Parameter Name	Parameter Description	Parameter Value
CMD_PATTERN	Parameter for setting command pattern circuits to be generated. For larger devices, the CMD_PATTERN can be set to "CGEN_ALL". This parameter enables all supported command pattern circuits to be generated. However, it is sometimes necessary to limit a specific command pattern because of limited resources in a smaller device.	Valid settings for this signal are: CGEN_FIXED: The address, burst length, and instruction are taken directly from the fixed_addr_i, fixed_bl_i, fixed_instr_i inputs. CGEN_SEQUENTIAL: The address is incremented sequentially, and the increment is determined by the data port size. CGEN_BRAM: The address, burst length, and instruction are taken directly from the bram_cmd_i input bus. CGEN_PRBS: A 32-stage LFSR generates pseudo-random addresses, burst lengths, and instruction sequences. The seed can be set from the 32-bit cmd_seed input. CGEN_ALL (Default): This option turns on all of the above options and allows addr_mode_i, instr_mode_i, and bl_mode_i to select the type of generation during run-time.

Traffic Generator Signal Descriptions

Table 1-12 describes all traffic generator signals. In the example design, the Init Memory Control block controls most of these signals to implement the default test flow (that is, initialize the memory with the data pattern, then start running traffic by generating pseudo-random command patterns). Any modification of the design to control these signals by other means should only be done with a thorough understanding of their behavior.