Xilinx Answer 43879
MIG 7 Series DDR3/DDR2 - Hardware Debug Guide

Introduction

Calibration failures and data errors can occur for many reasons and the debug of these errors can be time consuming. This answer record is intended to provide a clear step-by-step debug process to quickly identify the root cause of the failure and move to resolution.

MIG Usage

To focus the debug of calibration or data errors, use the provided MIG Example Design on the targeted board with the Debug Feature enabled through the MIG 7 Series GUI. The latest MIG 7 Series release should be used to generate the Example Design.

Debug Tools

Many tools are available to debug memory interface design issues. This section indicates which resources are useful for debugging a given situation.

Example Design

Generation of a DDR2 or DDR3 design through the MIG 7 series tool produces an example design and a user design. The example design includes a synthesizable testbench with a traffic generator that is fully verified in simulation and hardware. This example design can be used to observe the behavior of the MIG 7 series design and can also aid in identifying board-related problems. For complete details on the example design, see the “Quick Start Example Design” in the *7 Series FPGAs Memory Interface Solutions User Guide* (UG586). This debug guide answer record further describes using the example design to perform hardware validation.

Debug Signals

The MIG 7 series tool includes a Debug Signals Control option on the FPGA Options screen. Enabling this feature allows calibration, tap delay, and read data signals to be monitored using the ChipScope™ analyzer. Selecting this option port maps the debug signals to ILA and VIO cores of the ChipScope analyzer in the design top module. For details on enabling this debug feature, see the “Getting Started with the CORE Generator™ Tool” section in the *7 Series FPGAs Memory Interface Solutions User Guide* (UG586). . The debug port is disabled for functional simulation and can only be enabled if the signals are actively driven by the user design.

Reference Boards

The KC705 and VC707 evaluation kits are Xilinx development boards that include FPGA interfaces to a DDR3 SODIMM. These boards can be used to test user designs and analyze board layout.

ChipScope Pro Tool

The ChipScope™ Pro tool inserts logic analyzer, bus analyzer, and VIO software cores directly into the design. The ChipScope Pro tool allows the user to set trigger conditions to capture application and MIG signals in hardware. Captured signals can then be analyzed through the ChipScope Pro logic analyzer tool.

General Checks

This section details the list of general checks, primarily board level, which need to be verified before moving forward with the debug process. Strict adherence to the proper board design is critical in working with high speed memory interfaces.

- **Ensure all guidelines referenced in the "Design Guidelines" sections of the *7 Series FPGAs Memory Interface Solutions User Guide* (UG586) have been followed.** The Design Guidelines section includes information on trace matching, PCB Routing, noise, termination, I/O Standards, and pin/bank requirements. Adherence to these guidelines, along with proper board design and signal integrity analysis, is critical to the success of high-speed memory interfaces.
- Measure all voltages on the board during idle and non-idle times to ensure the voltages are set appropriately and noise is within specifications.
  - Ensure the termination voltage regulator (Vtt) is turned on (set to 0.75V).
  - Ensure Vref is measured.
- When applicable, check VRN/VRP resistors. Note the values are not the same as Virtex-6 FPGA.
- Look at the clock inputs to ensure they are clean.
- Check the reset to ensure the polarity is correct and the signal is clean.
- Check terminations. The *7 Series FPGAs Memory Interface Solutions User Guide* (UG586) should be used as a guideline.
- Perform general signal integrity analysis.
  - IBIS simulations should be run to ensure terminations, ODT, and output drive strength settings are appropriate.
  - Observe DQ/DQS on a scope at the memory. View the alignment of the signals and analyze the signal integrity during both writes and reads.
  - Observe the Address and Command signals on a scope at the memory. View the alignment and analyze the signal integrity.
- Verify the memory parts on the board(s) in test are the correct part(s) set through MIG. The timing parameters and signals widths (i.e., address, bank address) must match between the RTL and physical parts. Read/write failures can occur due to a mismatch.
- Verify SDRAM pins are behaving correctly. Look for floating or grounded signals. It is rare, but manufacturing issues with the memory devices can occur and result in calibration failures.
- If Data Mask (DM) is not being used, ensure DM is tied Low at the memory with the appropriate termination as noted in the memory datasheet.
- Measure the CK/CK_n, DQS/DQS_n, and system clocks for duty cycle distortion and general signal integrity.
- If internal Vref is used, ensure the constraints are set appropriately according to the Xilinx Constraints Guide. When the constraints are applied properly, a note similar to the following will be in the .bgn Bit
- Gen report file:
  - There were two CONFIG constraint(s) processed from example_top.pcf.

    ```
    CONFIG INTERNAL_VREF_BANK12 = "0.75"
    CONFIG INTERNAL_VREF_BANK14 = "0.75"
    ```

- Check the iodelay_ctrl ready signal.
- Check the PLL lock.
- Check the phaser_ref lock signal.
- Bring the init_calib_complete out to a pin and check with a scope.

Calibration Stages

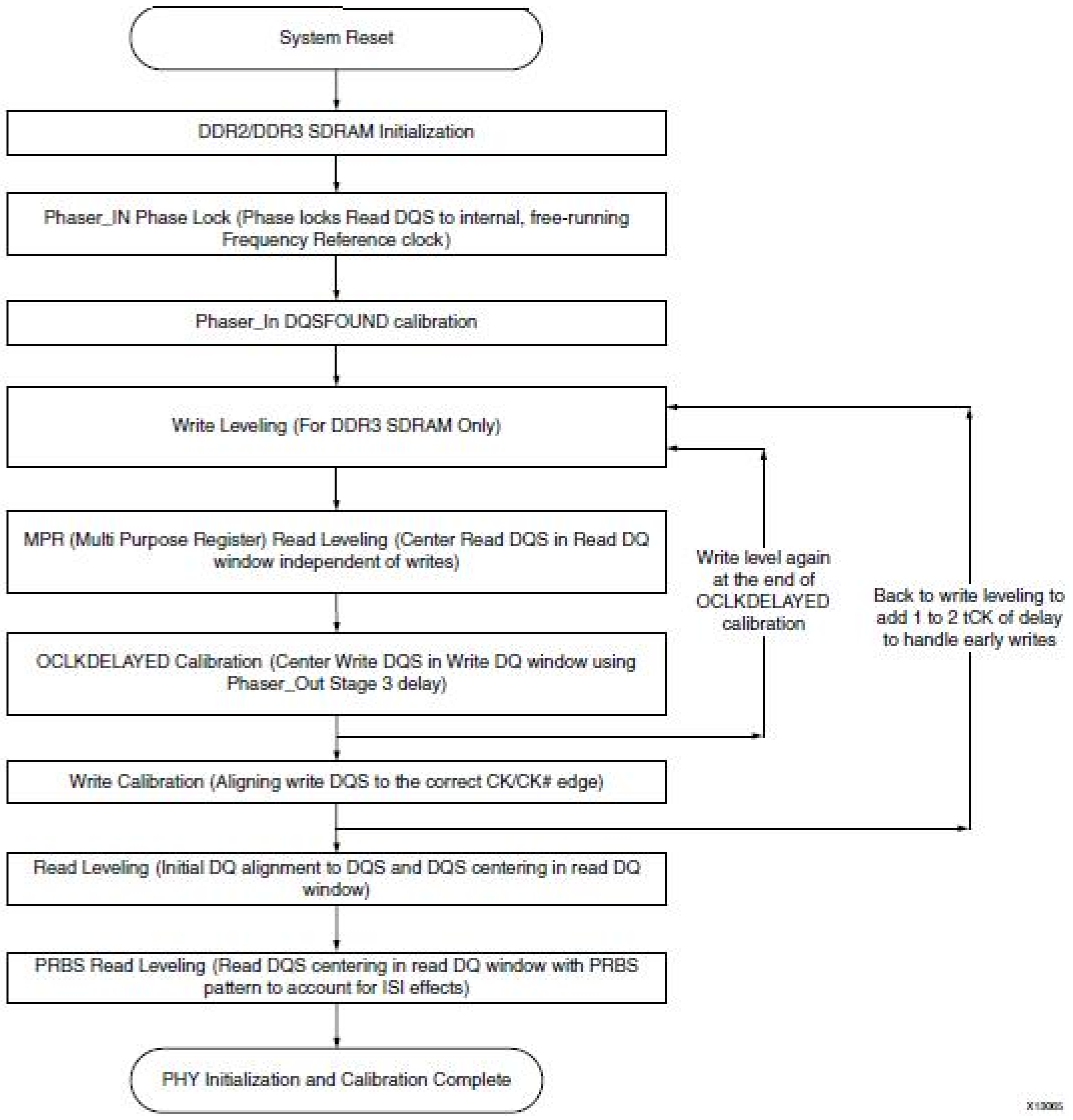

System Reset
DDR2/DDR3 SDRAM Initialization
Phaser_IN Phase Lock (Phase locks Read DQS to internal, free-running Frequency Reference clock)
Phaser_In DQSFOUND calibration
Write Leveling (For DDR3 SDRAM Only)
MPR (Multi Purpose Register) Read Leveling (Center Read DQS in Read DQ window independent of writes)
OCLKDELAYED Calibration (Center Write DQS in Write DQ window using Phaser_Out Stage 3 delay)
Write Calibration (Aligning write DQS to the correct CK/CK# edge)
Read Leveling (Initial DQ alignment to DQS and DQS centering in read DQ window)
PRBS Read Leveling (Read DQS centering in read DQ window with PRBS pattern to account for ISI effects)
PHY Initialization and Calibration Complete
Write level again at the end of OCLKDELAYED calibration
Back to write leveling to add 1 to 2 tCK of delay to handle early writes
X13005

Figure 1 - Calibration Stages

Memory Initialization

The PHY executes a JEDEC-compliant DDR2 or DDR3 initialization sequence following the de-assertion of system reset. Each DDR2 or DDR3 SDRAM has a series of mode registers accessed via mode register set (MRS) commands. These mode registers determine various SDRAM behaviors, such as burst length, read and write CAS latency, and additive latency. The MIG 7 series designs will never issue a calibration failure during Memory Initialization.

All other initialization/calibration stages are reviewed in the appropriate Debugging Calibration Stages section below.

Determine the Failing Calibration Stage

Using ChipScope analyzer, configure the device and open the provided “example_top.cpj” file. This file is generated in the “example_design/par” directory when the Debug Signals feature is enabled during core generation. Observe the following debug signals in the provided “Basic” ILA core. This will indicate which calibration stage failed:

Table 1: DDR3/DDR2 “Basic ILA” Debug Signals

Signal Name	ILA/VIO Connection	Signal Description
init_calib_complete	ila_basic[0]	Signifies memory initialization and calibration have completed successfully
dbg_wrlvl_start	ila_basic[1]	Signifies the start of the Write Leveling stage of calibration
dbg_wrlvl_done	ila_basic[2]	Signifies successful completion of the Write Leveling stage of calibration
dbg_wrlvl_err	ila_basic[3]	Signifies the Write Leveling stage of calibration exhibited errors and did not complete
dbg_pi_phaselock_start	ila_basic[4]	Signifies the start of the PHASELOCK stage of calibration
dbg_pi_phaselock_done	ila_basic[5]	Signifies successful completion of the PHASELOCK stage of calibration
dbg_pi_phaselock_err	ila_basic[6]	Signifies the PHASELOCK stage of calibration exhibited errors and did not complete
dbg_pi_dqsfound_start	ila_basic[7]	Signifies the start of the DQSFOUND stage of calibration
dbg_pi_dqsfound_done	ila_basic[8]	Signifies successful completion of the DQSFOUND stage of calibration
dbg_pi_dqsfound_err	ila_basic[9]	Signifies the DQSFOUND stage of calibration exhibited errors and did not complete
dbg_rdlvl_start[0]	ila_basic[10]	Signifies the start of Read Leveling Stage 1 calibration
dbg_rdlvl_start[1]	ila_basic[11]	Signifies the start of the MPR stage of calibration
dbg_rdlvl_done[0]	ila_basic[12]	Signifies the successful completion of Read Leveling Stage 1 calibration
dbg_rdlvl_done[1]	ila_basic[13]	Signifies the successful completion of the MPR Stage of calibration
dbg_rdlvl_err[0]	ila_basic[14]	Signifies Read Leveling Stage 1 calibration exhibited errors and did not complete
dbg_rdlvl_err[1]	ila_basic[15]	Signifies the MPR stage of calibration exhibited errors and did not complete
dbg_oclkdelay_calib_start	ila_basic[16]	Signifies the start of the OCLKDELAY stage of calibration
dbg_oclkdelay_calib_done	ila_basic[17]	Signifies successful completion of the OCLKDELAY stage of calibration
dbg_wrcal_start	ila_basic[19]	Signifies the start of the Write Calibration stage of calibration
dbg_wrcal_done	ila_basic[20]	Signifies successful completion of the Write Calibration stage of calibration
dbg_wrcal_err	ila_basic[21]	Signifies Write Calibration exhibited errors and did not complete

Each of these stages can be used as triggers in the Basic, Write Path, and Read Path ILA ChipScope cores to determine why the stage failed. The section below details these debugging steps.

Debug Signals

Table 2: DDR23/DDR2 Debug Signals

Debug Signal	ILA/VIO Connection	Signal Description

init_calib_complete	ila_basic[0]	Signifies memory initialization and calibration have completed successfully
dbg_wrlvl_start	ila_basic[1]	Signifies the start of the Write Leveling stage of calibration
dbg_wrlvl_done	ila_basic[2]	Signifies successful completion of the Write Leveling stage of calibration
dbg_wrlvl_err	ila_basic[3]	Signifies the Write Leveling stage of calibration exhibited errors and did not complete
dbg_pi_phaselock_start	ila_basic[4]	Signifies the start of the PHASELOCK stage of calibration
dbg_pi_phaselock_done	ila_basic[5]	Signifies successful completion of the PHASELOCK stage of calibration
dbg_pi_phaselock_err	ila_basic[6]	Signifies the PHASELOCK stage of calibration exhibited errors and did not complete
dbg_pi_dqsfound_start	ila_basic[7]	Signifies the start of the DQSFOUND stage of calibration
dbg_pi_dqsfound_done	ila_basic[8]	Signifies successful completion of the DQSFOUND stage of calibration
dbg_pi_dqsfound_err	ila_basic[9]	Signifies the DQSFOUND stage of calibration exhibited errors and did not complete
dbg_rdlvl_start[0]	ila_basic[10]	Signifies the start of Read Leveling Stage 1 calibration
dbg_rdlvl_start[1]	ila_basic[11]	Signifies the start of the MPR stage of calibration
dbg_rdlvl_done[0]	ila_basic[12]	Signifies the successful completion of Read Leveling Stage 1 calibration
dbg_rdlvl_done[1]	ila_basic[13]	Signifies the successful completion of the MPR Stage of calibration
dbg_rdlvl_err[0]	ila_basic[14]	Signifies Read Leveling Stage 1 calibration exhibited errors and did not complete
dbg_rdlvl_err[1]	ila_basic[15]	Signifies the MPR stage of calibration exhibited errors and did not complete
dbg_oclkdelay_calib_start	ila_basic[16]	Signifies the start of the OCLKDELAY stage of calibration
dbg_oclkdelay_calib_done	ila_basic[17]	Signifies successful completion of the OCLKDELAY stage of calibration
dbg_wrcal_start	ila_basic[19]	Signifies the start of the Write Calibration stage of calibration
dbg_wrcal_done	ila_basic[20]	Signifies successful completion of the Write Calibration stage of calibration
dbg_wrcal_err	ila_basic[21]	Signifies Write Calibration exhibited errors and did not complete
dbg_phy_init	ila_basic[27:22]	State variable for the PHY Init state machine. States can be decoded in the ddr_phy_init module.

dbg_rddata	ila_basic[92:29]	Read data read out of the IN_FIFO for the DQS group selected through dbg_dqs on the VIO. This is a 64-bit bus. For an example of how the data is captured, see the Debug Read Data table below. This debug port will not capture ECC data.
cmp_data	ila_basic[194:131]	Register version of compare data from the Traffic Generator.
dq_error_bytelane_cmp	ila_basic[203:195]	Indicates which byte has data comparison error for the Traffic Generator.
wl_state_r	ila_wrpath[4:0]	State variable for the Write Leveling State Machine. States can be decoded in the ddr_phy_wrlvl.v module.
wrcal_dqs_cnt_r	ila_wrpath[9:6]	Signifies the DQS byte group being calibrated during Write Leveling. The algorithm will sequentially step through the DQS byte groups until write leveling completes successfully or a data byte group fails due a 0 to 1 transition not being detected on DQ.
wl_edge_detect_valid	ila_wrpath[10]	Signifies valid time Write Leveling algorithm is searching for edge
rd_data_edge_detect	ila_wrpath[11]	Signifies Write Leveling calibration found the 0-to-1 edge transition
wl_po_fine_cnt	ila_wrpath	Phaser_out Fine Taps found during Write Leveling. Byte capture based on VIO dbg_dqs setting.
wl_po_fine_cnt_0-8	ila_wrpath	Phaser_out Fine Taps found during Write Leveling
wl_po_coarse_cnt	ila_wrpath	Phaser_out Coarse Taps found during Write Leveling. Byte capture based on VIO dbg_dqs setting.
wrl_po_coarse_cnt_0-8	ila_wrpath	Phaser_out Coarse Taps found during Write Leveling
ocal_tap_cnt	ila_wrpath[37:32]	Tap used to move to the center of the window after finding first and second edges
ocal_edge1_found	ila_wrpath[38]	Signifies that the first edge is found during OCLKDELAY calibration when the taps are decremented from 30
ocal_edge2_found	ila_wrpath[39]	Signifies that the second edge is found during OCLKDELAY calibration when the taps are incremented from 30
ocal_edge1_taps	ila_wrpath[45:40]	Tap value when the first edge is found during OCLKDELAY calibration
ocal_edge2_taps	ila_wrpath[51:46]	Tap value when the second edge is found during OCLKDELAY calibration
ocal_state_r	ila_wrpath[56:52]	State variable for the OCLKDELAY state machine. States can be decoded in the ddr_phy_oclkdelay_cal module
pat_data_match	ila_wrpath[64]	Asserts when the valid pattern is detected
pat_data_match_valid	ila_wrpath[65]	Toggles when the correct pattern is detected

wrcal_dqs_cnt_r	ila_wrpath[69:66]	Current DQS group being calibrated in Write Calibration. When wrcal_start asserts, wrcal_dqs_cnt_r is 0. The algorithm sequentially steps through the DQS byte groups checking to see if the read data pattern matches the expected FF00AA5555AA9966 pattern. If the pattern matches, wrcal_dqs_cnt increments by 1. The algorithm then starts looking for the correct data pattern on the next byte until it reaches DQS_WIDTH-1 or a data byte group fails due to the data pattern not being detected properly. The wrcal_dqs_cnt stays at DQS_WIDTH-1 after wrcal_done signal is asserted.
cal2_state	ila_wrpath[74:70]	Write Calibration state machine variable. States can be decoded in the ddr_phy_wrcal.v module.
not_empty_wait_cnt	ila_wrpath[79:75]	Count value during write calibration pattern detection. Maximum count is 'd31. If count reaches d'31, write calibration will fail with the assertion of wrcal_err.
early1_data	ila_wrpath[80]	Asserts when the pattern detected is one CK clock cycle early. When this is asserted, the write leveling algorithm moves the CK clock one cycle. After CK is moved, the write calibration algorithm restarts pattern detection.
early2_data	ila_wrpath[81]	Asserts when the pattern detected is two CK clock cycles early. When this is asserted, the write leveling algorithm moves the CK clock two cycles. After CK is moved, the write calibration algorithm restarts pattern detection.
stg2_tap_cnt	ila_wrpath	Final Phaser_OUT Stage 3 tap count found during OCLKDELAYED Calibration.
dbg_pi_phase_locked_phy4lanes	ila_rdpath[11:0]	Signifies which of the PHASER_IN lanes has achieved lock.
dbg_pi_dqs_found_lanes_phy4lanes	ila_rdpath[23:12]	Signifies which of the PHASER_IN lanes is able to find the DQS.
cal1_state_r	ila_rdpath[45:40]	State machine variable for MPR and Read Leveling Stage 1. States can be decoded in the ddr_phy_rdlvl.v module.
cal1_cnt_cpt_r	ila_rdpath[49:46]	Signifies the byte that failed MPR read leveling or read leveling stage 1
mux_rd_rise0	ila_rdpath / ila_wrpath	Data pattern received on rising edge 0.
mux_rd_fall0	ila_rdpath / ila_wrpath	Data pattern received on falling edge 0.
mux_rd_rise1	ila_rdpath / ila_wrpath	Data pattern received on rising edge 1.
mux_rd_fall1	ila_rdpath / ila_wrpath	Data pattern received on falling edge 1.
mux_rd_rise2	ila_rdpath / ila_wrpath	Data pattern received on rising edge 2.
mux_rd_fall2	ila_rdpath / ila_wrpath	Data pattern received on falling edge 2.
mux_rd_rise3	ila_rdpath / ila_wrpath	Data pattern received on rising edge 3.

mux_rd_fall3	ila_rdpath / ila_wrpath	Data pattern received on falling edge 3.
pat_data_match	ila_rdpath[114]	Toggles when the correct pattern is detected
mux_rd_valid	ila_rdpath[115],	Asserts when the valid pattern is detected
dbg_cpt_first_edge_cnt	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when the first edge in MPR and Read Leveling Stage 1 is found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_first_edge_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when the first edge in MPR and Read Leveling Stage 1 is found.
dbg_cpt_second_edge_cnt	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when then second edge in MPR and Read Leveling Stage 1 is found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_second_edge_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when then second edge in MPR and Read Leveling Stage 1 is found.
dbg_cpt_tap_cnt	ila_rdpath, per DQS and per RANK	Signifies the center tap moved to based on when the first and second edges were found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_tap_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies the center tap moved to based on when the first and second edges were found.
dbg_dq_idelay_tap_cnt	ila_rdpath, per DQS and per RANK	IDELAY tap value for MPR and Read Leveling Stage 1. This should be within 2-3 taps across all DQS byte groups. Byte capture based on VIO dbg_dqs setting.
dbg_dq_idelay_tap_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	IDELAY tap value for MPR and Read Leveling Stage 1. This should be within 2-3 taps across all DQS byte groups.
dbg_rd_data_offset_0	ila_rdpath[35:24].	Read Data Offset found during calibration
dbg_calib_rd_data_offset_1	ila_rdpath[175:164]	Read Data Offset found during calibration
dbg_calib_rd_data_offset_2	ila_rdpath[187:176]	Read Data Offset found during calibration
dbg_data_offset	ila_rdpath[193:188]	Data Offset used during normal operation. Value will change during writes, reads, and idle. During writes, it is CWL+2+slot#. During non-data commands, it is 0. During reads, it should match what was found during DQSFOUND calibration (rd_data_offset_ranks).
dbg_data_offset_1	ila_rdpath[199:194]	Data Offset used during normal operation. Value will change during writes, reads, and idle. During writes, it is CWL+2+slot#. During non-data commands, it is 0. During reads, it should match what was found during DQSFOUND calibration (rd_data_offset_ranks).
dbg_data_offset_2	ila_rdpath[205:200]	Data Offset used during normal operation. Value will change during writes, reads, and idle. During writes, it is CWL+2+slot#. During non-data commands, it is 0. During reads, it should match what was found during DQSFOUND calibration (rd_data_offset_ranks).
dbg_bit	vio_sync_out[8:0]	

dbg_dqs	vio_sync_out[12:9]	Selects the DQS byte group for the debug signals. For example, set to 4'b0000 to view the results on DQS[0].
vio_modify_enable	vio_sync_out[36]	Set to 1 to vary the command Traffic Generator command pattern.
vio_addr_mode_value	vio_sync_out[43:41]	Sets the address mode used by the Traffic Generator. 1: Fixed Address 2: PRBS Address 3: Sequential Address
vio_bl_mode_value	vio_sync_out[49:48]	Sets the burst mode used by the Traffic Generator
		1: Fixed bl.
		2: PRBS bl. If bl_mode value is set to 2, the addr_mode value is forced to 2 to generate the PRBS address.
vio_data_mask_gen	vio_sync_out[58]	Traffic generator Data Mask generation
vio_pause_traffic	vio_sync_out[59]	Set to 1 to pause the Traffic Generator.
dbg_clear_error	vio_sync_out[63]	Set to clear Traffic Generator errors. This signal can be used in checking for single bit errors or measuring a read window.

Debugging PHASER_IN PHASELOCKED Calibration Failures (dbg_pi_phaselock_err=1)

Calibration Overview

During this stage of calibration, each PHASER_IN is placed in the read calibration mode to phase align its free-running frequency reference clock to the associated read DQS. The calibration logic issues back-to-back read commands to provide the PHAESER_IN block with a continuous stream of DQS pulses for it to achieve lock. Each DQS has an associated PHASER_IN block. Dbg_pi_phase_locked asserts when all PHASER_INs have achieved lock and the PHASER_INs are then placed in normal operation mode.

Debug

If PHASER_IN PHASELOCKED calibration failed, probe the DQS at the memory. A continuous stream of DQS pulses must be seen for lock to occur. Verify the signal integrity of the DQS pulses.

Debugging PHASER_IN DQSFOUND Calibration Failures (dbg_pi_dqsfound_err=1)

Calibration Overview

In this stage of calibration, the different DQS groups are aligned to the same PHY_Clk and the optimal read data offset position is found with respect to the read command. The calibration logic issues a set of four back-to-back reads with gaps in between. Each Phaser_IN detects the read DQS preamble. A single read data offset value is determined for all DQS groups. This data offset is then used during read requests to the PHY_CONTROL block.

Debug

- If the DQSFOUND stage fails, probe DQS at the memory. Sets of four back-to-back reads should be seen. Read DQS(s) is required by the PHASER_IN(s) to establish the read_data_offset value. If the design is stuck in the DQSFOUND stage, start observing the quality of DQS at the memory.
- Look at the read_data_offset values. There are two sets of read_data_offset values that need to be compared.
    - To determine the read data offset found at the end of DQSFOUND calibration, look at dbg_rd_data_offset_0, dbg_calib_data_offset_1 (only when more than 1 bank is used), dbg_calib_data_offset_2 (only when 3 banks are used).
    - To determine the data offset used during normal operation reads, look at dbg_data_offset, dbg_data_offset_1 (only when more than one bank is used), and dbg_data_ofset_2 (only when three banks are used).
        - These signals will change 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 should match what was found during DQSFOUND calibration (dbg_rd_data_offset_0, dbg_rd_data_offset_1, and dbg_rd_data_offset_2).
- Compare the read data offset values used during calibration and normal operation reads. These values should match for reads with even CWL and be off by 1 for reads with odd CWL. One additional offset is added for odd CWL values because reads/writes are assigned to slot1 by the memory controller, whereas slot0 is used for even CWL.
- The read data offset should be equal to CL + 4 or 5 which is the CL plus the round trip delay on the PCB.
- When this stage fails (pi_dqsfound_err=1), look to see if any of the dbg_calib_rd_data_offset/_1/_2 have calculated offsets. If not, focus on the DQS signals associated with the failing bank by probing each and analyzing the signal integrity.

If pi_dqsfound_err asserted, denoting a failure during DQSFOUND calibration, use **pi_dqsfound_err=1 as the trigger.** If this stage completed successfully with the asserting of pi_dqsfound_done=1, use **pi_dqsfound_done=1 as the trigger** to analyze how the stage completed. Look at dbg_rd_data_offset, dbg_calib_rd_data_offest_1, and dbg_calib_rd_data_offest _2, these values should vary by one at the most. Next, compare these values to the values used during normal operation reads on the dbg_data_offset, dbg_data_offset_1 and dbg_data_offset_2 signals. Record the results in the "7 Series DDR3 Calibration Results" spreadsheet.

Table 3: Debug Signals of Interest for DQSFOUND Calibration

Debug Signal	ILA/VIO Connection	Signal Description
dbg_pi_dqsfound_start	ila_basic[7]	Signifies the start of the DQSFOUND stage of calibration
dbg_pi_dqsfound_done	ila_basic[8]	Signifies successful completion of the DQSFOUND stage of calibration
dbg_pi_dqsfound_err	ila_basic[9]	Signifies the DQSFOUND stage of calibration exhibited errors and did not complete
dbg_rd_data_offset_0	ila_rdpath[35:24].	Read Data Offset found during calibration
dbg_calib_rd_data_offset_1	ila_rdpath[175:164]	Read Data Offset found during calibration
dbg_calib_rd_data_offset_2	ila_rdpath[187:176]	Read Data Offset found during calibration
dbg_data_offset	ila_rdpath[193:188]	Data Offset used during normal operation. Value will change during writes, reads, and idle. During writes, it is CWL+2+slot#. During non-data commands, it is 0. During reads, it should match what was found during DQSFOUND calibration (rd_data_offset_ranks).
dbg_data_offset_1	ila_rdpath[199:194]	Data Offset used during normal operation. Value will change during writes, reads, and idle. During writes, it is CWL+2+slot#. During non-data commands, it is 0. During reads, it should match what was found during DQSFOUND calibration (rd_data_offset_ranks).

dbg_data_offset_2	ila_rdpath[205:200]	Data Offset used during normal operation. Value will change during writes, reads, and idle. During writes, it is CWL+2+slot#. During non-data commands, it is 0. During reads, it should match what was found during DQSFOUND calibration (rd_data_offset_ranks).

Expected ChipScope Tool Results

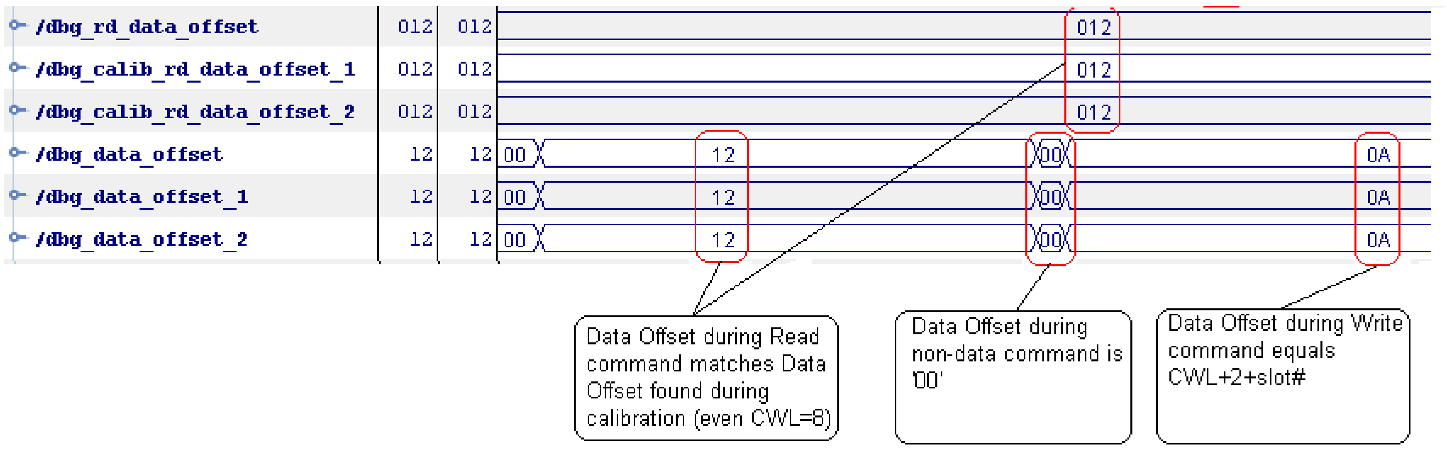

/dbg_rd_data_offset 012 012 012
/dbg_calib_rd_data_offset_1 012 012 012
/dbg_calib_rd_data_offset_2 012 012 012
/dbg_data_offset 12 12 00 12 00 0A
/dbg_data_offset_1 12 12 00 12 00 0A
/dbg_data_offset_2 12 12 00 12 00 0A
Data Offset during Read command matches Data Offset found during calibration (even CWL=8)
Data Offset during non-data command is '00'
Data Offset during Write command equals CWL+2+slot#

Figure 2 - Expected ChipScope Tool Results

Debugging Write Leveling Failures (dbg_wrlvl_err = 1)

Calibration Overview

Write leveling, a new features in DDR3 SDRAMs, allows the controller to adjust each write DQS phase independently with respect to the CK forwarded to the DDR3 SDRAM device. This compensates for the skew between DQS and CK and meets the tDQSS specification. During this stage, the PHY logic asserts the Write_Calib_N input to the PHY Control Block to indicate the start of write leveling. Periodic write requests are issued to the PHY Control Block to generate periodic DQS pulses. The PHASER_IN outputs a free-running clock to capture the DQ feedback into the DQ IN_FIFOs. The PHASER_OUT fine and coarse taps are used to phase shift DQS one tap at a time until a 0-to-1 transition is seen on the feedback DQ.

Write Leveling is performed at three different points during the calibration process. After memory initialization completes, the PHASER_OUT fine and coarse taps are set to zero. Write Leveling is then initially performed to align DQS to CK. After OCLKDELAYED calibration completes, the coarse tap values found during the initial Write Leveling are carried over and the fine taps are reset to zero. Write Leveling is performed again to ensure the DQS-to-CK relationship is still correct. Finally, during Write Calibration both the fine and coarse delays are carried over and final adjustments are made when necessary. During Write Calibration, the appropriate pattern must be detected. If Write Leveling aligned DQS to the wrong CK clock, final PHASER_OUT fine/coarse delay adjustments are required to move DQS up to two CK clock cycles. This section shows how to capture the Write Leveling results after each of these adjustments.

Debug Steps

- Verify DQS is toggling on the board. The FPGA sends DQS during Write Leveling. If DQS is not toggling, something is wrong with the setup and the General Checks section of this answer record should be thoroughly reviewed.
- Verify fly-by-routing is implemented correctly on the board.

- Verify CK to DQS trace routing. The CK clocks should be longer then DQS. The recommended value for additional total electrical delay on CK/CK# relative to DQS/DQS# is 150ps, but any value greater than 0ps is acceptable.
- The Mode Registers must be properly set up to enable Write Leveling. Specifically, address bit A7 must be correct. If the part chosen in MIG is not accurate or there is a problem with the connection of the address bits on the board, this could be an issue. If the Mode Registers are not set up to enable Write Leveling, the 0-to-1 transition will never be seen. Note that for dual rank design when address mirroring is used, address bit A7 is not the same between the two ranks.
- **When dbg_wrlvl_err asserts (equals 1)**, users must determine during which of the three different stages write leveling is performed the failure occurred. Set the **ChipScope trigger to dbg_wrlvl_err=1** and look at the other "DDR Basic" signals to see which stages completed.

  1. If only PHASELOCK and DQSFOUND completed, the write leveling failure occurred during the initial run through.
  2. If dbg_wrcal_start did not assert, the write leveling failure occurred after OCLKDELAYED calibration.
  3. If dbg_wrcal_start asserted but dbg_wrcal_done did not, the write leveling failure occurred during the final run through during Write Calibration.

- **When dbg_wrlvl_done asserts (equals 1)** and the results of each Write Leveling stage is of interest, separately use the following **three ChipScope triggers** to capture the Write Leveling tap results for each stage. Seeing how Write Leveling completed is useful to see how far apart the taps are for different DQS byte groups.

  1. **dbg_wrlvl_done=1**
  2. **dbg_wrcal_start=1**
  3. **init_calib_complete=1**

- To capture the write leveling results at each stage, change/increment dbg_dqs on the VIO and set the appropriate trigger as noted above. Look at the taps results and record in the "7 Series DDR3 Calibration Results" spreadsheet. Later releases of MIG include results for all DQS byte groups removing the need to use dbg_dqs.

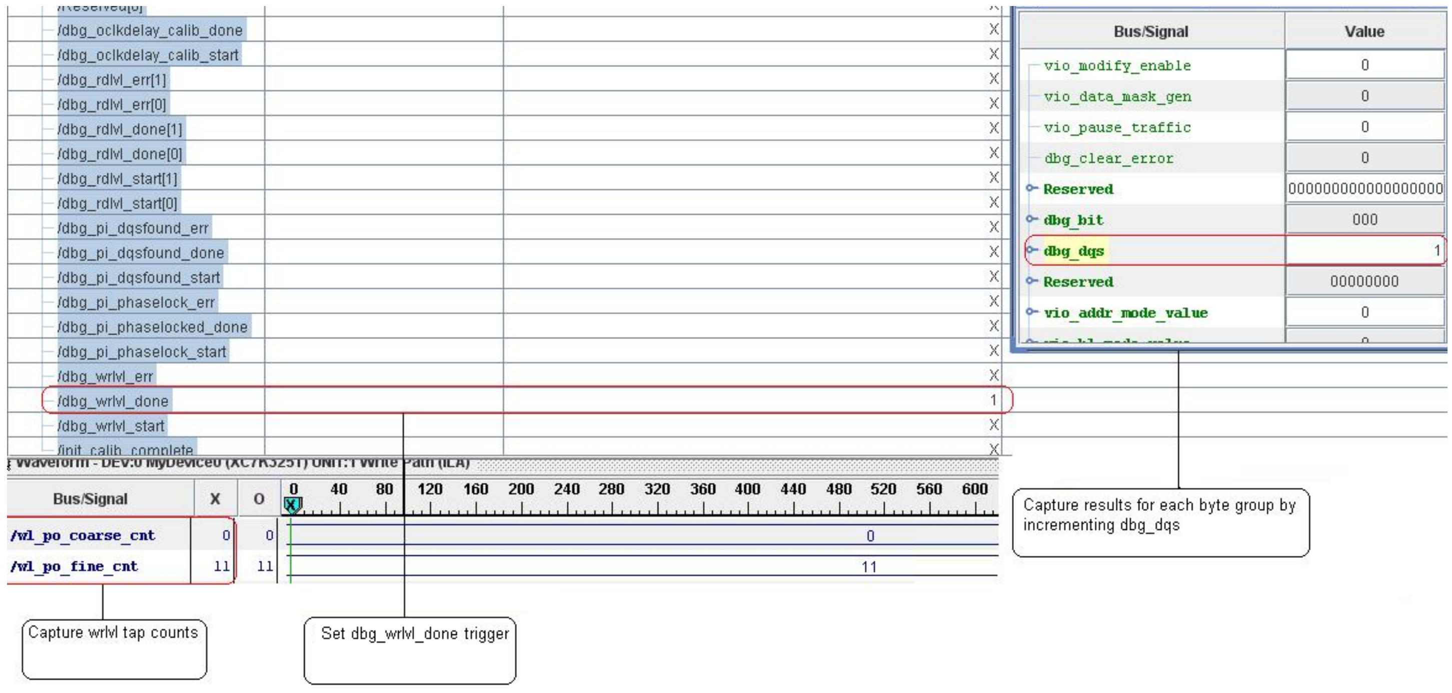
  NOTE: The tap variance across DQS byte groups will be quite different due to fly-by routing.

Table 4: Debug Signals of Interest for Write Leveling Calibration

Debug Signal	ILA/VIO Connection	Signal Description
dbg_wrlvl_start	ila_basic[1]	Signifies the start of the Write Leveling stage of calibration
dbg_wrlvl_done	ila_basic[2]	Signifies successful completion of the Write Leveling stage of calibration
dbg_wrlvl_err	ila_basic[3]	Signifies the Write Leveling stage of calibration exhibited errors and did not complete
wl_state_r	ila_wrpath[4:0]	State variable for the Write Leveling State Machine. States can be decoded in the ddr_phy_wrlvl.v module.
wrcal_dqs_cnt_r	ila_wrpath[9:6]	Signifies the DQS byte group being calibrated during Write Leveling. The algorithm will sequentially step through the DQS byte groups until write leveling completes successfully or a data byte group fails due a 0 to 1 transition not being detected on DQ.
wl_edge_detect_valid	ila_wrpath[10]	Signifies valid time Write Leveling algorithm is searching for edge
rd_data_edge_detect	ila_wrpath[11]	Signifies Write Leveling calibration found the 0-to-1 edge transition

wl_po_fine_cnt	ila_wrpath	Phaser_out Fine Taps found during Write Leveling. Byte capture based on VIO dbg_dqs setting.
wl_po_fine_cnt_0-8	ila_wrpath	Phaser_out Fine Taps found during Write Leveling
wl_po_coarse_cnt	ila_wrpath	Phaser_out Coarse Taps found during Write Leveling. Byte capture based on VIO dbg_dqs setting.
wrl_po_coarse_cnt_0-8	ila_wrpath	Phaser_out Coarse Taps found during Write Leveling

Expected ChipScope Tool Results



```
/dbg_oclkdelay_calib_done         X
/dbg_oclkdelay_calib_start        X
/dbg_rdlvl_err[1]                 X
/dbg_rdlvl_err[0]                 X
/dbg_rdlvl_done[1]                X
/dbg_rdlvl_done[0]                X
/dbg_rdlvl_start[1]               X
/dbg_rdlvl_start[0]               X
/dbg_pi_dqsfound_err              X
/dbg_pi_dqsfound_done             X
/dbg_pi_dqsfound_start            X
/dbg_pi_phaselock_err             X
/dbg_pi_phaselocked_done          X
/dbg_pi_phaselock_start           X
/dbg_wrlvl_err                    X
/dbg_wrlvl_done                   1
/dbg_wrlvl_start                  X
/init_calib_complete              X
```

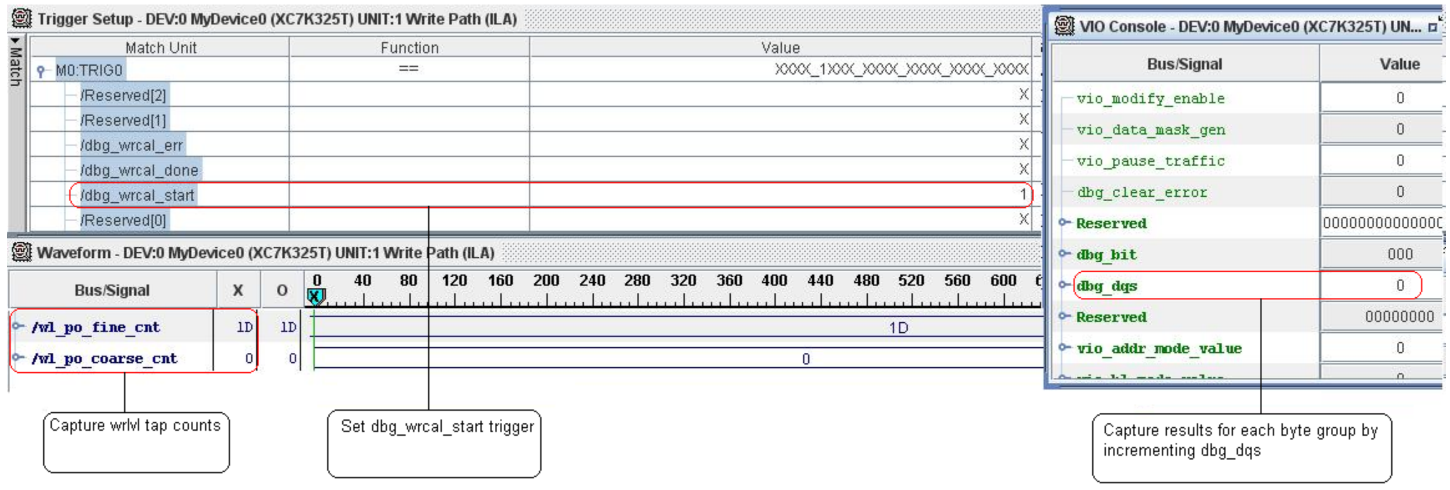
Bus/Signal	Value
vio_modify_enable	0
vio_data_mask_gen	0
vio_pause_traffic	0
dbg_clear_error	0
Reserved	000000000000000000
dbg_bit	000
dbg_dqs	1
Reserved	00000000
vio_addr_mode_value	0

Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)

Bus/Signal	X	O	0 … 600
/wl_po_coarse_cnt	0	0	0
/wl_po_fine_cnt	11	11	11

Capture results for each byte group by incrementing dbg_dqs
Capture wrlvl tap counts
Set dbg_wrlvl_done trigger

Figure 3 - Trigger = dbg_wrlvl_done


Capture wrlvl tap counts
Set dbg_wrcal_start trigger
Capture results for each byte group by incrementing dbg_dqs

Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)

Match Unit	Function	Value
M0:TRIG0	==	XXXX_1XXX_XXXX_XXXX_XXXX_XXXX
/Reserved[2]		X
/Reserved[1]		X
/dbg_wrcal_err		X
/dbg_wrcal_done		X
/dbg_wrcal_start		1
/Reserved[0]		X

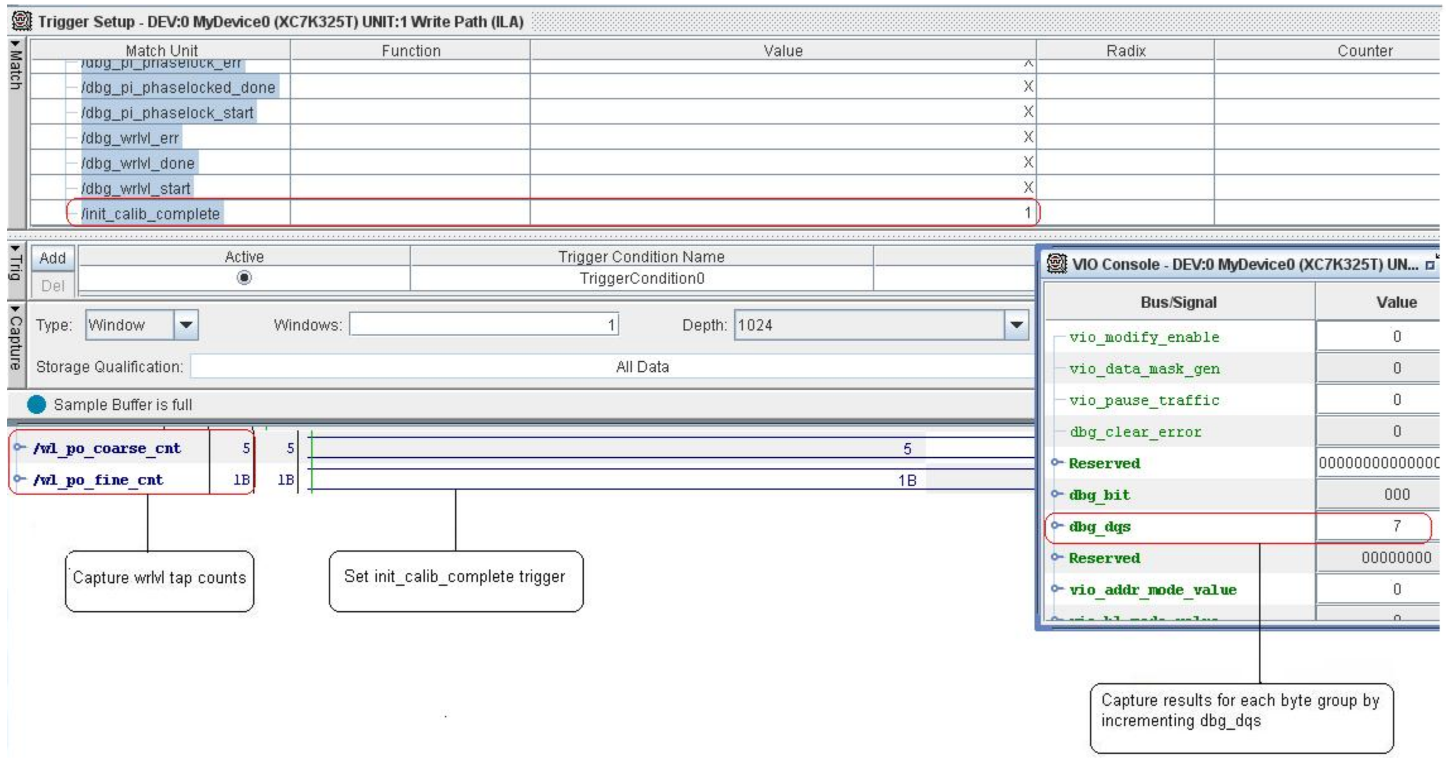
Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)

Bus/Signal	X	O	
/wl_po_fine_cnt	1D	1D	1D
/wl_po_coarse_cnt	0	0	0

VIO Console - DEV:0 MyDevice0 (XC7K325T) UN...

Bus/Signal	Value
vio_modify_enable	0
vio_data_mask_gen	0
vio_pause_traffic	0
dbg_clear_error	0
Reserved	0000000000000C
dbg_bit	000
dbg_dqs	0
Reserved	00000000
vio_addr_mode_value	0

Figure 4 - Trigger = dbg_wrcal_start


Capture wrlvl tap counts
Set init_calib_complete trigger
Capture results for each byte group by incrementing dbg_dqs

Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)

Match Unit	Function	Value	Radix	Counter
/dbg_pi_phaselock_err		X		
/dbg_pi_phaselocked_done		X		
/dbg_pi_phaselock_start		X		
/dbg_wrlvl_err		X		
/dbg_wrlvl_done		X		
/dbg_wrlvl_start		X		
/init_calib_complete		1		

Active	Trigger Condition Name
●	TriggerCondition0

Type: Window Windows: 1 Depth: 1024

Storage Qualification: All Data

Sample Buffer is full

Bus/Signal			
/wl_po_coarse_cnt	5	5	5
/wl_po_fine_cnt	1B	1B	1B

VIO Console - DEV:0 MyDevice0 (XC7K325T) UN...

Bus/Signal	Value
vio_modify_enable	0
vio_data_mask_gen	0
vio_pause_traffic	0
dbg_clear_error	0
Reserved	0000000000000C
dbg_bit	000
dbg_dqs	7
Reserved	00000000
vio_addr_mode_value	0

Figure 5 - Trigger = init_calib_complete

Debugging MPR Read Leveling Failures - DDR3 Only (dbg_rdlvl_done[1] does not assert)

Calibration Overview

At this stage of calibration, the write DQS is not centered in the write DQ window, nor is the read DQS centered in the read DQ window. The DDR3 Multi-Purpose Register (MPR) is used to center the read DQS in the read DQ window. The

MPR has a pre-defined “01010101” or “10101010” pattern that is read back during this stage of calibration. The read DQS centering is required for the next stage of calibration, OCLKDELAYED calibration.

Debug

- **If this stage of calibration failed with the assertion of dbg_rdlvl_err[1], set the ChipScope trigger to dbg_rdlvl_err[1].**
- **If this stage of calibration was successful and the results need to be analyzed, use the trigger dbg_rdlvl_done[1]=1.**
- Set the VIO dbg_dqs for each byte and capture the following signals; the results for each byte should be captured in the “7 Series DDR3 Calibration Results” spreadsheet. Later releases of MIG include results for all DQS byte groups removing the need to use dbg_dqs.

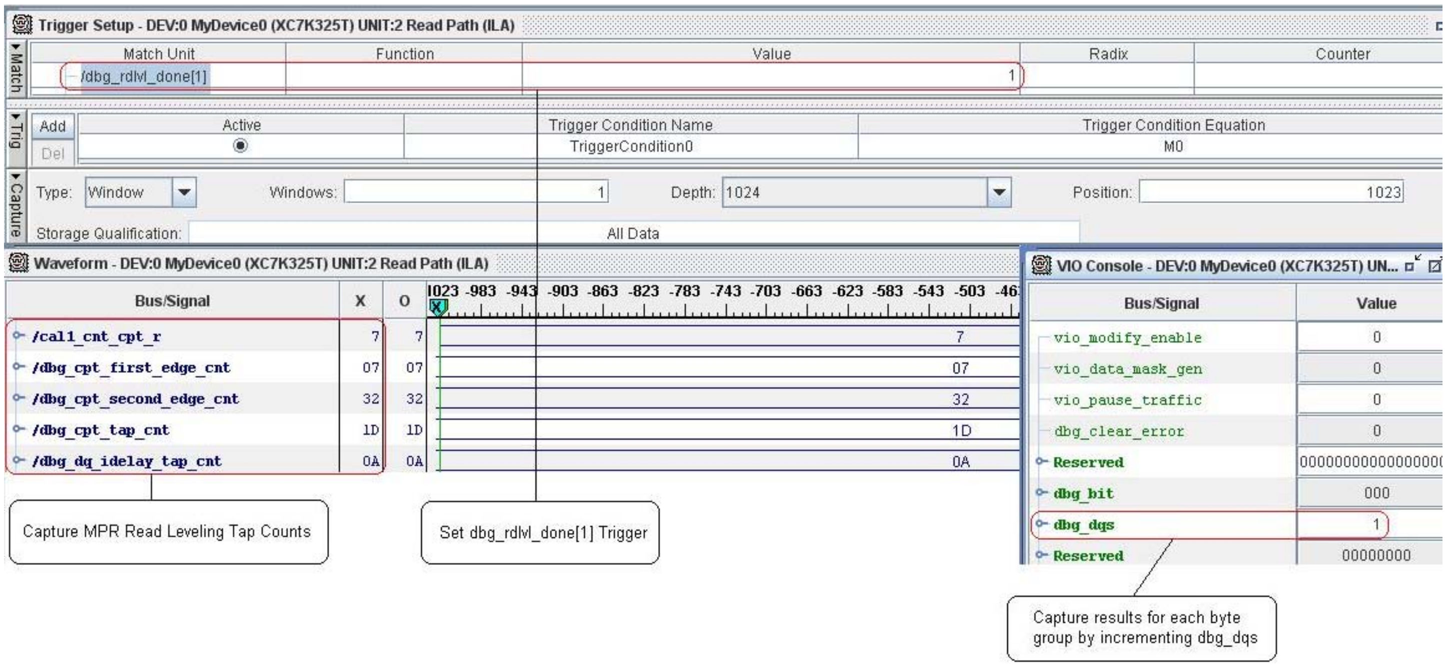
Table 5: Debug Signals of Interest for MPR Read Leveling Calibration

Debug Signal	ILA/VIO Connection	Signal Description
dbg_rdlvl_start[1]	ila_basic[11]	Signifies the start of the MPR stage of calibration
dbg_rdlvl_done[1]	ila_basic[13]	Signifies the successful completion of the MPR Stage of calibration
dbg_rdlvl_err[1]	ila_basic[15]	Signifies the MPR stage of calibration exhibited errors and did not complete
cal1_state_r	ila_rdpath[45:40]	State machine variable for MPR and Read Leveling Stage 1. States can be decoded in the ddr_phy_rdlvl.v module.
cal1_cnt_cpt_r	ila_rdpath[49:46]	Signifies the byte that failed MPR read leveling or read leveling stage 1
dbg_cpt_first_edge_cnt	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when the first edge in MPR and Read Leveling Stage 1 is found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_first_edge_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when the first edge in MPR and Read Leveling Stage 1 is found.
dbg_cpt_second_edge_cnt	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when then second edge in MPR and Read Leveling Stage 1 is found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_second_edge_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when then second edge in MPR and Read Leveling Stage 1 is found.
dbg_cpt_tap_cnt	ila_rdpath, per DQS and per RANK	Signifies the center tap moved to based on when the first and second edges were found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_tap_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies the center tap moved to based on when the first and second edges were found.
dbg_dq_idelay_tap_cnt	ila_rdpath, per DQS and per RANK	IDELAY tap value for MPR and Read Leveling Stage 1. This should be within 2-3 taps across all DQS byte groups. Byte capture based on VIO dbg_dqs setting.
dbg_dq_idelay_tap_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	IDELAY tap value for MPR and Read Leveling Stage 1. This should be within 2-3 taps across all DQS byte groups.

- Always look at DQ[0] for each component. Memory devices either send the “01010101” or “10101010” pattern on all DQ bits or on DQ[0] as specified by the JEDEC standard. The MIG design only looks at DQ[0]. If there is a problem with DQ[0], the MPR calibration stage would fail.
- If a DQS byte group failed this stage of calibration, cal1_cnt_cpt_r would equal the byte number that is failing as no further progress or increment on cal1_cnt_cpt_r occurred.

- Check if the failing DQS byte has an dq_idelay_tap_cnt value of 31. This means the algorithm ran out of taps searching for the capture edges.
- Check and compare the dq_idelay_tap_cnt, cpt_first_edge_cnt, cpt_second_edge_cnt, and cpt_tap_cnt values across bytes during MPR read leveling.
- Look at idelay_tap_cnt for each byte group. The idelay_tap_cnt across the DQS byte groups should only vary by 2-3 taps
- Look at how many edges (up to two) were found. Less than two edges may be found when running around or below 400MHz. Otherwise, two edges should always be found.
- Using high quality probes and scope, probe the address/command to ensure the load register command to the DRAM that enables MPR was correct. To enable the MPR, a MODE Register Set (MRS) command is issued to the MR3 Register with bit A2 =1. To make this measurement, bring mpr_rdlvl_start to an I/O pin and use as the trigger to capture A2 (must be ‘1’) and WE_n (must be ‘0’).

Expected ChipScope Tool Results


Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:2 Read Path (ILA)

Match Unit	Function	Value	Radix	Counter
/dbg_rdlvl_done[1]		1		

Active	Trigger Condition Name	Trigger Condition Equation
●	TriggerCondition0	M0

Type: Window Windows: 1 Depth: 1024 Position: 1023

Storage Qualification: All Data

Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:2 Read Path (ILA)

Bus/Signal	X	O	Waveform
/cal1_cnt_cpt_r	7	7	7
/dbg_cpt_first_edge_cnt	07	07	07
/dbg_cpt_second_edge_cnt	32	32	32
/dbg_cpt_tap_cnt	1D	1D	1D
/dbg_dq_idelay_tap_cnt	0A	0A	0A

VIO Console - DEV:0 MyDevice0 (XC7K325T) UN...

Bus/Signal	Value
vio_modify_enable	0
vio_data_mask_gen	0
vio_pause_traffic	0
dbg_clear_error	0
Reserved	0000000000000000
dbg_bit	000
dbg_dqs	1
Reserved	00000000

Add
Del
Match
Trig
Capture
1023 -983 -943 -903 -863 -823 -783 -743 -703 -663 -623 -583 -543 -503 -46
Capture MPR Read Leveling Tap Counts
Set dbg_rdlvl_done[1] Trigger
Capture results for each byte group by incrementing dbg_dqs

Figure 6 – Trigger = dbg_rdlvl_done[1]

Debugging OCLKDELAYED Calibration Failures

Calibration Overview

This stage of calibration centers the write DQS in the write DQ window. This centering is accomplished using the PHASER_OUT stage 3 delay line. The starting stage 3 tap value is 30. The taps are first decremented until either an edge is found or the tap value reaches 0. The stage 3 taps are then incremented back to 30 and edge detection begins increasing from 31 until either an edge is found, or the tap value reaches 63. The center point is then computed based on the detected edges and the stage 3 taps are decremented to the computed value.

Note that with every decrement of stage 3 tap the stage 2 taps are incremented by 2 to maintain the appropriate DQS to CK relationship established during write leveling. Similarly, with every increment of stage 3 tap, the stage 2 taps are decremented by 2. If stage 2 taps reach 0 or 63, stage 3 tap increment/decrement is allowed to proceed only 15 more times to avoid tDQSS violation. At the end of this stage of calibration, write leveling is re-performed to align DQS and CK using stage 2 taps.

Debug

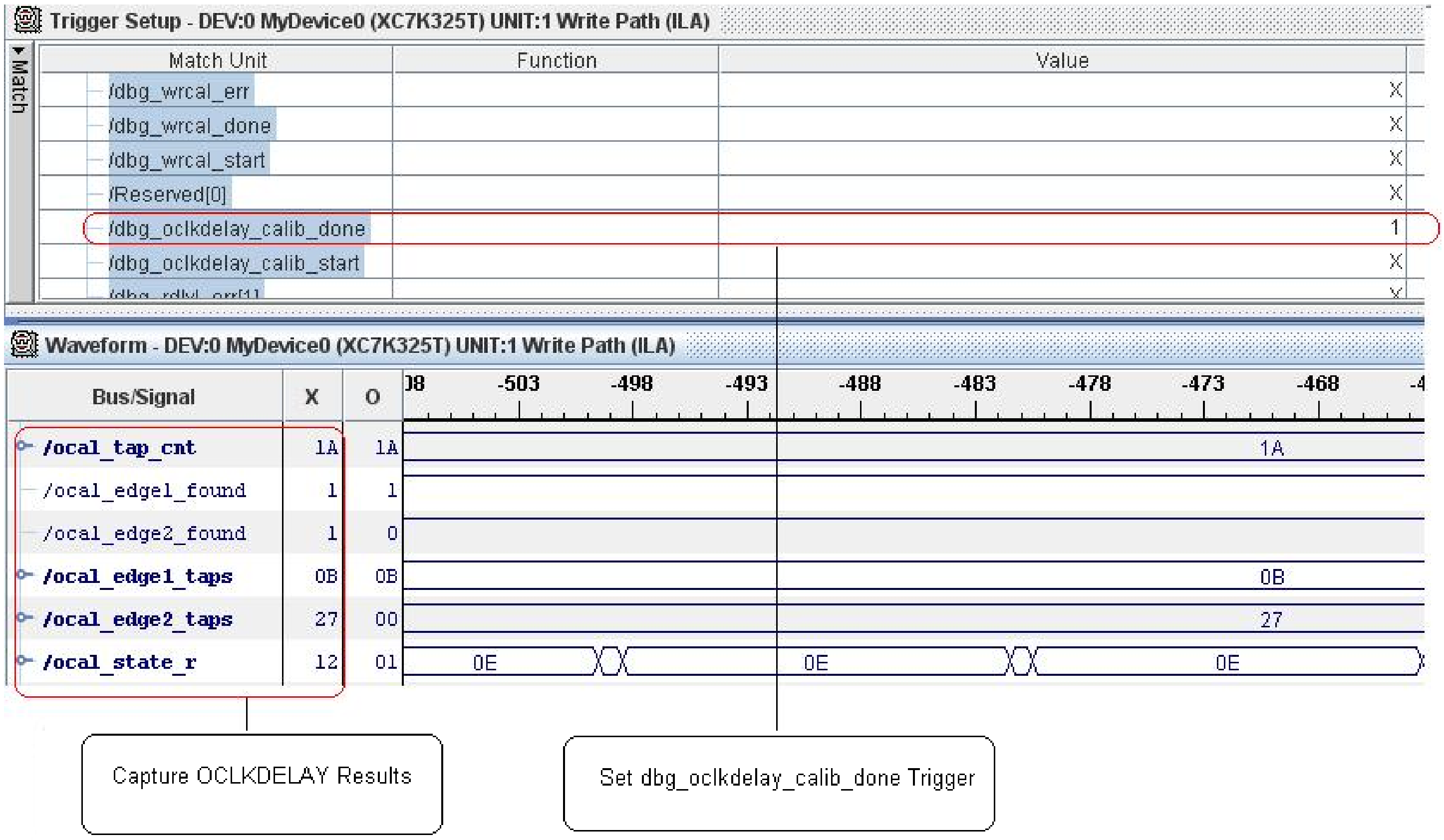
This stage of calibration will not fail. If no edges are detected (highly unlikely), the algorithm sets the tap to 30 and moves to the next calibration stage. Sub-optimal OCLKDELAYED calibration can result in data bit errors during normal operation. This occurs because the DQS to DQ 90 degree relationship is not correct. Full analysis of this calibration stage is critical.

- Probe the DQS to DQ phase relationship at the memory. DQS should be center aligned to DQ.
- Using **dbg_oclkdelay_calib_done=1 as the ChipScope trigger**, capture the below signals and record the results in the "7 Series DDR3 Calibration Results" spreadsheet.
- Look at the tap variance across byte lanes. It is expected to see a 5-6 tap difference.
- Look at how many edges (up to two) were found. Less than two edges many be found when running around or below 400MHz. Otherwise, two edges should always be found.

Table 6: Debug Signals of Interest for OCLKDELAYED Calibration

Debug Signal	ILA/VIO Connection	Signal Description
dbg_oclkdelay_calib_start	ila_basic[16]	Signifies the start of the OCLKDELAY stage of calibration
dbg_oclkdelay_calib_done	ila_basic[17]	Signifies successful completion of the OCLKDELAY stage of calibration
ocal_tap_cnt	ila_wrpath[37:32]	Tap used to move to the center of the window after finding first and second edges.
ocal_edge1_found	ila_wrpath[38]	Signifies that the first edge is found during OCLKDELAY calibration when the taps are decremented from 30
ocal_edge2_found	ila_wrpath[39]	Signifies that the second edge is found during OCLKDELAY calibration when the taps are incremented from 30
ocal_edge1_taps	ila_wrpath[45:40]	Tap value when the first edge is found during OCLKDELAY calibration
ocal_edge2_taps	ila_wrpath[51:46]	Tap value when the second edge is found during OCLKDELAY calibration
ocal_state	ila_wrpath[56:52]	State variable for the OCLKDELAY state machine. States can be decoded in the ddr_phy_oclkdelay_cal module
stg2_tap_cnt	ila_wrpath	Final Phaser_OUT Stage 3 tap count found during OCLKDELAYED Calibration.

Expected ChipScope Tool Results


Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)

Match Unit	Function	Value
/dbg_wrcal_err		X
/dbg_wrcal_done		X
/dbg_wrcal_start		X
/Reserved[0]		X
/dbg_oclkdelay_calib_done		1
/dbg_oclkdelay_calib_start		X

Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)

Bus/Signal	X	O
/ocal_tap_cnt	1A	1A
/ocal_edge1_found	1	1
/ocal_edge2_found	1	0
/ocal_edge1_taps	0B	0B
/ocal_edge2_taps	27	00
/ocal_state_r	12	01

-503
-498
-493
-488
-483
-478
-473
-468
1A
0B
27
0E
0E
0E
Capture OCLKDELAY Results
Set dbg_oclkdelay_calib_done Trigger

Figure 7 – Trigger = dbg_oclkdelay_calib_done

Debugging Write Calibration Failures

Calibration Overview

Write calibration is required to align DQS to the correct CK edge. During write leveling, DQS is aligned to the nearest rising edge of CK. However, this might not be the edge that captures the write command.

Depending on the interface type (UDIMM, RDIMM, or component), the DQS could either be one CK cycle earlier than, two CK cycles earlier than, or aligned to the CK edge that captures the write command.

This is a pattern based calibration; hence, multiple writes followed by a single read are issued during this stage. The following data patterns might be seen:

- On time write pattern read back: FF00AA5555AA9966
- One CK early write pattern read back: AA5555AA9966BB11
- Two CK early write pattern read back: 55AA9966BB11EE44
- One CK late write pattern read back: XXXXFF00AA5555AA
  - Calibration cannot correct for this pattern.
    This pattern indicates that the trace delays are incorrect where CK is incorrectly shorter than DQS.

If none of the above patterns are detected during reads, the algorithm assumes the MPR read leveling IDELAY settings are incorrect and the IDELAYs for the DQ bits associated with that byte are set to 0. MPR read leveling could have an incorrect IDELAY setting because with the “01010101” or “10101010” pattern, it is not possible to differentiate between clock cycles.

Debug

If dbg_wrcal_err asserted, denoting a Write Calibration failure, use **dbg_wrcal_err=1 as the trigger** and observe the following debug signals. If dbg_wrcal_done asserted but the results of this stage need to be analyzed, use **dbg_wrcal_done as the trigger**.

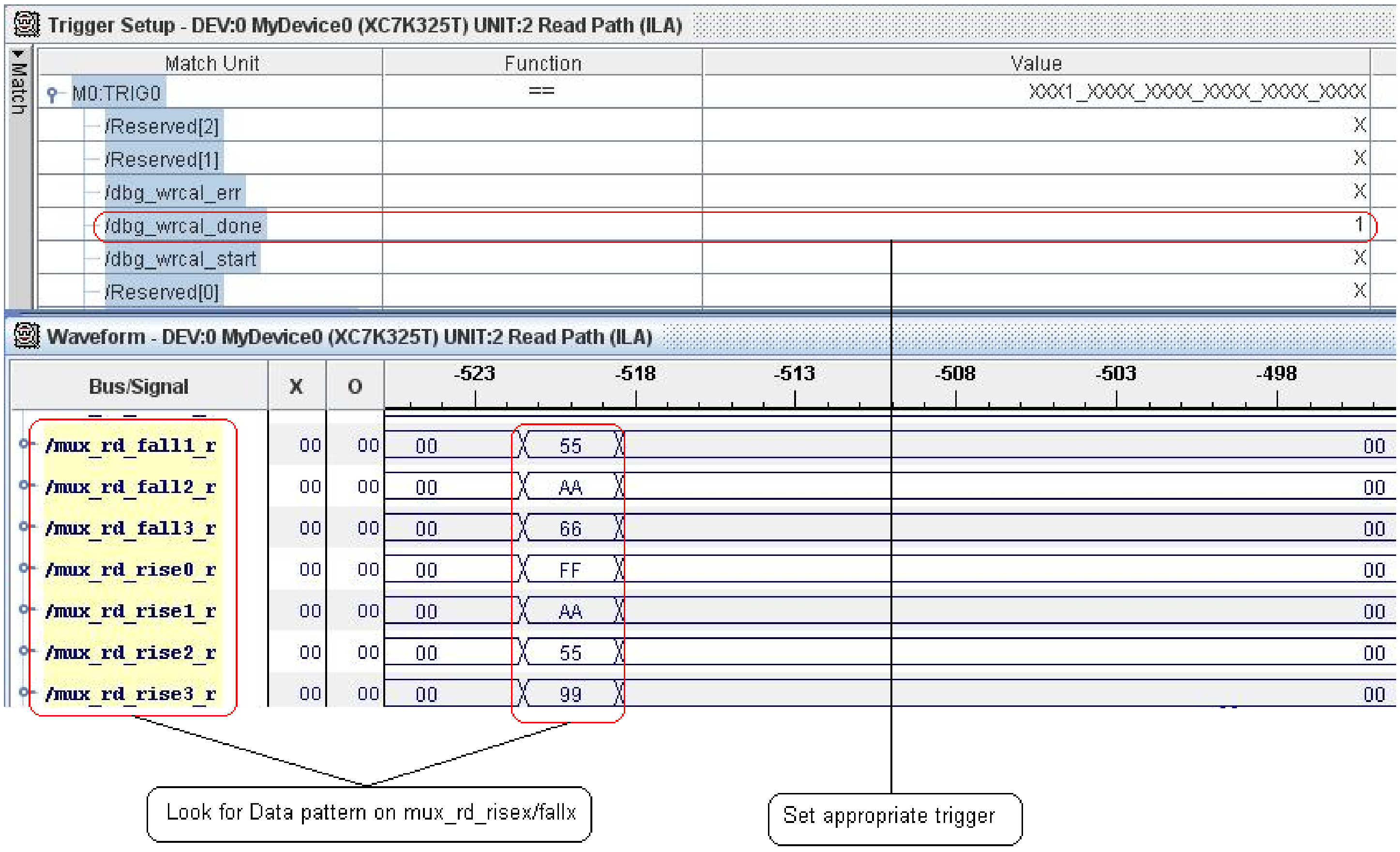
Table 7: Debug Signals of Interest for Write Calibration

Debug Signal	ILA/VIO Connection	Signal Description
dbg_wrcal_start	ila_basic[19]	Signifies the start of the Write Calibration stage of calibration
dbg_wrcal_done	ila_basic[20]	Signifies successful completion of the Write Calibration stage of calibration
dbg_wrcal_err	ila_basic[21]	Signifies Write Calibration exhibited errors and did not complete
pat_data_match	ila_wrpath[64]	Asserts when the valid pattern is detected
pat_data_match_valid	ila_wrpath[65]	Toggles when the correct pattern is detected
wrcal_dqs_cnt	ila_wrpath[69:66]	Current DQS group being calibrated in Write Calibration. When dbg_wrcal_start asserts, wrcal_dqs_cnt is 0. The algorithm sequentially steps through the DQS byte groups checking to see if the read data pattern matches the expected FF00AA5555AA9966 pattern. If the pattern matches, wrcal_dqs_cnt increments by 1. The algorithm then starts looking for the correct data pattern on the next byte until it reaches DQS_WIDTH-1 or a data byte group fails due to the data pattern not being detected properly. The wrcal_dqs_cnt stays at DQS_WIDTH-1 after dbg_wrcal_done signal is asserted.
cal2_state	ila_wrpath[74:70]	Write Calibration state machine variable
not_empty_wait_cnt	ila_wrpath[79:75]	Count value during write calibration pattern detection. Maximum count is 'd31. If count reaches d'31, write calibration will fail with the assertion of dbg_wrcal_err.
early1_data	ila_wrpath[80]	Asserts when the pattern detected is one CK clock cycle early. When this is asserted, the write leveling algorithm moves the CK clock one cycle. After CK is moved, the write calibration algorithm restarts pattern detection.
early2_data	ila_wrpath[81]	Asserts when the pattern detected is two CK clock cycles early. When this is asserted, the write leveling algorithm moves the CK clock two cycles. After CK is moved, the write calibration algorithm restarts pattern detection.
mux_rd_rise0	ila_rdpath / ila_wrpath	Data pattern received on rising edge 0.
mux_rd_fall0	ila_rdpath / ila_wrpath	Data pattern received on falling edge 0.
mux_rd_rise1	ila_rdpath / ila_wrpath	Data pattern received on rising edge 1.
mux_rd_fall1	ila_rdpath / ila_wrpath	Data pattern received on falling edge 1.
mux_rd_rise2	ila_rdpath / ila_wrpath	Data pattern received on rising edge 2.

mux_rd_fall2	ila_rdpath / ila_wrpath	Data pattern received on falling edge 2.
mux_rd_rise3	ila_rdpath / ila_wrpath	Data pattern received on rising edge 3.
mux_rd_fall3	ila_rdpath / ila_wrpath	Data pattern received on falling edge 3.

1. The number on wrcal_dqs_cnt when dbg_wrcal_err asserts signifies the byte that failed write calibration. Debug should be focused on this byte group.
2. Observe the rddata bus or the mux_rd_fall/riseX_r buses and look for the appropriate data pattern. Note, mux_rd_fall/rise_2/3_r will not be used with the half-rate controller and will always be 0. Again, the three scenarios that allow write calibration to continue are:
   - On-time write expected pattern – FF00AA5555AA9966
   - One cycle Early write expected pattern – AA5555AA9966BB11
   - Two cycles Early expected pattern – 55AA9966BB11EE44
3. If none of these three patterns are observed on a failing byte, look at the failing pattern and determine how the pattern is failing. Look if there are failing DQ bit(s) within a byte, failing bytes, etc. If the late write pattern noted above was detected, there is most likely a trace length issue between DQS and CK where CK is not longer than DQS as required.
4. If the design is stuck in the Write Calibration stage, the problem could be related to either the write or the read. **Determining whether the write or read is causing the failure is critical**. The following steps should be completed using dbg_wrcal_start as the scope trigger. To do this dbg_wrcal_start must be brought out to an I/O. For additional details and example Read and Write scope shots, review the below "Determining if a Data Error is due to the Write or Read" section.
   a. To ensure the writes are correct, observe the write DQS to write DQ relationship at the memory using high quality scope and probes. During write calibration, a write is followed by a read so care needs to be taken to ensure the write is captured. See the "Determining if a Data Error is due to the Write or Read" section for details. If there is a failing bit, determining the write DQS to write DQ relationship for the specific DQ bit is critical. The write will ideally have the DQS center aligned in the DQ window. Misalignment between DQS and DQ during Write Calibration points to a problem with OCLKDELAY calibration. Please review the "Debugging OCLKDELAY Calibration Failures" section.
   b. If the DQ-DQS alignment looks correct, next observe the WE_n to DQS relationship at the memory during a write again using high quality scope and probes. The WE_n to DQS delay must equal the CAS Write Latency (CWL).
   c. Using high quality scope and probes, verify the expected pattern (FF00AA5555AA9966) is being written to the DRAM during a write and that the expected pattern is being read back during the first Write Calibration read. If the pattern is correct during write and read at the DRAM, verify the DQS-CK alignment. During Write Calibration, these two signals should be aligned. Write Leveling aligned these two signals which has successfully completed before Write Calibration.
   d. Probe ODT and WE_n during a write command. In order for ODT to be properly turned on in the memory, ODT must assert before the write command.
   e. Probe DM to ensure it is held low during calibration. If a board issue exists causing DM to improperly assert, incorrect data will be read back during calibration causing a write calibration failure. An example of a board issue on DM is when DM is not used and tied low at the memory with improper termination.
5. It is possible for write calibration to fail due to rare manufacturing issues with the memory device. Verify SDRAM pins are behaving correctly. Look for floating or grounded signals. The debug signals should be used to determine which byte group is failing and if specific pin(s) within that byte group are causing the incorrect data pattern. These pins should be the focus at the memory device.
6. If the DQS-to-DQ, CWL, and DQS-to-CK look correct, review the above "Debugging MPR Read Leveling Failures" section.

Expected ChipScope Tool Results



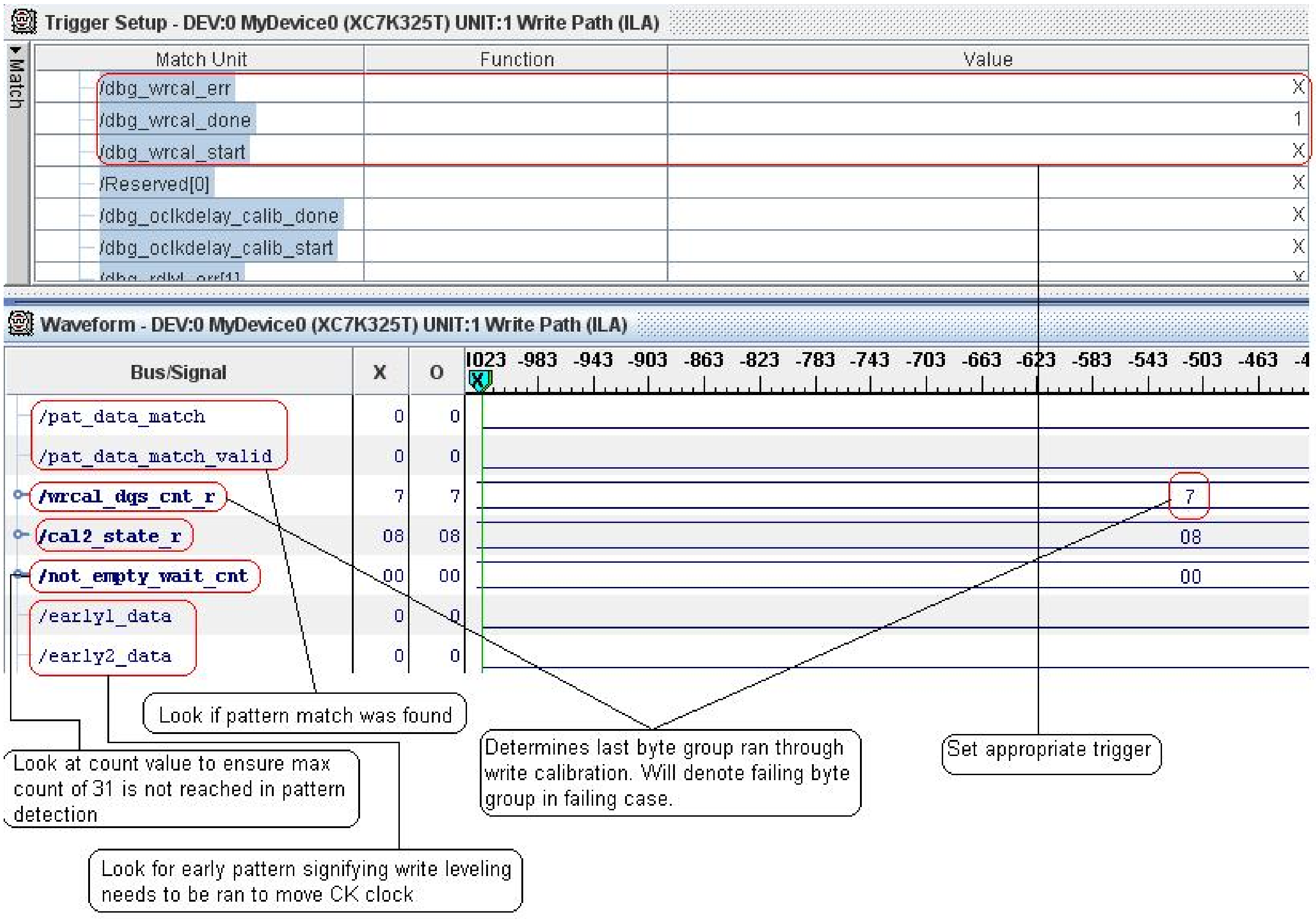
Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:2 Read Path (ILA)

Match Unit	Function	Value
M0:TRIG0	==	XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
/Reserved[2]		X
/Reserved[1]		X
/dbg_wrcal_err		X
/dbg_wrcal_done		1
/dbg_wrcal_start		X
/Reserved[0]		X

Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:2 Read Path (ILA)

Bus/Signal	X	O	Waveform		
/mux_rd_fall1_r	00	00	00	55	00
/mux_rd_fall2_r	00	00	00	AA	00
/mux_rd_fall3_r	00	00	00	66	00
/mux_rd_rise0_r	00	00	00	FF	00
/mux_rd_rise1_r	00	00	00	AA	00
/mux_rd_rise2_r	00	00	00	55	00
/mux_rd_rise3_r	00	00	00	99	00

Figure 8 – Trigger = dbg_wrcal_done



Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)
Match
Match Unit
Function
Value
/dbg_wrcal_err
X
/dbg_wrcal_done
1
/dbg_wrcal_start
X
/Reserved[0]
X
/dbg_oclkdelay_calib_done
X
/dbg_oclkdelay_calib_start
X
Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:1 Write Path (ILA)
Bus/Signal
X
O
/pat_data_match
/pat_data_match_valid
/wrcal_dqs_cnt_r
/cal2_state_r
/not_empty_wait_cnt
/early1_data
/early2_data
7
08
00
Look if pattern match was found
Look at count value to ensure max count of 31 is not reached in pattern detection
Determines last byte group ran through write calibration. Will denote failing byte group in failing case.
Set appropriate trigger
Look for early pattern signifying write leveling needs to be ran to move CK clock

Figure 9 – Trigger = dbg_wrcal_done

Debugging Read Leveling Failures

For memory clock frequencies of 400 MHz and above, Read Leveling is performed after Write Calibration.

Calibration Overview

The final read DQS to read DQ centering is done in this stage of calibration. The first step in this stage is to decrement the IDELAY and PHASER_IN stage 2 taps values to zero to undo MPR read leveling. MPR read leveling was only required for OCLKDELAYED calibration. This stage of read leveling accurately centers the read DQS in the read DQ window using a 993377EECC992244 data pattern. If this stage calibrates successfully, the init_calib_complete signal is asserted and calibration is complete.

Debug

- If this stage of calibration failed with the assertion of dbg_rdlvl_err[0], set the **ChipScope trigger to dbg_rdlvl_err[0].**
- If this stage of calibration was successful and the results need to be analyzed, set the **ChipScope trigger to dbg_rdlvl_done[0]=1.**
- Set the VIO dbg_dqs for each byte and capture the following signals. The results for each byte should be captured in the "7 Series DDR3 Calibration Results" spreadsheet. Later releases of MIG include results for all DQS byte groups removing the need to use dbg_dqs.

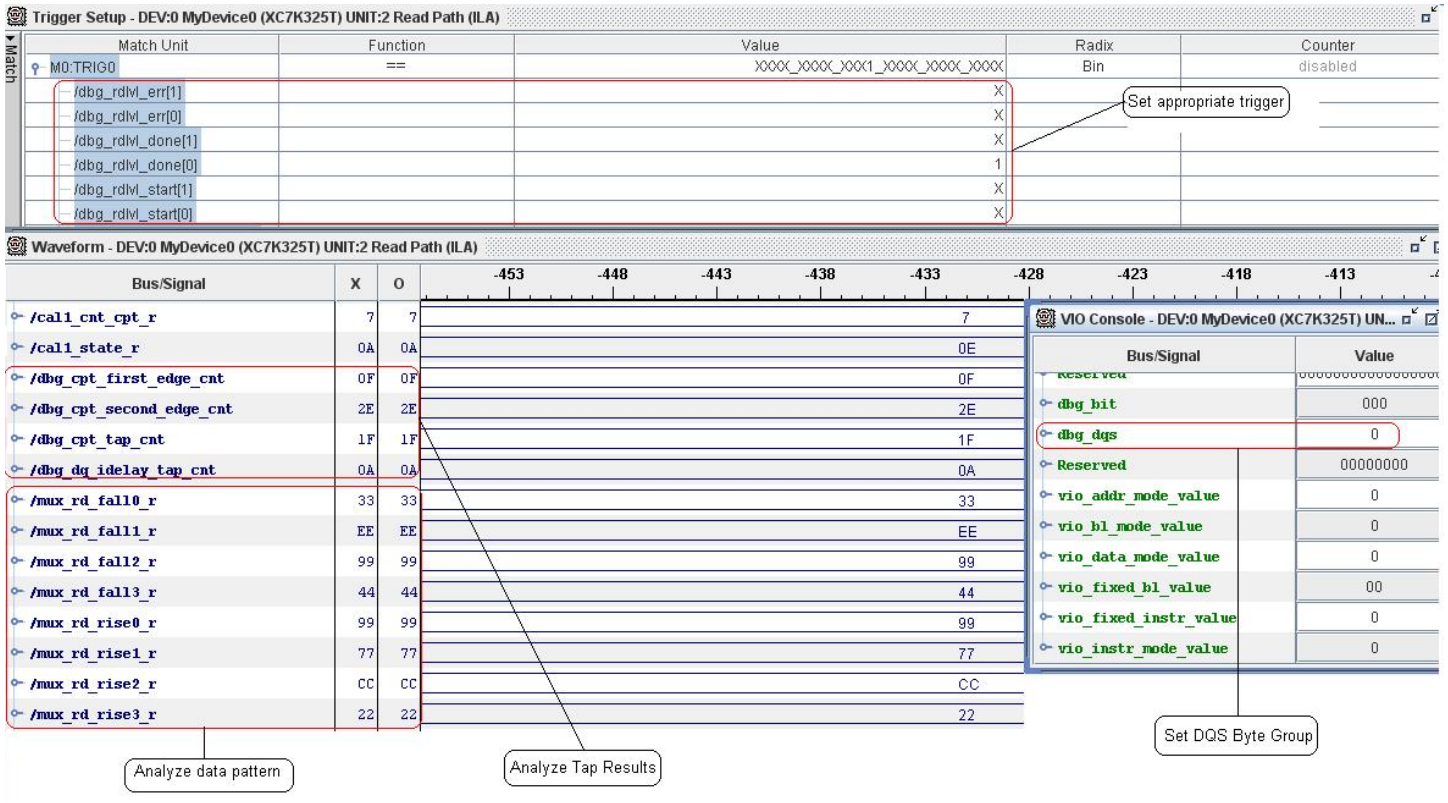
Table 8: Debug Signals of Interest for Read Leveling Stage 1 Calibration

Debug Signal	ILA/VIO Connection	Signal Description
dbg_rdlvl_start[0]	ila_basic[10]	Signifies the start of Read Leveling Stage 1 of calibration
dbg_rdlvl_done[0]	ila_basic[12]	Signifies the successful completion of Read Leveling Stage 1 of calibration
dbg_rdlvl_err[0]	ila_basic[14]	Signifies Read Leveling Stage 1 of calibration exhibited errors and did not complete
cal1_state_r	ila_rdpath[45:40]	State machine variable for MPR and Read Leveling Stage 1. States can be decoded in the ddr_phy_rdlvl.v module.
cal1_cnt_cpt_r	ila_rdpath[49:46]	Signifies the byte that failed MPR read leveling or read leveling stage 1
dbg_cpt_first_edge_cnt	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when the first edge in MPR and Read Leveling Stage 1 is found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_first_edge_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when the first edge in MPR and Read Leveling Stage 1 is found.
dbg_cpt_second_edge_cnt	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when then second edge in MPR and Read Leveling Stage 1 is found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_second_edge_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies PHASER_IN fine tap count when then second edge in MPR and Read Leveling Stage 1 is found.
dbg_cpt_tap_cnt	ila_rdpath, per DQS and per RANK	Signifies the center tap moved to based on when the first and second edges were found. Byte capture based on VIO dbg_dqs setting.
dbg_cpt_tap_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	Signifies the center tap moved to based on when the first and second edges were found.

dbg_dq_idelay_tap_cnt	ila_rdpath, per DQS and per RANK	IDELAY tap value for MPR and Read Leveling Stage 1. This should be within 2-3 taps across all DQS byte groups. Byte capture based on VIO dbg_dqs setting.
dbg_dq_idelay_tap_cnt_rnk0/1_byte0-8	ila_rdpath, per DQS and per RANK	IDELAY tap value for MPR and Read Leveling Stage 1. This should be within 2-3 taps across all DQS byte groups.

- Determine which stage is failing by observing cal1_state_r.
- Look at idelay_tap_cnt for each byte group. The idelay_tap_cnt across the DQS byte groups should only vary by 2-3 taps
- Look at how many edges (up to two) were found. Less than two edges may be found when running around or below 400MHz. Otherwise, two edges should always be found to then center the IDELAY taps.
- Determine if any bytes completed successfully. The read leveling algorithm will sequentially step through each DQS byte group detecting the capture edges. When the failure occurs, the value on cal1_cnt_cpt_r indicates the byte that failed edge detection.
- If the incorrect data pattern is detected, determine if the error is due to the write access or the read access. See the "Determining if a Data Error is due to the Write or Read" section below.
- If the dbg_rdlvl_err[0] is asserted (read leveling failure), use high quality probes and scope observe the DQS-to-DQ phase relationship during a write. The scope trigger should be dbg_rdlvl_start[0]. The alignment should be approximately 90 degrees.
- If the DQS-to-DQ alignment is correct, observe the we_n-to-DQS relationship to see if it meets CWL again using dbg_rdlvl_start[0] as a trigger.

Expected ChipScope Tool Results


Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:2 Read Path (ILA)

Match Unit	Function	Value	Radix	Counter
M0:TRIG0	==	XXXX_XXXX_XXX1_XXXX_XXXX_XXXX	Bin	disabled
/dbg_rdlvl_err[1]		X		
/dbg_rdlvl_err[0]		X		
/dbg_rdlvl_done[1]		X		
/dbg_rdlvl_done[0]		1		
/dbg_rdlvl_start[1]		X		
/dbg_rdlvl_start[0]		X		

Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:2 Read Path (ILA)

Bus/Signal	X	O	Waveform
/cal1_cnt_cpt_r	7	7	7
/cal1_state_r	0A	0A	0E
/dbg_cpt_first_edge_cnt	0F	0F	0F
/dbg_cpt_second_edge_cnt	2E	2E	2E
/dbg_cpt_tap_cnt	1F	1F	1F
/dbg_dq_idelay_tap_cnt	0A	0A	0A
/mux_rd_fall0_r	33	33	33
/mux_rd_fall1_r	EE	EE	EE
/mux_rd_fall2_r	99	99	99
/mux_rd_fall3_r	44	44	44
/mux_rd_rise0_r	99	99	99
/mux_rd_rise1_r	77	77	77
/mux_rd_rise2_r	CC	CC	CC
/mux_rd_rise3_r	22	22	22

VIO Console - DEV:0 MyDevice0 (XC7K325T) UN...

Bus/Signal	Value
Reserved	000000000000000000
dbg_bit	000
dbg_dqs	0
Reserved	00000000
vio_addr_mode_value	0
vio_bl_mode_value	0
vio_data_mode_value	0
vio_fixed_bl_value	00
vio_fixed_instr_value	0
vio_instr_mode_value	0

Figure 10 – Trigger = dbg_rdlvl_done[0]

Debugging PRBS Read Leveling Failures

Calibration Overview

This stage of calibration was added in MIG 7 Series 1.7 and determines the read data valid window using a 128 long PRBS sequence (generated through 64-bit LFSR logic) that is written once and read back continuously from the DDR3 SDRAM. The algorithm starts at the DQS PHASER_IN fine tap setting determined during the Read Leveling calibration stage (initial tap value) and decrements one tap at time until a data mismatch is found when comparing read data with the expected data. Note that the expected data is generated using the same 64-bit LFSR logic that was used to write the 128 long PRBS sequence to the SDRAM. The data mismatch tap value is recorded as the left edge. The algorithm then increments to the initial tap value and edge detection begins with every increment after the initial tap value until a data mismatch is found or the tap value is 63. The algorithm then computes the center of the read data valid window based on the detected edges.

Debug

PRBS Read Leveling debug signals are not currently added into the ChipScope debug cores. If rdlvl_done[0] asserts (signifying the previous calibration stage, Read Leveling Stage 1, has completed successfully) but init_calib_done does not assert, PRBS Read Leveling has most likely failed. To confirm, manually add the “prbs_rdlvl_start” and “prbs_rdlvl_done” signals into ChipScope from the “user_design/rtl/phy/mig_7series_v1_x_ddr_phy_prbs_rdlvl.v” module. If this stage of calibration fails, please open a WebCase for assistance debugging the error.

Calibration Times

- For IES with extended calibration, completing calibration in hardware should take about 30 seconds.
- For GES, completing calibration in hardware should take about 1 second.

Debugging Data Errors

General Checks

As with calibration error debug, the General Checks section of this answer record should be reviewed. Strict adherence to proper board design is critical in working with high speed memory interfaces. Violation of these general checks is often the root cause of data errors.

Replicating the Error Using the Traffic Generator

When data errors are seen during normal operation, the MIG 7 Series Example Design (Traffic Generator) should be used to replicate the error. The Traffic Generator can be configured to send a wide range of data, address, and command patterns allowing customers to test their target traffic pattern on a verified solution. The Traffic Generator stores the write data and compares it to the read data. This allows comparison of expected and actual data when errors occur. This is a critical step in Data Error debug as this section will go through in detail.

Table 9: Debug Signals used for configuring the Traffic Generator:

Debug Signal	ILA/VIO Connection	Signal Description
vio_modify_enable	ddr3_vio_sync_out[36]	Set to 1 to vary the command Traffic Generator command pattern.
vio_data_mask_gen	ddr3_vio_sync_out[58]	Traffic generator Data Mask generation

vio_pause_traffic	ddr3_vio_sync_out[59]	Set to 1 to pause the Traffic Generator.
dbg_clear_error	ddr3_vio_sync_out[63]	Set to clear Traffic Generator errors. This signal can be used in checking for single bit errors or measuring a read window.
vio_addr_mode_value	ddr3_vio_sync_out[43:41]	Valid settings for this signal are: • 0x1: FIXED address mode[1] • 0x2: PRBS address mode • 0x3: SEQUENTIAL address mode
vio_bl_mode_value	ddr3_vio_sync_out[49:48]	Valid settings for this signal are: • 0x1: FIXED burst length[1] • 0x2: PRBS burst length
vio_fixed_bl_value	ddr3_vio_sync_out[57:50];	Valid settings are 1 to 256
vio_fixed_instr_value	ddr3_vio_sync_out[62:60];	Valid settings are: • 0x0: Write instruction • 0x1: Read instruction
vio_instr_mode_value	ddr3_vio_sync_out[47:44];	Valid settings for this signal are: • 0x1: Command type (read/write) as defined by fixed_instr_i[1] • 0x2: Random read/write commands • 0xE: Write only at address zero • 0xF: Read only at address zero
vio_data_mode_value	ddr3_vio_sync_out[40:37];	Valid settings for this signal are: • 0x0: Reserved. • 0x1: FIXED - 32 bits of fixed_data as defined through fixed_data_i inputs.[1] • 0x2: ADDRESS - 32 bits address as data. Data is generated based on the logical address space. If a design has a 256-bit user data bus, each write beat in the user bus would have a 256/8 address increment in byte boundary. If the starting address is 1300, the data is 1300, followed by 1320 in the next cycle. To simplify the logic, the user data pattern is a repeat of the increment of the address value bit[31:0].

		• 0x3: HAMMER - All 1s are on DQ pins during the rising edge of DQS, and all 0s are on the DQ pins during the falling edge of DQS, except the VICTIM line as defined in the parameter "SEL_VICTIM_LINE". This option is only valid if parameter DATA_PATTERN = "DGEN_HAMMER" or "DGEN_ALL". • 0x4: SIMPLE8 - Simple 8 data pattern that repeats every 8 words. The patterns can be defined by the "simple_datax" inputs.[1] • 0x5: WALKING1s - Walking 1s are on the DQ pins. The starting position of 1 depends on the address value. This option is only valid if the parameter DATA_PATTERN = "DGEN_WALKING" or "DGEN_ALL". • 0x6: WALKING0s - Walking 0s are on the DQ pins. The starting position of 0 depends on the address value. This option is only valid if the parameter DATA_PATTERN = "DGEN_WALKING0" or "DGEN_ALL". • 0x7: PRBS - A 32-stage LFSR generates random data and is seeded by the starting address. This option is only valid if the parameter DATA_PATTERN = "DGEN_PRBS" or "DGEN_ALL". • 0x9: SLOW HAMMER - This is the slow MHz hammer data pattern. • 0xF: PHY_CALIB pattern - 0xFF, 00, AA, 55, 55, AA, 99, 66. This mode only generates READ commands at address zero. This is only valid in the Virtex®-7 family.

Notes:

1. This setting does not work by default and additional RTL modifications are required.

Table 10: Debug Signals of Interest when isolating the data error using the Traffic Generator:

Debug Signal	ILA/VIO Connection	Signal Description
dbg_rddata_r	ddr3_ila_basic[92:64]	Read data read out of the IN_FIFO for the DQS group selected through dbg_dqs on the VIO. This is a 64-bit bus. For an example of how the data is captured, see the Debug Read Data table below. This debug port will not capture ECC data.
cmp_data_r	ddr3_ila_basic_w[131+:64]	Expected data to be compared with read back data from memory[1]

dbg_rddata_valid	ddr3_ila_basic_w[28]	Signifies that the read data is valid
cmp_data_valid	ddr3_ila_basic_w[129]	Signifies the compare data is valid
cmp_error	ddr3_ila_basic_w[130]	Signifies the cmp_data is not the same as the readback data from memory
error_status[n:0]	ddr3_ila_basic_w	This signal latches these values when the error signal is asserted: • [31:0]: cmp_addr_i • [37:32]: cmp_bl_i [40]: mcb_cmd_full • [41]: mcb_wr_full • [42]: mcb_rd_empty

Notes:
1. Cmp_data_r is not cycle aligned with dbg_rddata_r and may vary from 1 burst before to 3 bursts after dbg_rddata_r.

Isolating the Data Error

Using either the MIG 7 Series Traffic Generator or the user design, the first step in data error debug is to isolate when and where the data errors occur. In order to do this, the expected data and actual data must be known and compared. Looking at the data errors, the following should be identified:

- Are the errors bit or byte errors?
  - Are errors seen on data bits belonging to certain DQS groups?
  - Are errors seen on specific DQ bits?
- Is the data shifted, garbage, swapped, etc?
- Are errors seen on accesses to certain addresses, banks, or ranks of memory?
  - Designs that can support multiple varieties of DIMM modules, all possible address and bank bit combinations should be supported.
- Do the errors only occur for certain data patterns or sequences?
  - This can indicate a shorted or open connection on the PCB. It can also indicate an SSO or crosstalk issue.
- Determine the frequency and reproducibility of the error
  - Does the error occur on every calibration/reset?
  - Does the error occur at specific temperature or voltage conditions?
- Determine if the error is correctable
  - Rewriting, rereading, resetting, recalibrating.

To isolate the data error using the MIG 7 Series Example Design Traffic Generator, use the following steps:

- Determine what type of data error is being seen (bit or byte errors).
  1. **Set the ChipScope trigger to cmp_error=1**
  2. Observe the "**dbg_rddata_r**" and "**cmp_data_r**" signals in ChipScope
     - Are errors seen on a data bit/s belonging to a certain DQS group/s?
     - Does the data appear shifted, garbage, swapped, etc.?
- Determine if errors are seen on accesses to a certain address, bank, or rank of the memory.
  1. **Set the ChipScope trigger to cmp_error=1**
  2. Set the ChipScope VIO's
     vio_modify_enable=1

vio_instr_mode_value=2
vio_data_mode_value=2
and **vio_addr_mode_value=3**

3. Observe the "**cmp_addr_i**" bits of the **error_status[31:0]** in ChipScope

- Determine if errors only occur for certain data patterns or sequences. This can indicate a shorted or open connection on the PCCB or can also indicate an SSO or crosstalk issue.
    1. **Set the ChipScope trigger to cmp_error=1**
    2. Set the ChipScope VIOs
       vio_modify_enable=1
       vio_instr_mode_value=2
       vio_data_mode_value=2
       vio_addr_mode_value=3
    3. Observe the "**dbg_rddata_r**" and "**cmp_data_r**" signals and the "**cmp_addr_i**" bits of the **error_status[31:0]** bus in ChipScope
    4. Repeat steps 1-3 with setting vio_data_mode_value to values varying from 3-F
- Determine the frequency and reproducibility of the error
    - Does the error occur after every calibration or reset?
    - Does the error occur at specific temperature or voltage conditions?
- Determine if the error is correctable
    - Rewriting, rereading, resetting, recalibrating

NOTE: **vio_pause_traffic** should be asserted and de-asserted each time the VIO's inputs are changed.

Determining If a Data Error is Due to the Write or Read

Determining whether a data error is due to the write or the read can be difficult because if writes are the cause, read back of data is bad as well. In addition, issues with control or address timing affect both writes and reads. Some experiments that can help to isolate the issue are:
If errors are intermittent, issue a small initial number of writes, followed by continuous reads from those locations. If the reads intermittently yield bad data, there is a potential read problem. If the reads always yield the same (wrong) data, there is a write problem.

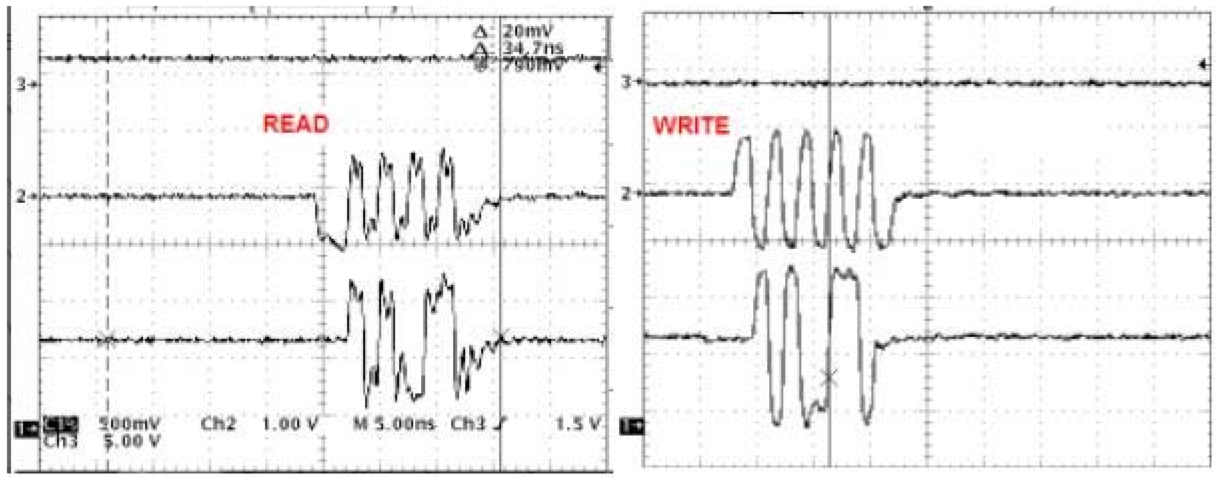
- Determine if this is a Write or Read problem using the MIG 7 Series Example Design Traffic Generator within ChipScope:
    1. Setup all the FIXED parameter values in the RTL:
        1. **Open example_top.v and change fixed_data_i and fixed_addr_i under the "traffic_gen_top" instantiation.**
            - .fixed_addr_i (32'b00000000000000000000000000001000),
            - .fixed_data_i (32'b11111111111111111111111111111111),
        2. Regenerate bitstream
    2. **Set the ChipScope trigger to cmp_error=1**
    3. Set ChipScope VIO's to:
       vio_modify_enable=1
       vio_pause_traffic=1
       vio_addr_mode_value=1
       vio_bl_mode_value=1
       vio_fixed_bl_value=8
       vio_instr_mode_value=1
       vio_fixed_instr_value=0 (Write Only)
       vio_data_mode_value=1
       vio_pause_traffic=0
    4. Set the ChipScope VIO's to:
       vio_pause_traffic=1
       vio_fixed_instr_value=1 (Read Only)

vio_pause_traffic=0

5. Observe the “**dbg_rddata_r**” and “**cmp_data_r**” signals in ChipScope

- This can also be done using high quality probes and a scope using the Traffic Generator or your own user design.
    1. Capture the write at the memory and the read at the FPGA to view data accuracy, appropriate DQS-to-DQ
    2. Look at the initial transition on DQS from tri-state to active.
    3. During Write, DQS does not have a preamble.
    4. During Read, the DQS has a low preamble that is 1 clock cycle long.

5. Following is an example of a Read and a Write to illustrate the difference:



- Analyze write timing:
  - If on-die termination (ODT) is used, check that the correct value is enabled in the DDR2/DDR3 device and that the timing on the ODT signal relative to the write burst is correct.
  - Measure the phase of DQ relative to DQS. During a Write, DQS should be center aligned to DQ. If the alignment is not correct, focus on the Debugging OCLKDELAYED Calibration section.
  - For debugging purposes only, use ODELAY to vary the phase of DQ relative to DQS.
- Analyze read timing:
  - Check the IDELAY values after calibration. Look for variations between IDELAY values. IDELAY values should be very similar for DQs in the same DQS group.
  - For debugging purposes only, vary the IDELAY taps after calibration for the bits that are returning bad data.

Checking and Varying Read/Write Timing

Debug signals are provided to verify read and write window margin on a per-byte basis and should be used for debugging purposes only. Determining if sufficient margin is available for reliable operation can be useful for debugging purposes if data errors are seen after calibration. There is an automated window check flow that can be used to step through the entire interface and provides the # of PHASER taps required to reach the left edge and right edge of the data window. The window checking can also be manually verified by manually incrementing and decrementing the PHASER taps to verify how much window margin is available.

Table 11: Debug Signals used for checking and varying read/write timing:

Debug Signal	ILA/VIO Connection	Signal Description
win_start	ddr3_vio_async_in_twm[0]/ ddr3_vio_sync_out_twm[0]	Single pulse that starts the window check logic.
win_sel_pi_pon	ddr3_vio_sync_out_twm[1]	Controls window check logic on read path or write path. Valid settings for this are: • 0x0: Enables Write Path • 0x1: Enables Read Path
dbg_sel_pi_incdec	ddr3_vio_sync_out_twm[2]	Enables manual incrementing and decrementing of the PHASER_IN taps
dbg_pi_f_inc	ddr3_vio_sync_out_twm[3]	Increments PHASER_IN fine taps when win_sel_pi_pon=0x1

dbg_pi_f_dec	ddr3_vio_sync_out_twm[4]	Decrements PHASER_IN fine taps when win_sel_pi_pon=0x1
dbg_sel_po_incdec	ddr3_vio_sync_out_twm[5]	Enables manual incrementing and decrementing of the PHASER_OUT taps
dbg_po_f_inc	ddr3_vio_sync_out_twm[6]	Increments PHASER_OUT fine taps when win_sel_pi_pon=0x0
dbg_po_f_dec	ddr3_vio_sync_out_twm[7]	Decrements PHASER_OUT fine taps when win_sel_pi_pon=0x0
vio_win_byte_select_inc	ddr3_vio_sync_out_twm[8]	Increments the byte group being checked
vio_win_byte_select_dec	ddr3_vio_sync_out_twm[9]	Decrements the byte group being checked
dbg_po_f_stg23_sel	ddr3_vio_sync_out_twm[10]	Selects phaser_out Stage 3 or Stage 2 taps for changing DQS tap value. Valid settings for this are: • 0x0: Enables Stage 2 taps • 0x1 Enables Stage 3 taps
dbg_pi_counter_read_val	ddr3_vio_async_in_twm [12:7]	Current PHASER_IN tap count corresponding to current byte being checked
pi_win_left_ram_out	ddr3_vio_async_in_twm [18:13]	PHASER_IN tap count to reach the left edge of the read window for a given byte.
pi_win_right_ram_out	ddr3_vio_async_in_twm [24:19]	PHASER_IN tap count to reach the right edge of the read window for a given byte.
win_active	ddr3_vio_async_in_twm [25]	Flag to indicate the Window check logic is active and measuring window margins. While active, the other VIO's should not be changed.
win_current_byte	ddr3_vio_async_in_twm [68:65]	Feedback to indicate which byte is currently being monitored.
win_byte_select	ddr3_vio_async_in_twm [75:69]	Selects which byte group to display the measured results for.
po_win_left_ram_out	ddr3_vio_async_in_twm [84:76]	PHASER_OUT tap count to reach the left edge of the write window for a given byte.
po_win_right_ram_out	ddr3_vio_async_in_twm [93:85]	PHASER_OUT tap count to reach the right edge of the write window for a given byte.
dbg_po_counter_read_val	ddr3_vio_async_in_twm [102:94]	Current PHASER_OUT tap count corresponding to current byte being checked
dbg_clear_error	ddr3_vio_sync_out[63]	Clears error in Traffic Generator as a result of changing tap values.

Automated Window Check

The automated window checking is enabled by asserting **win**_start with a single pulse. **win_active** should then assert until all byte groups have been measured. **win_sel_pi_pon** can be used to select between Read or Write measuring and **win_byte_select** can be used to select between each byte groups measured results and display them to the ChipScope window. To calculate the total data valid window use the following equation:

("Total # of taps" x CLK_PERIOD)/128 = Total Valid Data Window

NOTE: Both the Read and Write measured results are stored in separate block ram until **win_start** is asserted again.

Manual Window Check

To manually measure the data window margin, follow these steps:

1. Enable the manual window check by asserting **dbg_sel_pi_incdec** or **dbg_sel_po_incdec**.
   NOTE: When **dbg_sel_pi_incdec** or **dbg_sel_po_incdec** are enabled, **dbg_po_counter_read_val** and **dbg_pi_counter_read_cal** do not represent the true centered PHASER_IN/PHASER_OUT tap value.
2. Set the **ChipScope trigger to cmp_error=1**
3. Manually increment/decrement the taps using the **dbg_pi_f_inc**/**dbg_po_f_inc** or **dbg_pi_f_dec**/**dbg_po_f_dec** an event is triggered indicating a left or right edge was found. Make note of the # of taps that occurred until event triggered.
4. Manually increment/decrement the taps back the same # of taps
5. Issue a single pulse event to **dbg_clear_error**, and reset the ChipScope trigger
6. Manually increment/decrement the taps in the other direction using the **dbg_pi_f_inc**/**dbg_po_f_inc** or **dbg_pi_f_dec**/**dbg_po_f_dec** an event is triggered indicating a left or right edge was found. Make note of the # of taps that occurred until event triggered.
7. Add up the left and right tap values determined and calculate the total data valid window using the following equation:

   ("Total # of taps" x CLK_PERIOD)/128 = Total Valid Data Window

NOTE: dbg_po_f_stg23_sel is used to switch between the PHASER_OUT Stage 2 and Stage 3 tap delays. Stage 2 taps are used to adjust DQS/DQS# phase with respect to the CK/CK# during Write Leveling. Stage 3 taps are used to center DQS/DQS# within the corresponding DQ data window. Stage 2 tap shifts both the DQS/DQS# and the corresponding DQ bits while Stage 3 taps only shift DQS/DQS#.

Analyzing Calibration Results

When data errors occur, the results of calibration should be analyzed to ensure the results are expected and accurate. Each of the above debugging calibration sections notes what the expected results are such as how many edges should be found, how much variance across byte groups should exist, etc. Follow these sections to capture and then analyze the calibration results.

Conclusion

If this document does not help to resolve calibration or data errors, please create a WebCase with Xilinx Technical Support. Attach all of the captured ChipScope Pro tool waveforms, the completed 7 series DDR3 Calibration Results spreadsheet, and the details of your investigation and analysis.

Revision History

05/08/2012 – Initial release.
07/19/2012 – Added Data Error Debug Information.
07/31/2012 – Minor updates to document properties; updated keywords.
10/05/2012 – Added PRBS Read Leveling, Traffic Generator Data Error Debug, and Window Margin Check.