Xilinx Answer 51204
MIG 7 Series DDR2/3 – PHY Only Design

Important Note: This downloadable PDF of an Answer Record is provided to enhance its usability and readability. It is important to note that Answer Records are Web-based content that are frequently updated as new information becomes available. You are reminded to visit the Xilinx Technical Support Website and review (Xilinx Answer 51204) for the latest version of this Answer.

Introduction

The MIG 7 series DDR3/DDR2 IP LogiCORE is provided as a full memory interface design with physical layer (PHY), highly efficient memory controller, and user interface blocks. All blocks are provided as HDL source code. Generally, the full 7 series MIG DDR3/DDR3 design meets or exceeds customer memory design requirements. However, some applications may benefit from a custom controller that is designed specifically for the target access pattern. In these cases, Xilinx supports using the PHY only portion of the MIG 7 series IP to interface to the custom controller. This answer record provides the necessary information to interface a custom controller to the MIG 7 series PHY design.

Where to Start

When designing a memory interface that will use a custom controller with the MIG 7 series DDR3/DDR2 PHY, the best starting place is to review the MIG 7 series FPGAs Memory Interface Solutions User Guide to gain a general understanding of the MIG 7 series DDR3/DDR2 Design. It is especially important to review the design of the PHY. The PHY is a complex block that controls the signal timing and sequencing between the 7 series hard blocks and the external DDR2 or DDR3 device. Additionally, it contains the state logic to perform the SDRAM initialization after power-up and the calibration logic to perform timing training of the read and write data paths to account for system static and dynamic delays. Before designing the interface between the custom controller and the PHY, a general understanding of the PHY architecture is necessary. The most important aspects to understand are what the IN_FIFOs, OUT_FIFOs, and PHY Control Block hard blocks are, how they are used in the PHY design, and what a PHY Control Word is. The User Guide contains this information and can be found here:

http://www.xilinx.com/support/documentation/ipinterconnect_mig-7series.htm

This answer record is intended to supplement the information provided within the User Guide and in no way replace the existing content.

MIG PHY Only Basic Signals

The MIG PHY is part of the overall MIG Controller. Figure-1 shows a block diagram of the MIG Controller and its connection to the MIG PHY. As you can see, the PHY Interface is composed of the Initialization and Calibration Logic besides the actually PHY itself.

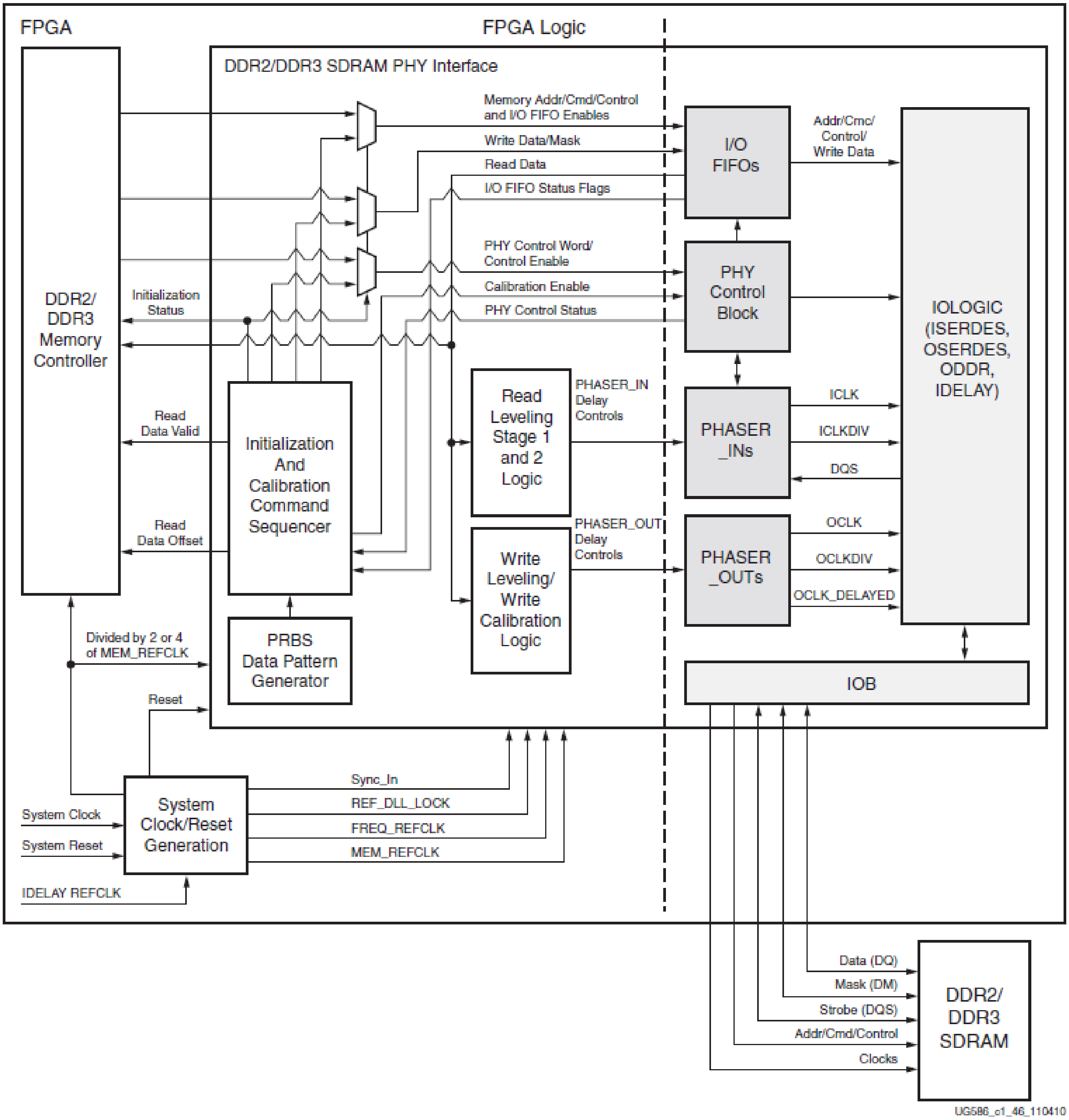

FPGA
FPGA Logic
DDR2/DDR3 SDRAM PHY Interface
DDR2/ DDR3 Memory Controller
Memory Addr/Cmd/Control and I/O FIFO Enables
Write Data/Mask
Read Data
I/O FIFO Status Flags
PHY Control Word/ Control Enable
Calibration Enable
PHY Control Status
Initialization Status
Read Data Valid
Read Data Offset
Initialization And Calibration Command Sequencer
PRBS Data Pattern Generator
Read Leveling Stage 1 and 2 Logic
Write Leveling/ Write Calibration Logic
PHASER_IN Delay Controls
PHASER_OUT Delay Controls
I/O FIFOs
PHY Control Block
PHASER _INs
PHASER _OUTs
Addr/Cmc/ Control/ Write Data
ICLK
ICLKDIV
DQS
OCLK
OCLKDIV
OCLK_DELAYED
IOLOGIC (ISERDES, OSERDES, ODDR, IDELAY)
IOB
Divided by 2 or 4 of MEM_REFCLK
Reset
System Clock
System Reset
System Clock/Reset Generation
IDELAY REFCLK
Sync_In
REF_DLL_LOCK
FREQ_REFCLK
MEM_REFCLK
Data (DQ)
Mask (DM)
Strobe (DQS)
Addr/Cmd/Control
Clocks
DDR2/ DDR3 SDRAM
UG586_c1_46_110410

Figure 1 – MIG Controller and PHY Interface Block Diagram

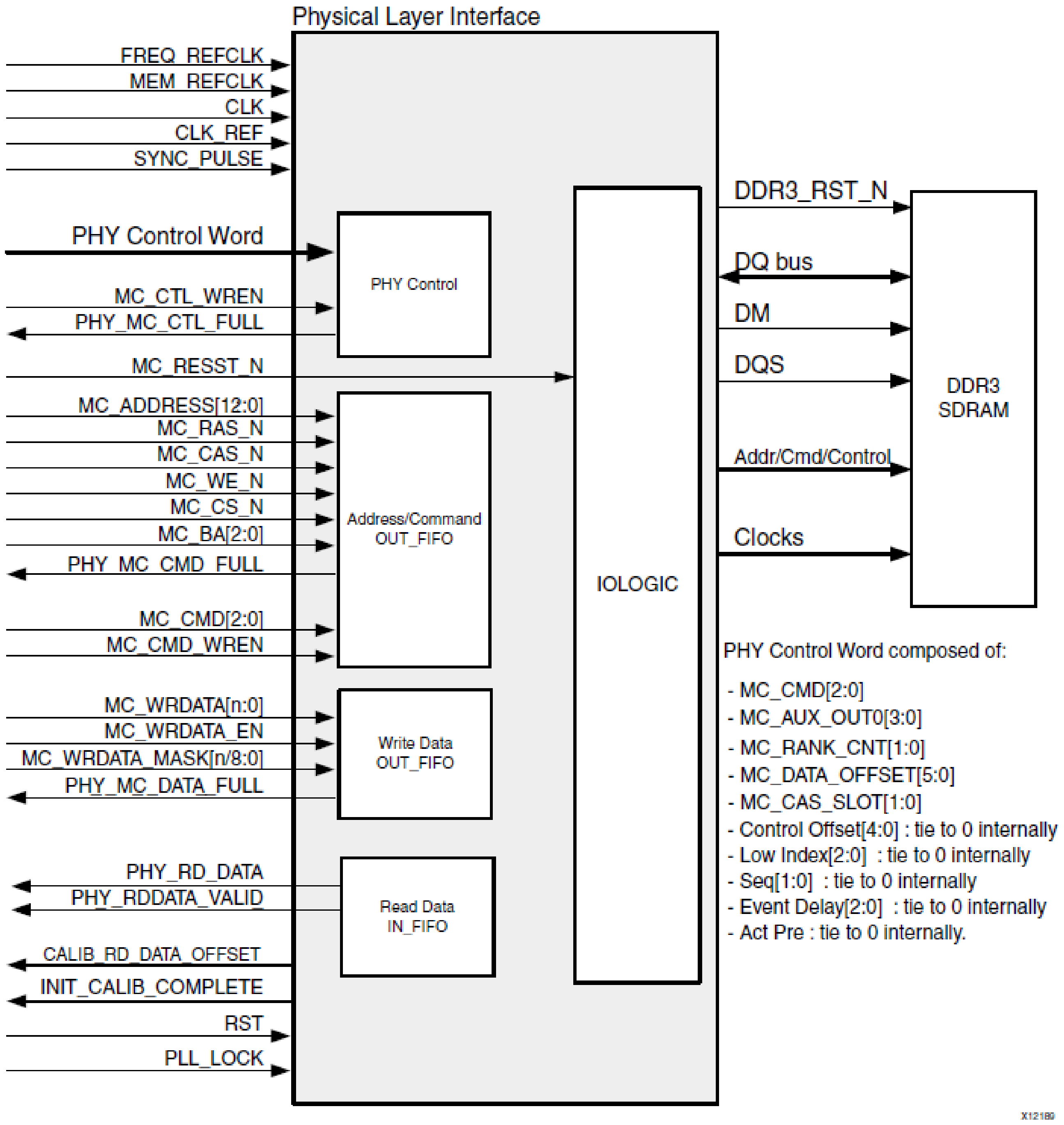


Figure 2 – PHY Block Diagram

The PHY Block itself consists of the PHY Control, Address and Command OUT_FIFO, Write Data OUT_FIFO, Read Data IN_FIFO, and IOLOGIC. Figure 2 is a simple representation of these blocks as shown above. The PHY Control will control the PHY block as a whole in conjunction with other blocks. The Address & Command OUT_FIFO is used to store address/command. The Write Data OUT_FIFO is used for write data storage. The Read Data IN_FIFO is used for read data storage. Lastly, IOLOGIC is used to bring the internal signals out of the FPGA to the SDRAM on board.

The IN/OUT FIFOs are used for serialization/deserialization and is used to cross clock domains. The OUT_FIFO will receive 8 bit at ¼ the DDR2/3 frequency and converts this to 4 bits and transfers this to the OSERDES at ½ of the DDR2/3 frequency, which is the ICLKDIV domain. While the IN_FIFO takes 4 bits from the DQ ISERDES and serializes this data by writing two packets into a single memory array cell. This successfully brings it from the ½ rate domain back into the ¼ rate domain to be processed by the controller.

Signals of Interest

Much of the signals below are explained in the User Guide, but for the sake of this article we will explain them again.

Note: All debug (dbg_*) signals will not be covered in this guide and can be found in the "7 Series MIG DDR3/DDR2 – Hardware Debug Guide"

http://www.xilinx.com/support/answers/43879.htm

Clocking

- Explained in the User Guide more thoroughly and outside the scope of this answer record. We will only cover what is necessary for this article.
- Iserdes_clkdiv - Phaser_IN output clock used to write data into the Data (DQ) IN_FIFO(s). There is one Iserdes_clkdiv per data byte group. This is called ICLKDIV in the User Guide and Iserdes_clkdiv is the name used in the RTL.
- clk - MMCM output clock routed on a global clock network (BUFG) to read data out of the data (DQ) IN_FIFO(s).
- clk_ref – This is a reference frequency for the IDELAY control. This is a 200 MHz input. The clk_ref input can be generated internally or connected to an external source.
- mem_refclk – This is the DDR2 or DDR3 frequency clock.
- freq_refclk – This signal is the same frequency as the mem_refclk between 400 MHz and 933 MHz, and ½ or ¼ of mem_refclk for frequencies below 400 MHz.

Memory Controller to Calibration

- pll_lock – The LOCKED output of the PLL instantiated in the infrastructure module.
- sync_pulse – This is the synchronization pulse output by the PLL.
- rst – The rstdiv0 output from the infrastructure module synchronized to the PHY_Clk domain.
- mc_aux_out0 – This auxiliary outputs field in the PHY control word used to control ODT and CKE assertions.
- mc_aux_out1 - This auxiliary outputs field in the PHY control word used to control ODT and CKE assertions. mc_rank_cnt – This is the rank accessed by the command sequence in the PHY control word.
- phy_rddata_valid – This signal is asserted when valid read data is available.
- phy_rd_data – This is the read data from the dedicated PHY. It is 8x the memory DQ width for a 4:1 clock ratio. It will be 4x for a 2:1 clock ratio.

PHY Control Block

- phy_ctl_wd – Defines a set of actions that the PHY control block does to initiate the execution of a DDR2 or DDR3 SDRAM command.
- mc_ctl_wren – Write enable for the Control FIFO, generally always tied high so the PHY can receive data.
- phy_mc_ctl_full – Pre_Fifo is full and can no longer accept commands. To prevent overflow, mc_ctl_wren should be set to '0'.

IOLOGIC

- DDR PHY signals (ddr_address, ddr_ba, ddr_cs_n, ddr_ras_n, ddr_cas_n, ddr_we_n, ddr_ck, ddr_ck_n, ddr_cke, ddr_dm, ddr_odt, ddr_parity, ddr_reset_n, ddr_dq, ddr_dqs, ddr_dqs_n) – external connection to the SDRAM

Address/Command Out_FIFO

- Memory Controller DDR signals (mc_address, mc_bank, mc_cs_n, mc_ras_n, mc_cas_n, mc_we_n, mc_odt, mc_cke) – used to drive the PHY, connection from the memory controller into the PHY. Note: There are four bits for each of the command signals in 4:1 designs and 2 bits in 2:1 designs. The signals should be asserted on the bit that corresponds to the slot number used for the command. For example, a read command on slot 1, would assert the JEDEC read command on bit 1 of all of the command signals.
- phy_mc_cmd_full – PHY is full and can no longer accept commands, recommend that mc_cmd_wren is set to '0' to prevent overflow.
- mc_cmd_wren – write enable for the Address/Command Out_FIFOs.

- mc_cmd – This signal is used for PHY_Ctl_Wd configuration:

  - 0x04: Non-data command (No column command in the sequence of commands)
  - 0x01: Write command
  - 0x03: Read command

Write Data -Out_FIFO

- mc_wrdata - This is the write data to the dedicated PHY. It is 8x the memory DQ width for a 4:1 clock ratio.
- mc_wrdata_en - This signal is the write enable input to the DQ OUT_FIFO.
- mc_wrdata_mask - This is the write data mask to the dedicated PHY. It is 8x the memory DM width for a 4:1 clock ratio.
- phy_mc_data_full – Not used in the MIG core.
- mc_data_offset – This is the write data offset value.

Read Data - In_FIFO

- phy_rd_data -This is the read data from the dedicated PHY. It is 8x the memory DQ width for a 4:1 clock ratio.
- phy_rddata_valid – This signal is asserted when valid read data is available.

Other

- Idle – See the IDLE section below
- Mc_cas_slot – See the CAS Slot Number Usage section below.
- Device_temp –this the raw value as read from the DRP port as defined by the XADC specification. Please view UG 772 for more information on this.
- Init_wrcal_complete - write calibration is complete.
- Ref_dll_lock – Phaser_ref block DLL lock signal
- Rst_phaser_ref – Reset to the Phaser_ref block
- Slot_0_present – Signal to show a DIMM is present in slot 0.
- Slot_1_present - Signal to show a DIMM is present in slot 1. Note: The MIG PHY only supports a single DIMM.
- Mc_data_offset_1/2 – Value passed from calibration that lets the controller know what the data offset will be when the MC is in control.
- Calib_rd_data_offset_0/1/2 – Calculated read data offset value during calibration with respect to command 0 in the sequence of the four commands. If the Phy Only Design spans only one I/O bank, then only calib_rd_data_offset_0 is used. If it spans two I/O banks, then both calib_rd_data_offset 0 and 1 are used. If three, then all three are used. When un-used, the value is set to 0, otherwise they will be set during calibration.

MIG Parameters/Signals that are ignored by the PHY Only Design

- Error – Error signal of the traffic generator
- Rst_tg_mc – reset to the traffic generator

Calibration

Calibration is handled by the PHY. It is required for the custom memory controller to wait until the init_calib_complete signal goes high before beginning any reads or writes to the memory. More information on this can be found in the User Guide under the "Calibration and Initialization Stages" section.

PHY Control Word

The PHY Control Word is composed of these signals:

- MC_CMD[2:0]
- MC_AUX_OUT[3:0]
- MC_RANK_CNT[1:0]
- MC_DATA_OFFSET[5:0]
- MC_CAS_SLOT[1:0]
- Control_Offset[4:0] : tied to 0 internally
- Low_Index[2:0] : tied to 0 internally
- Seq[1:0]
- Act_Pre : tied to 0 internally
- Event_Delay[2:0] : tied to 0 internally

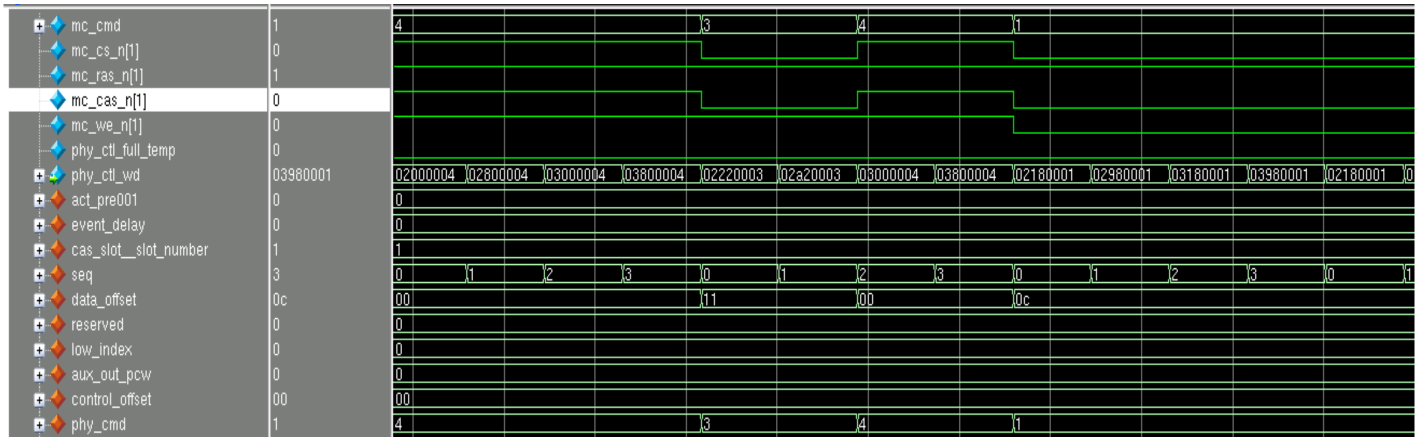

mc_cmd 1 4 3 4 1
mc_cs_n[1] 0
mc_ras_n[1] 1
mc_cas_n[1] 0
mc_we_n[1] 0
phy_ctl_full_temp 0
phy_ctl_wd 03980001 02000004 02800004 03000004 03800004 02220003 02a20003 03000004 03800004 02180001 02980001 03180001 03980001 02180001 0
act_pre001 0 0
event_delay 0 0
cas_slot__slot_number 1 1
seq 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1
data_offset 0c 00 11 00 0c
reserved 0 0
low_index 0 0
aux_out_pcw 0 0
control_offset 00 00
phy_cmd 1 4 3 4 1

Figure-3 – The PHY Control Word broken up into its individual buses.

The command signals will control the PHY block and as well as feed the phy_ctl_wd the necessary information. Above is a simple representation of the command signals driving the phy_cmd as they are connected, Figure-3. As the CAS slot number is set to 1, we have pulled down only the command signals for bus1 and the ras/cas/we signals shows non-data, read, non-data, and finally a few write commands afterwards. The figure above give a clear understand of what occurs with the phy_ctl_wd. Each phy clock cycle shows that it goes through one sequence only based on the CAS slot number.

CAS Slot Number Usage

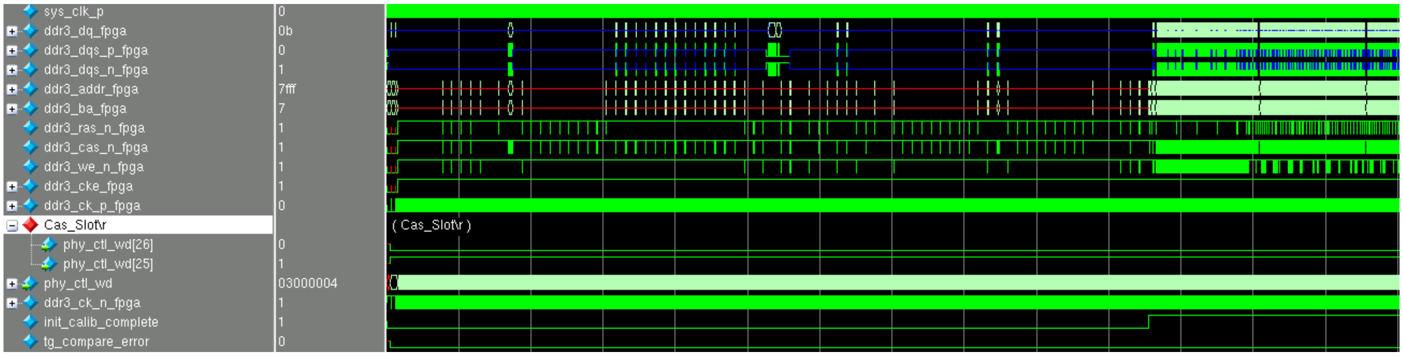
One PHY Control Word is sent every slow clock cycle (BUFG clock domain) to account for 4 CK clock cycles (in 4:1 mode). There are 4 slots (slot 0/1/2/3) that represent each of the 4 CK cycles. Simultaneously 4 clock cycles worth of address, command and possibly data (for BL8 writes) are loaded into the OUT_FIFOs. A portion of the PHY Control Word is the Data Offset. The PHY Control Block handles the timing of the PHY. It uses the Data Offset on data commands to enable the OUT_FIFOs to transfer the write data to the I/O during writes and to enable the IN_FIFOs during a read. There is only 1 data offset within a PHY Control Word and therefore within 4 CK clock cycles (slot 0/1/2/3). Additionally the data OUT_FIFOs are loaded with 4 CK cycles of data (which accounts for a single BL8 write). The CAS slot number is simply telling the PHY Control block on which of the 4 CK clock cycles the OUT_FIFO/IN_FIFO information should be transferred. You cannot use more than 1 slot in a PHY Control Word as there is only 1 data offset and there is only information stored within the FIFOs for 4 CK clock cycles.

Slot 0 and slot 2 can be used for data commands but NOT within the same PHY Control Word. Slot 0 would be used on in 1 PHY Control Word with the corresponding Data Offset. Slot 2 would be used in a different PHY Control Word with its corresponding Data Offset. It is simplest to avoid this.

When CWL is odd, RAS commands are issued on slot 0 and CAS commands are issued on slot 1. There is a natural 1 cycle separation between RAS and CAS in the DRAM clock domain so the RCD can expire in the same FPGA cycle as the RAS command. In 2:1 mode, there are only 2 slots so direct translation correctly places the CAS with respect to the corresponding RAS. In the Figure-4 below, is a simulation of the MIG 2:1 Example Design, you'll see that the CAS Slot is essentially tied to 01 for this particular simulation.

The MIG design either uses slots 0 or slot 1 for reads and writes depending on the CWL.

There are certain advantages to different CAS slot number usages and this will depend on your target application. For example you may have an application that constantly switches banks, so constant precharges are needed. If the precharge to precharge time is 4 CK cycles, you can have non-data transactions on slot 0 and slot 3. This will increase the efficiency of the overall system for this particular use case. Selecting which CAS slots to use for data and non-data transactions will depend on the pattern type a designer plans to use and will vary greatly from each design.



```
sys_clk_p              0
ddr3_dq_fpga           0b
ddr3_dqs_p_fpga        0
ddr3_dqs_n_fpga        1
ddr3_addr_fpga         7fff
ddr3_ba_fpga           7
ddr3_ras_n_fpga        1
ddr3_cas_n_fpga        1
ddr3_we_n_fpga         1
ddr3_cke_fpga          1
ddr3_ck_p_fpga         0
Cas_Slot\r
  phy_ctl_wd[26]       0
  phy_ctl_wd[25]       1
phy_ctl_wd             03000004
ddr3_ck_n_fpga         1
init_calib_complete    1
tg_compare_error       0
```

Figure-4 – CAS Slot Value

Sequence

The sequence number is used in combination with the Sync_In control signal from the PLL to keep two or more PHY control blocks executing the commands read from their respective control queues in sync. Commands with a given seq value must be executed by the command parser within the PHY control block during the specific phase indicated by the Seq field. The Seq count must be incremented with every command sequence of four. The Seq field is used to synchronize PHY control blocks across multiple I/O banks.

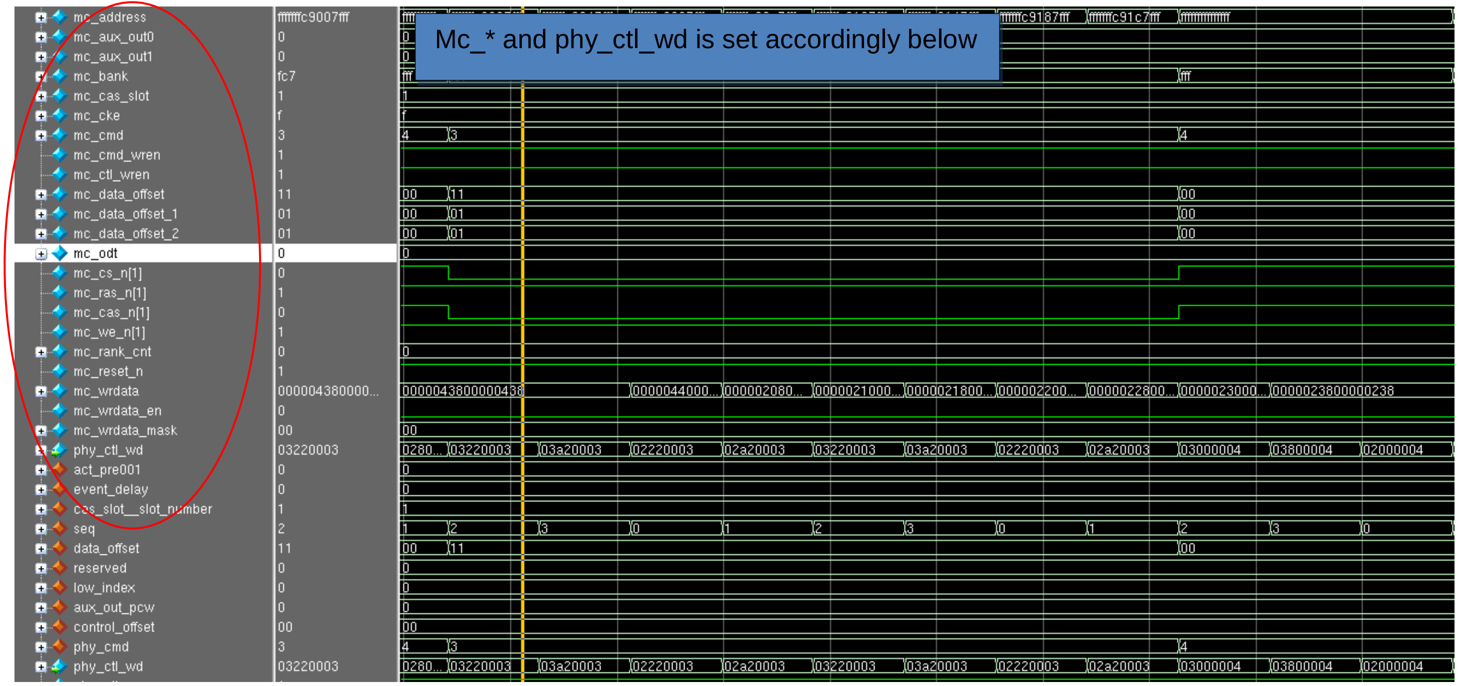
Usage is very simple; the sequence should loop from 0 to 3 continuously. There is no correlation to when reads or writes are executed. The Seq is purely used to ensure the PHY works properly across multiple I/O banks.

Reads

A read begins with sending the phy_ctl_wd; the lower three bits that control the phy_cmd show a value of 0x03. Besides sending the phy_ctl_wd, the mc_address, mc_bank, mc_cs_n, mc_ras_n, mc_cas_n, mc_we_n, mc_odt, and mc_cke signals along with it. This will place the proper command and the correct address for the SDRAM to execute this instruction. Remember to assert the read command on the command bits associated with the slot used. For example, a read on slot 1 would correlate to the JEDEC read command being sent on bit 1 of all command signals (i.e., mc_ras_n[1], mc_cas_n[1], mc_we_n[1]).

For example:
Phy_ctl_wd[2:0] – “011”
Mc_address – set to the target address
Mc_bank – set to the target bank
Mc_cs_n – ‘0’
Mc_ras_n – ‘1’
Mc_cas_n – ‘0’
Mc_we_n - ‘1’
Mc_odt – ‘0’
Mc_cke – 1’

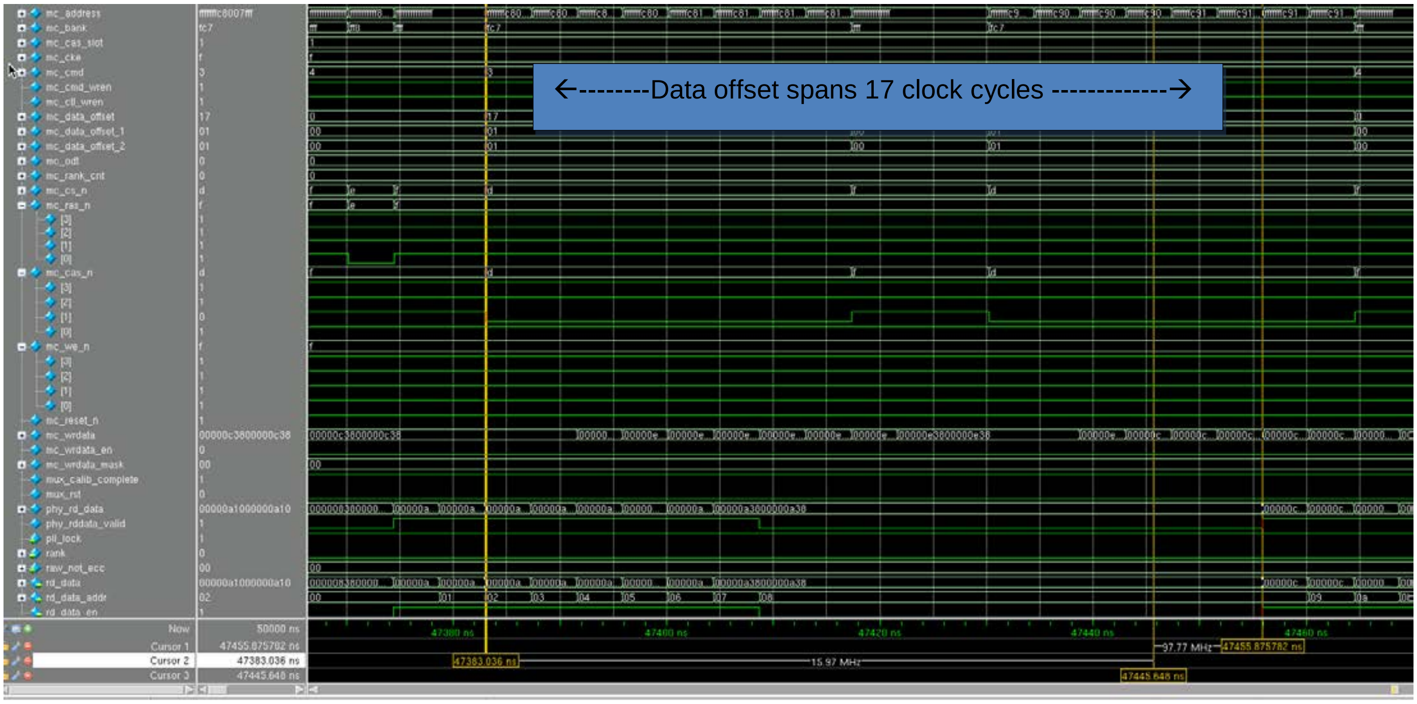
You will want to target the appropriate address and bank, and then the rest of the signals are set to do a read on the SDRAM memory. ODT will be ‘0’ for reads and will be set to ‘1’ for writes. In the Figure-5 below, you can see how the controller signals are shared with the PHY commands. Only slot 1 is shown as it is used for reads in this simulation.



Mc_* and phy_ctl_wd is set accordingly below

mc_address	ffffffc9007fff
mc_aux_out0	0
mc_aux_out1	0
mc_bank	fc7
mc_cas_slot	1
mc_cke	f
mc_cmd	3
mc_cmd_wren	1
mc_ctl_wren	1
mc_data_offset	11
mc_data_offset_1	01
mc_data_offset_2	01
mc_odt	0
mc_cs_n[1]	0
mc_ras_n[1]	1
mc_cas_n[1]	0
mc_we_n[1]	1
mc_rank_cnt	0
mc_reset_n	1
mc_wrdata	000004380000...
mc_wrdata_en	0
mc_wrdata_mask	00
phy_ctl_wd	03220003
act_pre001	0
event_delay	0
cas_slot__slot_number	1
seq	2
data_offset	11
reserved	0
low_index	0
aux_out_pcw	0
control_offset	00
phy_cmd	3
phy_ctl_wd	03220003

Figure – 5 – Controller to PHY CMD



←--------Data offset spans 17 clock cycles -------------→

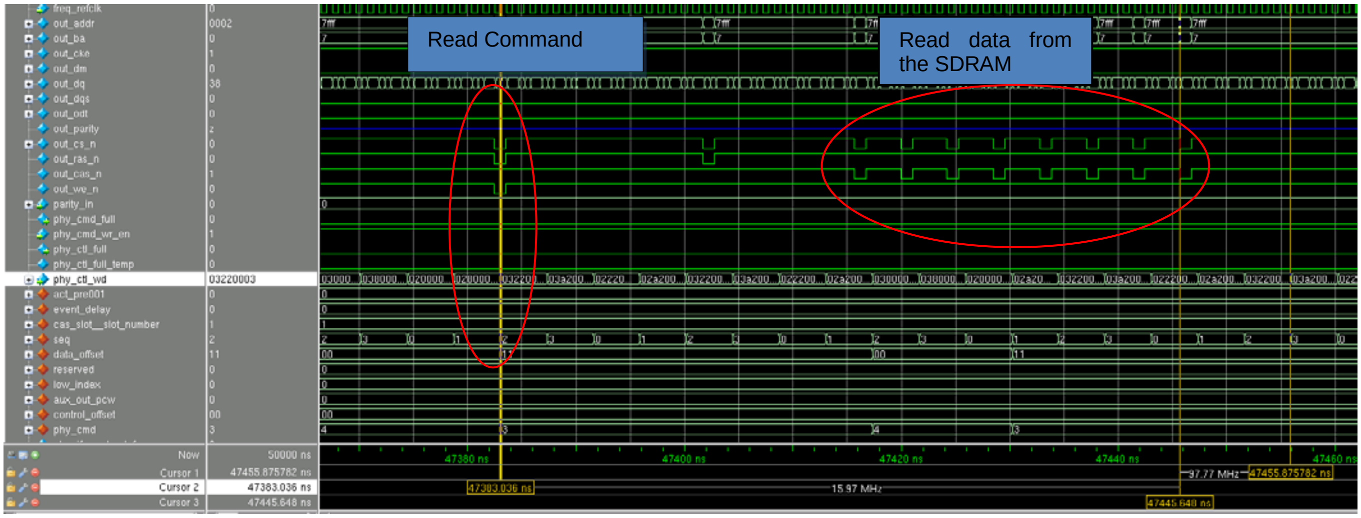
Figure- 6 – Data Offset

Viewing the address/command Out_Fifo, you can see that the data off set is 17 clock cycles.
This data offset comes from the equation of:

Read Data Offset = Calibrated PHY read data offset (calib_rd_data_offset_*) + slot number

In this case, 16 + 1 = 17, though calib_rd_data_offset is not shown above in Figure-6. Note that the MIG example shown above has the read occurring on slot 1.

From viewing the out* signals in the Figure-7, you'll see the data is being pulled from the SDRAM memory. You'll see 8 read transactions that are back to back as earlier in the simulation, 8 read commands are sent from the PHY.



Read Command

Read data from the SDRAM

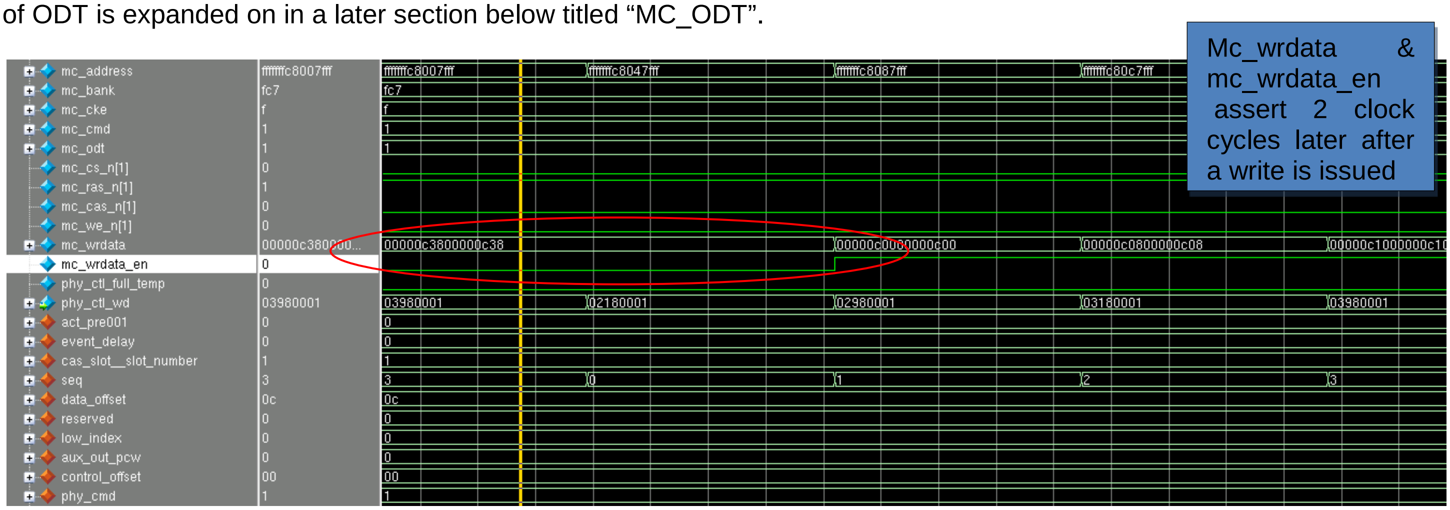
Figure-7 – After the read is called for by the PHY, data is pulled from the SDRAM.

Writing

A write begins with sending the phy_ctl_wd ,the lower three bits that control the phy_cmd show a value of 0x01, Figure 8. Besides sending the phy_ctl_wd, the mc_address, mc_bank, mc_cs_n, mc_ras_n, mc_cas_n, mc_we_n, mc_odt, and mc_cke signals along with it. This will place the proper command and the correct address for the SDRAM to execute this instruction. Remember to assert the write command on the command bits associated with the slot used. For example, a write on slot 1 would correlate to the JEDEC read command being sent on bit 1 of all command signals (i.e., mc_ras_n[1], mc_cas_n[1], mc_we_n[1]).

For example:
Phy_ctl_wd[2:0] – “001”
Mc_address – set to the target address
Mc_bank – set to the target bank
Mc_cs_n – ‘0’
Mc_ras_n – ‘1’
Mc_cas_n – ‘0’
Mc_we_n - ‘0’
Mc_odt – ‘1’
Mc_cke – 1’

Similar to the read, you will set the controller to write to the appropriate address and bank. You also send to the SDRAM memory a write operation. Because ODT is only used during writes, you will set ODT to ‘1’ for this operation. The duration of ODT is expanded on in a later section below titled “MC_ODT”.



Mc_wrdata & mc_wrdata_en assert 2 clock cycles later after a write is issued

Figure-8 –Controller and PHY Control Word initiating a write

The Out_FIFOs send a signal to show a write on slot 1 as expected and mc_wr_data and mc_wr_data_en is expected to see data 2 clock cycles later or 8 memory clock cycles. As the data is sent through the FIFOs, a short period of time into the writing the PHY begins writing the data to the memory, Figure-8. As with the read scenario, the user must ensure all of the same Memory Controller SDRAM signals are also being driven as they will alert the SDRAM of the oncoming write.

The data offset is based on several parameters: nCK_PERCLK, CWL, and the slot number

For nCK_PER_CLK = 4:

- Write Data Offset = CWL + 2 + slot number

For nCK_PER_CLK = 2:

- Write Data Offset = CWL - 2 + slot number

The write data offset can be seen by looking at the mc_data_offset signal.

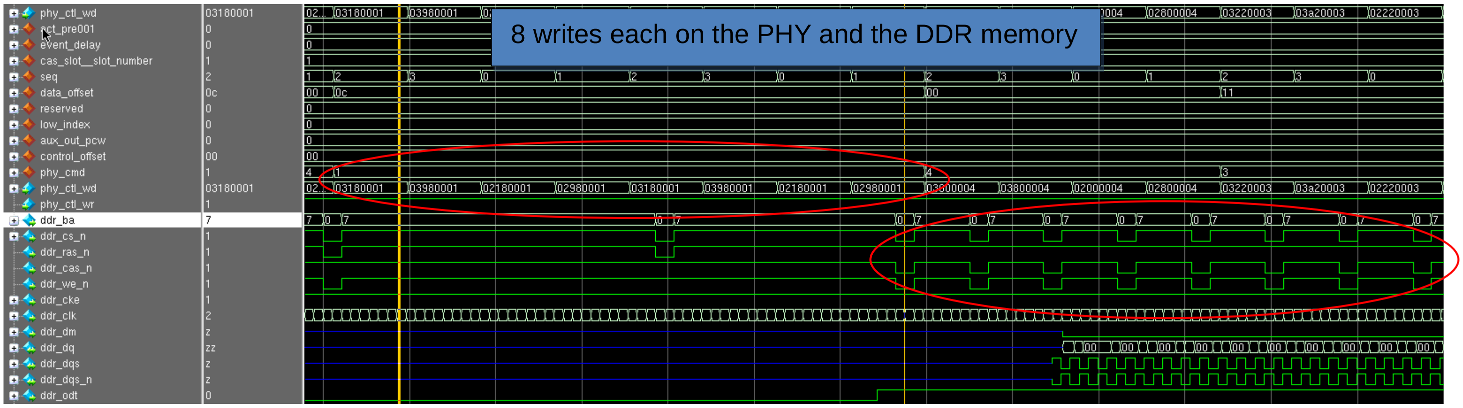

8 writes each on the PHY and the DDR memory

Figure -9 – PHY to DDR commands

For back to back transactions, it is the same as doing a single transaction except you will hold the values for multiple cycles. Above in Figure-9 is a capture of the PHY to DDR commands. Notice how there are 8 back to back writes and a short time later on the SDRAM memory you’ll see 8 back to back writes occurring.

Data Offset Calculation and Usage

During DQSFOUND calibration, the PHY determines the read data offset (time between read command and valid read data) for each bank containing a data group. The values found during this stage are used by the PHY during the remaining calibration stages and are sent by the calibration logic to the PHY on the calib_data_offset_0/1/2 signals. The calib_data_offset_0 signal is always used and calib_data_offset_1/2 are used when the memory interface is contained in 2/3 banks. After calibration completes, the memory controller is responsible for sending mc_data_offset/_1/_2 to the PHY when 1, 2, or 3 banks are used. Each bank may have different data offset values. The separate values are used for each bank's PHY Control Block's Data Offset field.

The data offset changes between reads, writes, and non-data commands. During writes, the value is CWL+2+slot#. During non-data commands, the value is 0. During reads, the value is calib_rd_data_offset_0/1/2 (found during DQSFOUND calibration) + slot#.

When using a custom controller, the data offset values used during calibration and normal operation reads may be different depending on the CWL. The values should match for reads with even CWL, and be off by 1 for reads with odd CWL. This is because reads/writes are assigned to slot1 by the memory controller whereas slot0 is used for even CWL for the MIG controller as only slot0 and slot1 are used for reads/writes.

When using a PHY only design, the custom memory controller must send the data offset values to the PHY. The above calculations should be used based on the values found during DQSFOUND calibration, CWL, and Slot#. The MIG memory controller does not support non-zero AL values however, the PHY does. If the custom controller supports a non-zero AL, the write and read offsets need to additionally add the AL value. For example, mc_data_offset = calib_data_offset_0 + AL + Slot#.

As a general rule of thumb, the read data offset should be approximately CL + 4 or 5, which is the CL plus the round trip delay on the PCB.

PHY Flag Usage

Of all of the flags, the phy_data_full flag is not used while phy_cmd_full and phy_ctl_full are both used in the MIG controller.
The Pre-FIFO (4 deep) is used when the Out_FIFO (8 deep) is full but with the 2 additional clock cycles added in MIG 1.4 this should prevent the Out_FIFOs from going full.

If you would like to avoid a FULL condition, then tying the entry logic to the phy_ctl_full and phy_cmd_full signals to hold off from sending another entry when they are asserted, table-1 and table-2.

phy_cmd_full	mc_cmd_wren	mc_cs	mc_ras	mc_cas	mc_we
0	1	CTT	CTT	CTT	CTT
1	0	0	1	1	1

Table -1

phy_ctl_full	mc_ctl_wren
0	1
1	0

Table-2

Note: CTT stands for Command Truth Table. This can be found in the DDR3 JEDEC Specification, Section 4.1

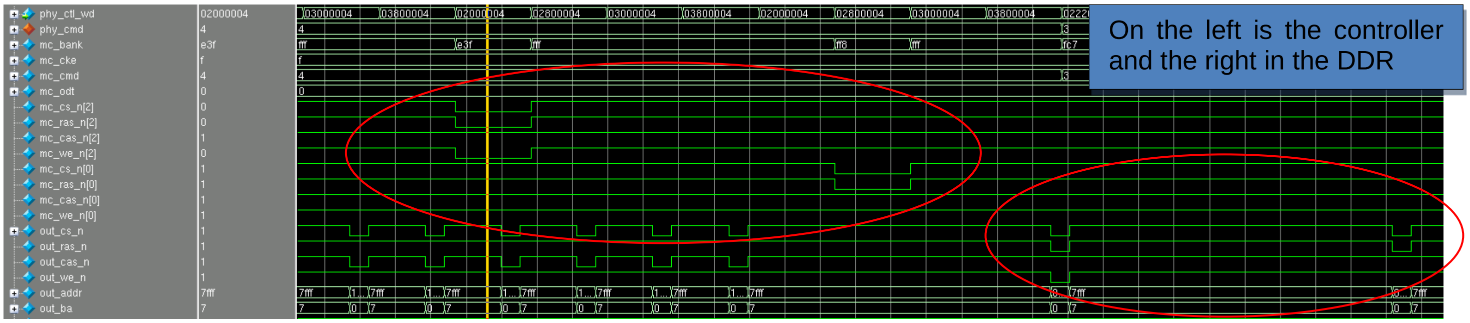
Activate/Precharge

For the MIG controller, the activate/precharge bits of the phy_ctl_wd are tied to 0 as they are not used as part of the design. The MIG controller takes care of this during non-data and data situations respectively. When doing a PHY Only design, you will want to put activate/precharge commands on non-data transactions and separate them according. Your controller will need to take into account tRRD, tRAS, tRCD, tRFC, tRP, and tRTP. Please view the User Guide for a more thorough explanation of all of these configuration parameters.

Precharge example:
Phy_ctl_wd[2:0] – "100"
Mc_address – set to the target address
Mc_bank – set to the target bank
Mc_cs_n – '0'
Mc_ras_n – '0'
Mc_cas_n – '1'
Mc_we_n - '0'
Mc_odt – '1'
Mc_cke – 1'

Activate example:
Phy_ctl_wd[2:0] – "100"
Mc_address – set to the target address
Mc_bank – set to the target bank
Mc_cs_n – '0'
Mc_ras_n – '0'
Mc_cas_n – '1'
Mc_we_n - '1'
Mc_odt – '1'
Mc_cke – 1'

Figure-10 shows slot 2 assigning a precharge command, slot 0 assigning an activate command, and slot 1 will have a write or read command later. Several clock cycles later the commands will be executed on the DDR interface.



On the left is the controller and the right in the DDR

Figure-10 – Activate/Precharge on the Controller and then at the DDR

Periodic Reads

Periodic reads will be up to the user to add into their controllers. Periodic reads are used to prevent drift due to PVT. This is required by the PHY and should be done every 1us.
If the periodic reads are not included, two things will happen that can cause problems:
1. The free running Phaser_IN ICLK will drift away from DQS. This exposes the memory system to issues when ICLK switches.
2. Read latency adjustments will not be done within the Phaser. This can cause issues with the switching logic in the Phaser_IN.

2:1 Designs

This AR is made to cover primarily a 4:1 controller. A 2:1 controller will be same in essence except that there are a few differences in the timing of the design. For example in a 4:1 controller each there are 4 possible CAS Slot positions, where as a 2:1 controller will only have 2 possible positions.

Burst Length Support

PHY only design is burst length 8 only. DDR3 itself is actually burst length 8 only also. Burst chop 4 is essentially a burst of 8 while masking the last for packets.

MC_ODT

The controller to PHY interfaces includes one ODT signal per rank (mc_odt[1:0]). The controller asserts the appropriate mc_odt signal the entire time we_n is asserted for write(s), plus one additional slow/fabric clock cycle (bufg clock domain), Figure-11. Therefore, for a single write command when loading the Address/Control OUT_FIFOs, the mc_odt[x] is asserted for two PHY Control Words (two slow/fabric clock cycles). Depending on the PHY Control Word slot assignment for the write command, ODT may be asserted a cycle earlier than the write command, which is acceptable.

CWL does not come into play from the controllers perspective because the ODT on and ODT off (internal RTT termination) are defined as CWL-2 in the DDR3 SDRAM spec. Thus, the Xilinx MIG 7 series controller and custom controllers need not worry about timing the ODT with respect to CWL. The extra fabric clock cycle in 4:1 mode accounts for the -2 and allows time for the write data to clear the DQ bus.

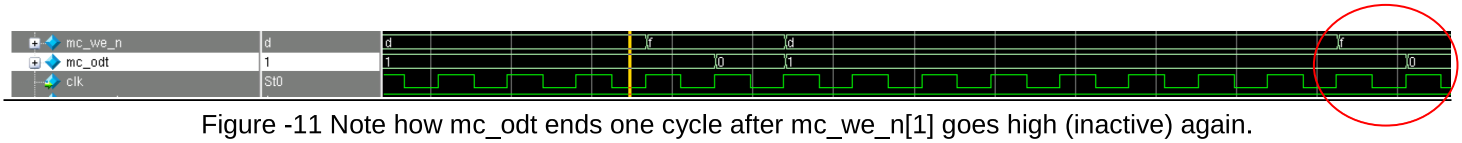


Figure -11 Note how mc_odt ends one cycle after mc_we_n[1] goes high (inactive) again.

Temperature Monitor Calibration

Starting in ISE 14.3/Vivado 2012.3 tools, DDR designs will include a temperature monitor system to maintain DQS center alignment in the read data window due to temperature drift and variation. The temperature monitor makes use of the 7 series XADC module.

The user_design/rtl/clocking/infrastructure.v module instantiates the XADC module and handles polling at configurable intervals. The XADC outputs the current temperature which the MIG design then uses to update the window if needed.

The rtl that receives the XADC temperature and makes adjustments as required is located in the user_design/rtl/phy/ddr_phy_tempmon.v module.

The memory controller sends an enable signal, tempmon_sameple_en, to the ddr_phy_tempmon module whenever a Refresh or ZQ Short Calibration has been sent to the DRAM and all pending transactions have cleared the DQ bus. The initial temperature read is performed to establish a baseline after the MIG 7 series calibration has completed. After each subsequent enable, the current temperature is compared to the baseline temperature. If the temperature change is sufficient, the module will adjust the PHASER_IN fine delay to mitigate temperature drift and sets a new baseline temperature. This process continues throughout normal operation.

For PHY Only Designs, it is up to the custom controller to send the enable signal, tempmon_sample_en. Xilinx recommends sending these during Refresh and ZQ Short Calibration so that any necessary adjustments can be made during these idle periods. The tempmon_sample_en should be held for the duration of the Refresh or ZQ Short Calibration and released after the command is complete before the next command occurs.

There is only one XADC per 7 series device. For designs that already make use of the XADC block, the user design can keep its current usage of the block and supply the MIG 7 series code with the temperature periodically. To do this, set the "XADC Instantiation" option to "Disabled". This option is available on the "FPGA Options" screen of the MIG 7 series tool. Disabling the XADC instantiation will create a top-level input port "device_temp[11:0]. The user design needs to drive this input port with the raw value as read from the DRP port defined by the XADC spec without any conversion.

Idle

The idle signal is used to reduce I/O power when the I/O is not required to wait for data. The idle signal is active high and will only be driven low when a read is to occur. The idle signal should be driven low the cycle after a read is assigned and will go high again once there are no pending requests in the memory controller.

The PHY will set the DCITERMDISABLE, IBUFDISABLE, and INTERMDISABLE on the I/O based on the following assignment (see ddr_mc_phy_wrapper):

```
// Idle powerdown when there are no pending reads in the MC
  assign data_io_idle_pwrdwn = DATA_IO_IDLE_PWRDWN == "ON" ? idle : 1'b0;
```

Example of the I/O Instantiation:

```
IOBUF_DCIEN #
           (
            .IBUF_LOW_PWR (IBUF_LOW_PWR)
            )
           u_iobuf_dqs
             (
              .DCITERMDISABLE (data_io_idle_pwrdwn),
              .IBUFDISABLE    (data_io_idle_pwrdwn),
              .I              (out_dqs[p]),
              .T              (ts_dqs[p]),
              .O              (in_dqs[p]),
              .IO             (ddr_dqs[p])
              );
```

Sending Non-Data Commands during Idle Periods

When designing a PHY Only design, the designer must be careful of the CMD_FIFO going empty, but not the CTL_FIFO. The CTL_FIFO does not know the status of the CMD_FIFO and its job is to pull any command from the CMD_FIFO if there is a control world inside of the CTL_FIFO.

This means the designer should not stop writing commands to the CTL_FIFO, via the CTL_WD, as this will kill the flow control. You should be sending non-data commands when nothing is happening, such as a NOP.

Phy_ctl_wd[2:0] – “100”
Mc_cmd_cs/ras/cas/we = NOOP

Constraining the PHY Only Design

All MIG generated XDC (for Vivado designs) or UCF (for ISE designs) must be used with the PHY Only design. These constraints are required for proper operation of the MIG PHY and all must be applied.

Conclusion

If this document does not help to resolve the problem, please create a WebCase with Xilinx Technical Support. Attach all of the captured ChipScope Pro waveforms, and the details of your investigation and analysis.

Revision History

08/20/2012 - Initial release
08/28/2012 - Added MC_ODT section
09/07/2012 – Added examples Read, Write, Activate/Precharge sections.
09/25/2012 – Formatted the Read, Write and Activate/Precharge sections. Added Temperature Monitor Calibration Section
11/01/2012 – Added additional signals to the Signals of Interest. Added Idle and Sequence sections.
02/11/2013- Added section “Sending non-data commands for idle periods”
02/05/2014 – Added Constraint the PHY Only Design section. Added information on command usage for slots.
Corrected the command examples for Activate and Precharge commands.