

This chapter describes the UniPHY layer of the external memory interface, and includes related information.

The major sections of this chapter are as follows:

- Block Description
- Interfaces
- UniPHY Signals
- PHY-to-Controller Interfaces
- Using a Custom Controller
- Using a Vendor-Specific Memory Model
- AFI 3.0 Specification
- Register Maps
- Efficiency Monitor and Protocol Checker
- UniPHY Calibration Stages

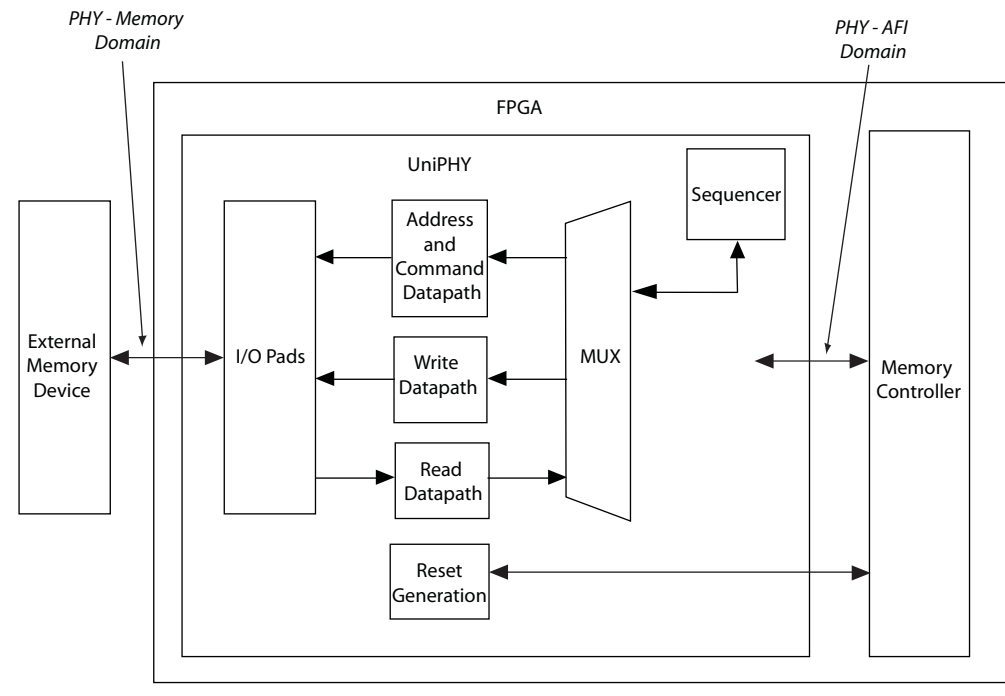
Block Description

This section describes the major functional units of the UniPHY layer, which include the following:

- Reset and Clock Generation
- Address and Command Datapath
- Write Datapath
- Read Datapath
- Sequencer

Figure 1-1 shows the PHY block diagram.

Figure 1-1. PHY Block Diagram



I/O Pads

The I/O pads contain all the I/O instantiations.

Reset and Clock Generation

At a high level, clocks in the PHY can be classified into two domains: the PHY-memory domain and the PHY-AFI domain. The PHY-memory domain interfaces with the external memory device and always operate at full-rate. The PHY-AFI domain interfaces with the memory controller and can be a full-rate, half-rate, or quarter-rate clock, based on the controller in use.

The number of clock domains in a memory interface can vary depending on its configuration; for example:

- At the PHY-memory boundary, separate clocks may exist to generate the memory clock signal, the output strobe, and to output write data, as well as address and command signals; these clocks include `p11_dq_write_clk`, `p11_write_clk`, `p11_mem_clk`, and `p11_addr_cmd_clk`. The aforementioned clocks are phase-shifted as required to achieve the desired timing relationships between memory clock, address and command signals, output data, and output strobe.
- For quarter-rate interfaces, additional clock domains such as `p11_hr_clock` are required to convert signals between half-rate and quarter-rate.

- For high-performance memory interfaces using Stratix V devices, additional clocks may be required to handle transfers between the device core and the I/O periphery for timing closure. For core-to-periphery transfers, the latch clock is `p11_c2p_write_clock`; for periphery-to-core transfers, it is `p11_p2c_read_clock`. These clocks are automatically phase-adjusted for timing closure during IP generation, but can be further adjusted in the parameter editor if necessary. If the phases of these clocks are zero, the fitter may remove these clocks during optimization.

Also, high-performance interfaces using a Nios II-based sequencer require two additional clocks, `p11_avl_clock` for the Nios II processor, and `p11_config_clock` for clocking the I/O scan chains during calibration.

For a complete list of clocks in your memory interface, compile your design and run the **Report Clocks** command in the TimeQuest Timing Analyzer.

Dedicated Clock Networks

The UniPHY layer employs three types of dedicated clock networks:

- Global clock network
- Dual-regional clock network
- PHY clock network (applicable only to Stratix V devices, and later)

The PHY clock network is a dedicated high-speed, low-skew, balanced clock tree designed for high-performance external memory interface. For device families that support the PHY clock network, UniPHY always uses the PHY clock network for all clocks at the PHY-memory boundary.

For families that do not support the PHY clock network, UniPHY uses either dual-regional or global clock networks for clocks at the PHY-memory boundary. During generation, the system selects dual-regional or global clocks automatically, depending on whether a given interface spans more than one quadrant. UniPHY does not mix the usage of dual-regional and global clock networks for clocks at the PHY-memory boundary; this ensures that timing characteristics of the various output paths are as similar as possible.

The `<variation_name>_pin_assignments.tcl` script creates the appropriate clock network type assignment. The use of the PHY clock network is specified directly in the RTL code, and does not require an assignment.

The UniPHY uses an active-low, asynchronous assert and synchronous de-assert reset scheme. The global reset signal resets the PLL in the PHY and the rest of the system is held in reset until after the PLL is locked.

Address and Command Datapath

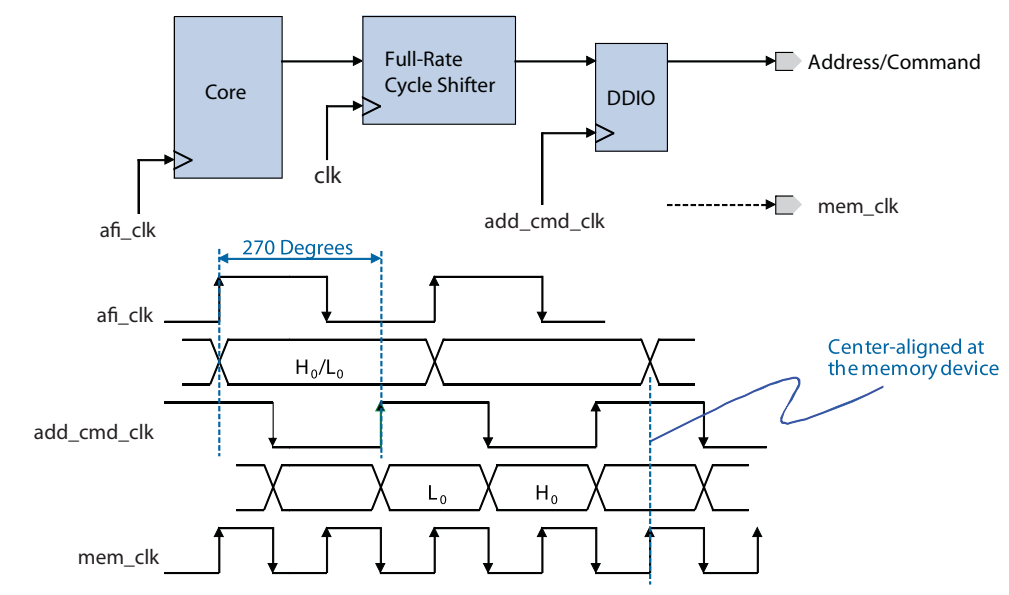
The memory controller controls the read and write addresses and commands to meet the memory specifications. The PHY is indifferent to address or command—that is, it performs no decoding or other operations—and the circuitry is the same for both. In full-rate and half-rate interfaces, address and command is full rate, while in quarter-rate interfaces, address and command is half rate.

Address and command signals are generated in the Altera PHY interface (AFI) clock domain and sent to the memory device in the address and command clock domain. The double data rate input/output (DDIO) stage converts the half-rate signals into full-rate signals, when the AFI clock runs at half-rate. For quarter-rate interfaces, additional DDIO stages exist to convert the address and command signals in the quarter-rate AFI clock domain to half-rate.

The address and command clock is offset with respect to the memory clock to balance the nominal setup and hold margins at the memory device (center-alignment). In the example of Figure 1-2, this offset is 270 degrees. The fitter can further optimize margins based on the actual delays and clock skews. In half-rate and quarter-rate designs, the full-rate cycle shifter blocks can perform a shift measured in full-rate cycles to implement the correct write latency; without this logic, the controller would only be able to implement even write latencies as it operates at half the speed. The full-rate cycle shifter is clocked by either the AFI clock or the address and command clock, depending on the PHY configuration, to maximize timing margins on the path from the AFI clock to the address and command clock.

Figure 1-2 illustrates the address and command datapath.

Figure 1-2. Address and Command Datapath (Half-rate example shown)

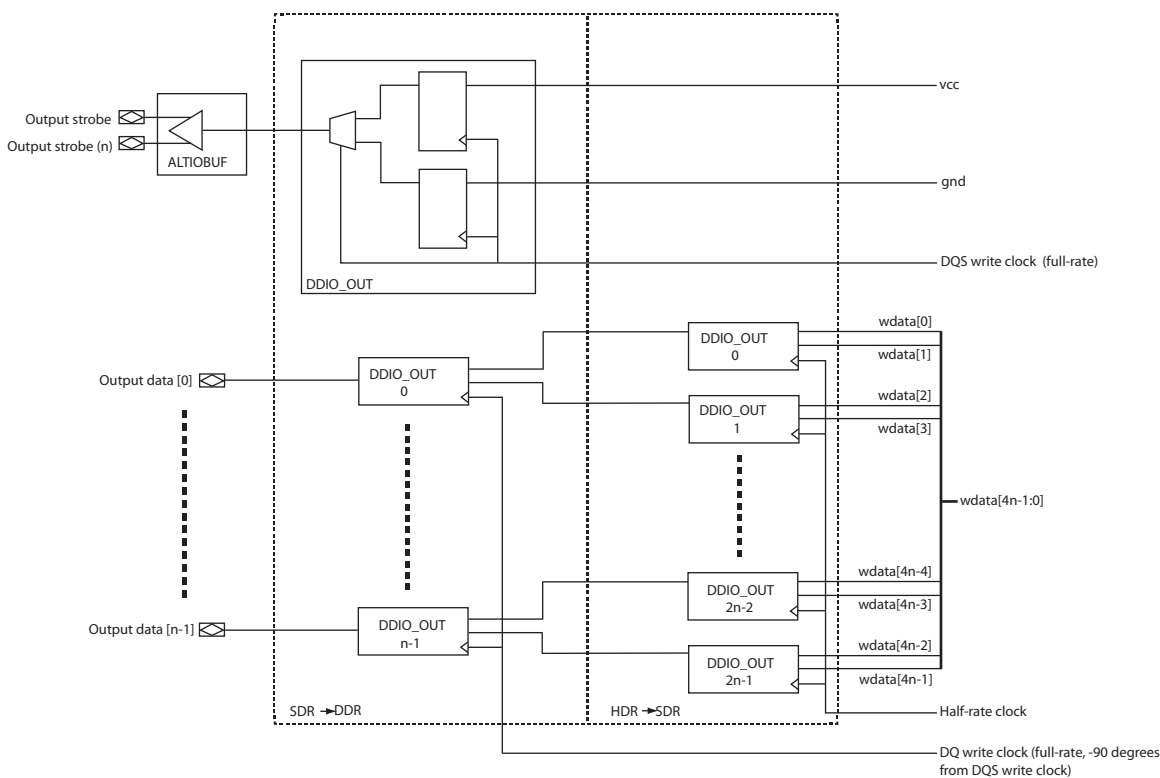


Write Datapath

The write datapath passes write data from the memory controller to the I/O. The write data valid signal from the memory controller generates the output enable signal to control the output buffer. For memory protocols with a bidirectional data bus, it also generates the dynamic termination control signal, which selects between series (output mode) and parallel (input mode) termination.

Figure 1-3 illustrates a simplified write datapath of a typical half-rate interface. The full-rate DQS write clock is sent to a DDIO_OUT cell. The output of DDIO_OUT feeds an output buffer which creates a pair of pseudodifferential clocks that connects to the memory. In full-rate mode, only the SDR-DDR portion of the path is used; in half-rate mode, the HDR-SDR circuitry is also required. The use of DDIO_OUT in both the output strobe and output data generation path ensures that their timing characteristics are as similar as possible. The `<variation_name>_pin_assignments.tcl` script automatically specifies the logic option that associates all data pins to the output strobe pin. The Fitter treats the pins as a DQS/DQ pin group.

Figure 1-3. Write Datapath



Leveling Circuitry

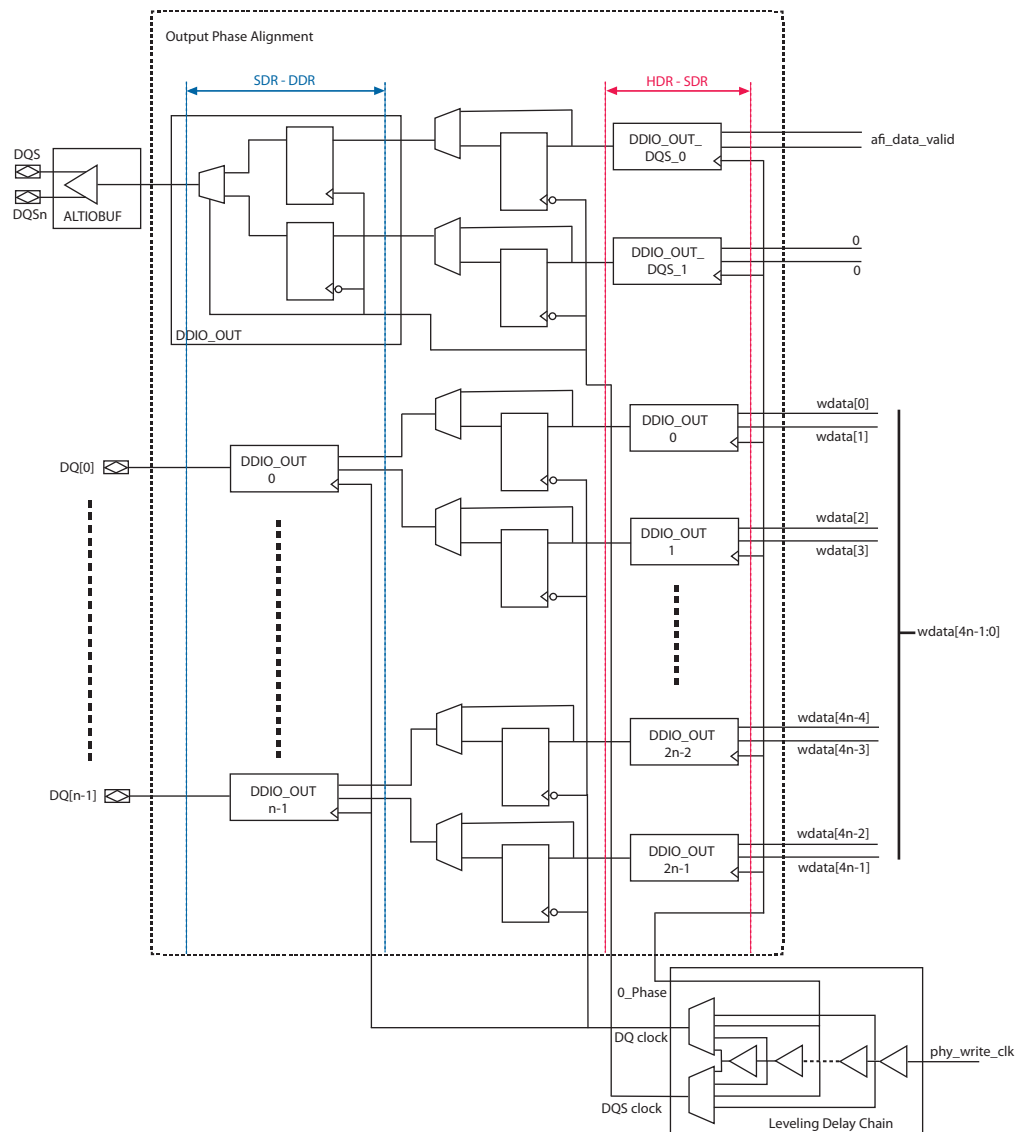
For DDR2, leveling circuitry is invoked automatically for frequencies above 240 MHz; no leveling is used for frequencies below 240 MHz. For DDR3, leveling is always invoked, whether the interface targets a DIMM or a single component.

For DDR2 at frequencies below 240 MHz, you should use a tree-style layout. For frequencies above 240 MHz, you can choose either a leveled or balanced-T or Y topology, as the leveled PHY calibrates correctly to either implementation. For DDR3 implementations at higher frequencies, a fly-by topology is recommended for optimal performance. For DDR3 with Arria II GZ devices, a balanced-T PCB topology for address/command/clock must be used because fly-by topology is not supported.

For details about leveling delay chains, consult the memory interfaces hardware section of the device handbook for your FPGA.

Figure 1-4 shows the write datapath for a leveling interface. The full-rate PLL output clock `phy_write_clk` goes to a leveling delay chain block which generates all other peripheral clocks that are needed. The data signals that generate DQ and DQS signals pass to an output phase alignment block. The output phase alignment block feeds an output buffer which creates a pair of pseudo differential clocks that connect to the memory. In full-rate designs, only the SDR-DDR portion of the path is used; in half-rate mode, the HDR-SDR circuitry is also required. The use of DDIO_OUT in both the output strobe and output data generation paths ensures that their timing characteristics are as similar as possible. The `<variation_name>_pin_assignments.tcl` script automatically specifies the logic option that associates all data pins to the output strobe pin. The Quartus II Fitter treats the pins as a DQS/DQ pin group.

Figure 1-4. Write Datapath for a Leveling Interface



Read Datapath

Figure 1-5 shows the read datapath. This section describes the blocks and flow in the read datapath.

For all protocols, the DQS logic block delays the strobe by 90 degrees to center-align the rising strobe edge within the data window. For DDR2 and DDR3 protocols, the logic block also performs strobe gating, holding the DQS enable signal high for the entire period that data is received. One DQS logic block exists for each data group.

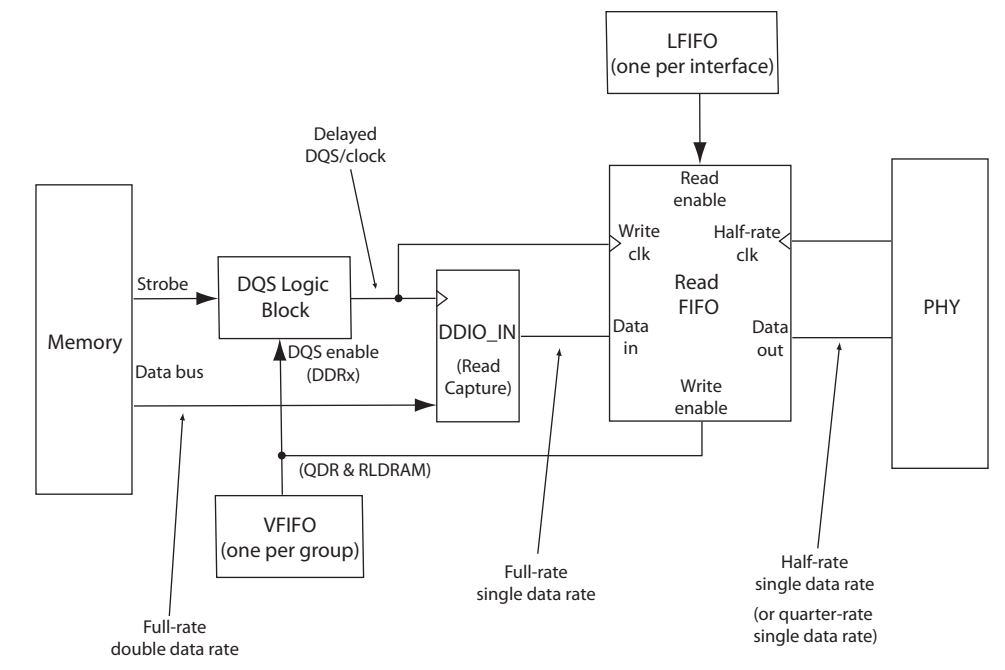
One VFIFO buffer exists for each data group. For DDR2 and DDR3 protocols, the VFIFO buffer generates the DQS enable signal, which is delayed (by an amount determined during calibration) to align with the incoming DQS signal. For QDR and RLDRAM protocols, the output of the VFIFO buffer serves as the write enable signal for the Read FIFO buffer, signaling when to begin capturing data.

DDIO_IN receives data from memory at double-data rate and passes data on to the Read FIFO buffer at single-data rate.

The Read FIFO buffer temporarily holds data read from memory; one Read FIFO buffer exists for each data group. For half-rate interfaces, the Read FIFO buffer converts the full-rate, single data-rate input to a half-rate, single data-rate output which is then passed to the PHY core logic. In the case of a quarter-rate interface, soft logic in the PHY performs an additional conversion from half-rate single data rate to quarter-rate single data rate.

One LFIFO buffer exists for each memory interface; the LFIFO buffer generates the read enable signal for all Read FIFO blocks in an interface. The read enable signal is asserted when the Read FIFO blocks have buffered sufficient data from the memory to be read. The timing of the read enable signal is determined during calibration.

Figure 1-5. Read Datapath



Sequencer

Depending on the combination of protocol and IP architecture in your external memory interface, you may have either an RTL-based sequencer or a Nios II based sequencer. This section discusses the Nios II-based sequencer and the RTL-based sequencer.



[Table 8–6](#) in section 2 of this volume shows the sequencer support for different protocol-architecture combinations.



Be aware that RTL-based sequencer implementations and Nios II-based sequencer implementations can have different pin requirements. You may not be able to migrate from an RTL-based sequencer to a Nios II-based sequencer and maintain the same pinout.



For information on pin planning, refer to [Planning Pin and FPGA Resources](#) in volume 2 of the *External Memory Interface Handbook*.

Nios II-Based Sequencer

The DDR2 and DDR3 SDRAM controllers with UniPHY employ a Nios[®] II-based sequencer that is parameterizable and is dynamically generated at run time. The Nios II-based sequencer is also available with the QDR II and RLDRAM II controllers.

Function

The sequencer enables high-frequency memory interface operation by calibrating the interface to compensate for variations in setup and hold requirements caused by transmission delays.

The UniPHY converts the double-data rate interface of high-speed memory devices to a full-rate or half-rate interface for use within an FPGA. To compensate for slight variations in data transmission to and from the memory device, double-data rate is usually center-aligned with its strobe signal; nonetheless, at high speeds, slight variations in delay can result in setup or hold time violations. The sequencer implements a calibration algorithm to determine the combination of delay and phase settings necessary to maintain center-alignment of data and clock signals, even in the presence of significant delay variations. Programmable delay chains in the FPGA I/Os then implement the calculated delays to ensure that data remains centered. Calibration also applies settings to the FIFO buffers within the PHY to minimize latency and ensures that the read valid signal is generated at the appropriate time.

When calibration is completed, the sequencer returns control to the memory controller.



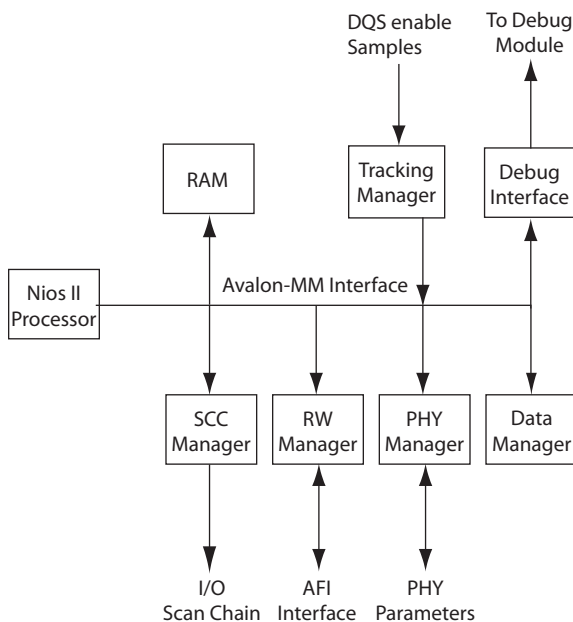
For more information about calibration, refer to [UniPHY Calibration Stages](#).

Architecture

[Figure 1–6](#) shows the sequencer block diagram. The sequencer is composed of a Nios II processor and a series of hardware-based component managers, connected together by an Avalon bus. The Nios-II processor performs the high-level algorithmic operations of calibration, while the component managers handle the lower-level timing, memory protocol, and bit-manipulation operations.

The high-level calibration algorithms are specified in C code which is compiled into Nios II code that resides in the FPGA RAM blocks. The debug interface provides a mechanism for interacting with the various managers and for tracking the progress of the calibration algorithm, and can be useful for debugging problems that arise within the PHY. The various managers are specified in RTL and implement operations that would be slow or inefficient if implemented in software.

Figure 1-6. Sequencer Block Diagram



The C code that defines the calibration routines is available for your reference in the `\<name>_s0_software` subdirectory. Altera does not recommend that you modify this C code.

SCC Manager

The scan chain control (SCC) manager allows the sequencer to set various delays and phases on the I/Os that make up the memory interface. The latest Altera device families provide dynamic delay chains on input, output, and output enable paths which can be reconfigured at runtime. The SCC manager provides the calibration routines access to these chains to add delay on incoming and outgoing signals. A master on the Avalon-MM interface may require the maximum allowed delay setting on input and output paths, and may set a particular delay value in this range to apply to the paths.

The SCC manager implements the Avalon-MM interface and the storage mechanism for all input, output, and phase settings. It contains circuitry that configures a DQ- or DQS-configuration block. The Nios II processor may set delay, phases, or register settings; the sequencer scans the settings serially to the appropriate DQ or DQS configuration block.

RW Manager

The read write (RW) manager encapsulates the protocol to read and write to the memory device through the AFI. It provides a buffer that stores the data to be sent to and read from memory, and provides the following commands:

- Write configuration—configures the memory for use. Sets up burst lengths, read and write latencies, and other device specific parameters.
- Refresh—initiates a refresh operation at the DRAM. The command does not exist on SRAM devices. The sequencer also provides a register that determines whether the RW manager automatically generates refresh signals.
- Enable or disable multi-purpose register (MPR)—some memory devices provide a special register that contains calibration specific patterns that you can read. This command enables or disables access to this register.
- Activate row—for memory devices that have both rows and columns, this command activates a specific row. Subsequent reads and writes operate on this specific row.
- Precharge—closes a row before you can access a new row.
- Write or read burst—writes or reads a burst length of data.
- Write guaranteed—write with a special mode where the memory holds address and data lines constant. Altera guarantees this type of write to work in the presence of skew, but constrains to write the same data across the entire burst length.
- Write and read back-to-back—performs back-to-back writes or reads to adjacent banks. Most memory devices have strict timing constraints on subsequent accesses to the same bank, thus back-to-back writes and reads have to reference different banks.
- Protocol-specific initialization—which the initialization sequence requires.

PHY Manager

The PHY manager provides access to the PHY for calibration, and passes relevant calibration results to the PHY. For example, the PHY Manager sets the VFIFO and LFIFO buffer parameters resulting from calibration, signals the PHY when the memory initialization sequence finishes, and reports the pass/fail status of calibration.

Data Manager

The Data Manager stores parameterization-specific data in RAM, for the software to query.

Tracking Manager

The Tracking Manager detects the effects of voltage and temperature variations that can occur on the memory device over time resulting in reduced margins, and adjusts the DQS capture delay as necessary to maintain adequate operating margins.

The Tracking Manager operates by briefly assuming control of the AFI interface after each memory refresh cycle, issuing a read routine to the RW Manager, and then sampling the DQS tracking. Ideally, the falling edge of the DQS enable signal would align to the last rising edge of the raw DQS signal from the memory device. The Tracking Manager determines whether the DQS enable signal is leading or trailing the raw DQS signal.

When the Tracking Manager determines that the DQS enable signal is either leading or lagging the raw DQS signal, it adjusts the DQS enable appropriately.

Nios II Processor

The Nios II processor manages the calibration algorithm; the NIOS II processor is unavailable once calibration is completed.

The same calibration algorithm supports all device families, with some differences. The following sections describe the calibration algorithm for DDR3 SDRAM on Stratix III devices. Calibration algorithms for other protocols and families are a subset and significant differences are pointed out when necessary. As the algorithm is fully contained in the software of the sequencer (in the C code) enabling and disabling specific steps involves turning flags on and off.

Calibration comprises the following stages:

- Initialize memory
- Calibrate read datapath
- Calibrate write datapath
- Run diagnostics

Initialize Memory

Calibration must initialize all memory devices before they can operate properly. The sequencer performs this memory initialization stage when it takes control of the PHY at startup.

Calibrate Read Datapath

Calibrating the read datapath comprises the following steps:

- Calibrate DQS enable cycle and phase
- Perform read per-bit deskew to center the strobe signal within data valid window
- Reduce LFIFO latency

Calibrate Write Datapath

Calibrating the write datapath involves the following steps:

- Center align DQS with respect to DQ.
- Align DQS with `mem_clk`.

Test Diagnostics

The sequencer estimates the read and write margins under noisy conditions, by sweeping input and output DQ and DQS delays to determine the size of the data valid windows on the input and output sides. The sequencer stores this information in the local memory and you can access it through the debugging interface.

When the diagnostic test finishes, control of the PHY interface passes back to the controller and the sequencer issues a pass or fail signal.

RTL-based Sequencer

The RTL-based sequencer is available for QDR II and RLDRAM II interfaces; it is a state machine that processes the calibration algorithm. The sequencer assumes control of the interface at reset (whether at initial startup or when the IP is reset) and maintains control throughout the calibration process, relinquishing control to the memory controller only after successful calibration. Table 1-1 shows the major states in the RTL-based sequencer.

Table 1-1. Sequencer States (Part 1 of 2)

State	Description
RESET	Remain in this state until reset is released.
LOAD_INIT	Load any initialization values for simulation purposes.
STABLE	Wait until the memory device is stable.
WRITE_ZERO	Issue write command to address 0.
WAIT_WRITE_ZERO	Write all 0s to address 0.
WRITE_ONE	Issue write command to address 1.
WAIT_WRITE_ONE	Write all 1s to address 1.
Valid Calibration States	
V_READ_ZERO	Issue read command to address 0 (expected data is all 0s).
V_READ_NOP	This state represents the minimum number of cycles required between 2 back-to-back read commands. The number of NOP states depends on the burst length.
V_READ_ONE	Issue read command to address 1 (expected data is all 1s).
V_WAIT_READ	Wait for read valid signal.
V_COMPARE_READ_ZERO_READ_ONE	Parameterizable number of cycles to wait before making the read data comparisons.
V_CHECK_READ_FAIL	When a read fails, the write pointer (in the AFI clock domain) of the valid FIFO buffer is incremented. The read pointer of the valid FIFO buffer is in the DQS clock domain. The gap between the read and write pointers is effectively the latency between the time when the PHY receives the read command and the time valid data is returned to the PHY.
V_ADD_FULL_RATE	Advance the read valid FIFO buffer write pointer by an extra full rate cycle.
V_ADD_HALF_RATE	Advance the read valid FIFO buffer write pointer by an extra half rate cycle. In full-rate designs, equivalent to V_ADD_FULL_RATE.
V_READ_FIFO_RESET	Reset the read and write pointers of the read data synchronization FIFO buffer.
V_CALIB_DONE	Valid calibration is successful.

Table 1-1. Sequencer States (Part 2 of 2)

State	Description
Latency Calibration States	
L_READ_ONE	Issue read command to address 1 (expected data is all 1s).
L_WAIT_READ	Wait for read valid signal from read datapath. Initial read latency is set to a predefined maximum value.
L_COMPARE_READ_ONE	Check returned read data against expected data. If data is correct, go to L_REDUCE_LATENCY; otherwise go to L_ADD_MARGIN.
L_REDUCE_LATENCY	Reduce the latency counter by 1.
L_READ_FLUSH	Read from address 0 (expected data is all 0s), to flush the contents of the read data resynchronization FIFO buffer.
L_WAIT_READ_FLUSH	Wait until the whole FIFO buffer is flushed, then go back to L_READ and try again.
L_ADD_MARGIN	Increment latency counter by 3 (1 cycle to get the correct data, 2 more cycles of margin for run time variations). If latency counter value is smaller than predefined ideal condition minimum, then go to CALIB_FAIL.
CALIB_DONE	Calibration is successful.
CALIB_FAIL	Calibration is not successful.

DLL Offset Control Block

For designs generated with HardCopy compatibility enabled, the UniPHY layer includes DLL offset control blocks, which allow adjustment of the DLL control word for modifying the DQS delay chains to compensate for variations in silicon.

A DLL offset control block can feed only one side of the chip, therefore the IP instantiates two DLL offset control blocks for each DLL, to permit control of resources on two sides of the chip, as in the case of wraparound designs. In non-wraparound designs, where the second DLL offset control block is not needed, the second control block remains unused and disappears during synthesis.

One complication of the dual DLL offset control block process, is that at generation time it is impossible to know where resources are going to be placed, so the system automatically connects the output of the first DLL offset control block to all DQS groups. In the case of a wraparound design, you must modify the RTL code after pin placement, to connect the second DLL offset control block.

To connect the second DLL offset control block, follow these steps:

1. Open the file `<variation_name>_new_io_pads.v` in an editor.
2. Find a line of a form similar to the following:

```
.dll_offsetdelay_in((i < 0) ?  
hc_dll_config_dll_offset_ctrl_offsetctrlout :  
hc_dll_config_dll_offset_ctrl_offsetctrlout),
```

This line connects the output of the first DLL offset control block (`hc_dll_config_dll_offset_ctrl_offsetctrlout`) to the offset control input of the DQS delay chain, for every DQS delay chain where $i < 0$.

- Modify the above line to connect the second DLL offset control block. The following example shows the correct syntax to connect groups 0 to 3 to the output of the first DLL offset control block, and groups 4 and above to the output of the second DLL offset control block:

```
.dll_offsetdelay_in((i < 4) ?
hc_dll_config_dll_offset_ctrl_offsetctrlout :
hc_dll_config_dll_offset_ctrl_b_offsetctrlout),
```

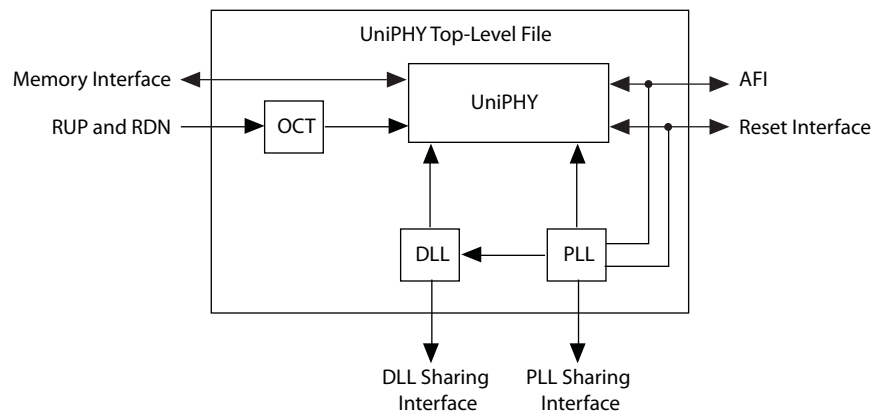
Interfaces

Figure 1-7 shows the major blocks of the UniPHY and how it interfaces with the external memory device and the controller.



Instantiating the delay-locked loop (DLL) and the phase-locked loop (PLL) on the same level as the UniPHY eases DLL and PLL sharing.

Figure 1-7. UniPHY Interfaces with the Controller and the External Memory



The following interfaces are on the UniPHY top-level file:

- AFI
- Memory interface
- DLL sharing interface
- PLL sharing interface
- OCT interface

AFI

The UniPHY datapath uses the Altera PHY interface (AFI). The AFI is a simple connection between the PHY and controller; the AFI is based on the DDR PHY interface (DFI) specification, with some calibration-related signals not used and some additional Altera-specific sideband signals added.


For more information about the AFI, refer to [AFI 3.0 Specification](#).

The Memory Interface

For more information on the memory interface, refer to “UniPHY Signals” on page 1-18.


The DLL and PLL Sharing Interface


You can generate the UniPHY memory interface and configure it to share its PLL and/or DLL interfaces. By default, a UniPHY memory interface variant contains a PLL and DLL; the PLL produces a variety of required clock signals derived from the reference clock, and the DLL produces a delay codeword. In this case the PLL sharing mode is "No sharing". A UniPHY variant can be configured as a PLL Master and/or DLL Master, in which case the corresponding interfaces are exported to the UniPHY top-level and can be connected to an identically configured UniPHY variant PLL Slave and/or DLL Slave. The UniPHY slave variant is instantiated without a PLL and/or DLL, which saves device resources.

 For Arria II GX, Arria II GZ, Stratix III, and Stratix IV devices, the PLL and DLL must both be shared at the same time—their sharing modes must match. This restriction does not apply to Arria V, Cyclone V, or Stratix V devices.

To share PLLs or DLLs, follow these steps:


1. To create a PLL or DLL master, create a UniPHY memory interface IP core. To make the PLL and/or DLL interface appear at the top-level in the core, on the **PHY Settings** tab in the parameter editor, set the **PLL Sharing Mode** and/or **DLL Sharing Mode** to **Master**.
2. To create a PLL or DLL slave, create a second UniPHY memory interface IP core. To make the PLL and/or DLL interface appear at the top-level in the core, on the **PHY Settings** tab set the **PLL Sharing Mode** and/or **DLL Sharing Mode** to **Slave**.
3. Connect the PLL and/or DLL sharing interfaces by following the appropriate step, below:
 - **For cores generated with Megawizard Plug-in Manager:** connect the PLL and/or DLL interface ports between the master and slave cores in your wrapper RTL. When using PLL sharing, connect the `afi_clk`, `afi_half_clk`, and `afi_reset_n` outputs from the UniPHY PLL master to the `afi_clk`, `afi_half_clk`, and `afi_reset_in` inputs on the UniPHY PLL slave
 - **For cores generated with Qsys,** connect the PLL and/or DLL interface in the Qsys GUI. When using PLL sharing, connect the `afi_clk`, `afi_half_clk`, and `afi_reset` interfaces from the UniPHY PLL master to the `afi_clk_in`, `afi_half_clk_in`, and `afi_reset_in` interfaces on the UniPHY PLL slave. You can connect only one slave to the master using the GUI; if you want to connect multiple slaves to one master, you must edit the generated RTL code manually.

 The master **.qip** file must appear before the slave **.qip** file in the Quartus II Settings File (**.qsf**) when using Megawizard, or in the Qsys system file (**.qip**) when using Qsys.

 If you generate a slave IP core, you must modify the timing scripts to allow the timing analysis to correctly resolve clock names and analyze the IP core. Otherwise the software issues critical warnings and an incorrect timing report.


To modify the timing script, follow these steps:

1. In a text editor, open your system's Tcl timing scripts file, as follows:
 - For systems generated with the MegaWizard Plug-In Manager:
Open the `<IP core name>/<IP core name>_p0_timing.tcl` file.
 - For systems generated with Qsys or SOPC Builder:
Open the `<HDL Path>/<submodules>/<master_corename>_timing.tcl` file.
2. Search for the following 2 lines:
 - `set ::master_corename "_MASTER_CORE_"`
 - `set ::master_instname "_MASTER_INST_"`
3. Replace `_MASTER_CORE_` with the core name and `_MASTER_INST_` with instance name of the UniPHY master to which the slave is connected.

 You can determine the master core name from the name of the Tcl timing file:

For systems generated with Megawizard, this is: `<IP master variant name>/<master_corename>_timing.tcl`, where `<master_corename>` is the name of the core.

For systems generated with Qsys or SOPC Builder, this is: `<HDL Path>/<submodules>/<master_corename>_timing.tcl`, where `<master_corename>` is the name of the core.

 The instance name is the full path to the instance and is in the `<IP core name>_all_pins.txt` file that is automatically generated after the `<IP core name>_pin_assignments.tcl` script runs.


4. If the slave component is connected to a user-defined PLL rather than a UniPHY master, you must manually enter all clock names.
 - In the `<instance_name>_timing.tcl` file, remove the `master_corename` and `master_instname` variables with the checks performed in the eight lines following them.
 - In the `<instance_name>.sdc` file, manually define all local pll clock name variables. For example:

```
set local_pll_afi_clk "mycomponent|mypll|my_afi_clk"
```

 where "mycomponent" is the path to where the PLL is instantiated from the top level, "mypll" is the PLL name, and "my_afi_clk" is the name of the wire connected to the PLL. If you are unsure of the exact clock name and path, use the Node Finder in the TimeQuest timing analyzer to find it.



You must be very careful when connecting clock signals to the slave. Connecting to clocks with frequency or phase different than what the core expects may result in hardware failure.

 The PLL and DLL sharing interfaces are available when using SOPC Builder, however the PLL and/or DLL interfaces cannot be connected using the SOPC Builder GUI interface. To complete master-slave connections when using SOPC Builder, you must edit the generated RTL code manually.

About PLL Simulation

PLL frequencies may differ between the synthesis and simulation file sets. In either case the achieved PLL frequencies and phases are calculated and reported in real time in the parameter editor.

For the simulation file set, clocks are specified in the RTL, not in units of frequency but by the period in picoseconds, thus avoiding clock drift due to picosecond rounding error.

For the synthesis file set, there are two mechanisms by which clock frequencies are specified in the RTL, based on the target device family:

- For Arria V, Cyclone V, and Stratix V, clock frequencies are specified in megahertz.
- For Arria II GX, Arria II GZ, Stratix III, and Stratix IV, clock frequencies are specified by integer multipliers and divisors. For these families, the real simulation model—as opposed to the default abstract simulation model—also uses clock frequencies specified by integer ratios.

The OCT Sharing Interface

By default, the UniPHY IP generates the required OCT control block at the top-level RTL file for the PHY. If you want, you can instantiate this block elsewhere in your code and feed the required termination control signals into the IP core by turning off **Master for OCT Control Block** on the **PHY Settings** tab. If you turn off **Master for OCT Control Block**, you must instantiate the OCT control block or use another UniPHY instance as a master, and ensure that the parallel and series termination control bus signals are connected to the PHY.

Figure 1-8 and Figure 1-9, respectively, show the PHY architecture with and without Master for OCT Control Block

Figure 1-8. PHY Architecture with Master for OCT Control Block

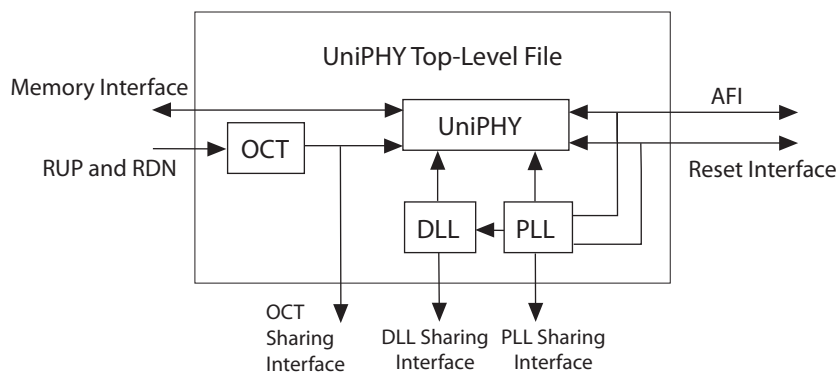
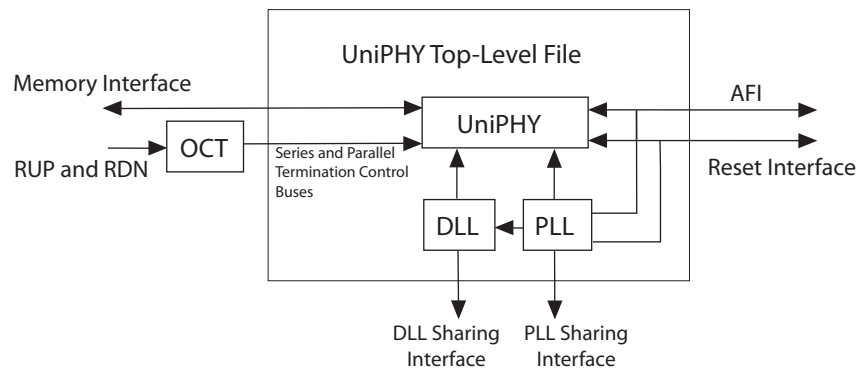



Figure 1-9. PHY Architecture without Master for OCT Control Block

 The OCT sharing interface is available when using SOPC Builder, however the OCT sharing interface cannot be connected using the SOPC Builder parameter editor. To complete master-slave connections when using SOPC Builder, you must edit the generated RTL code manually.

If you generate a QDR II or RLDRAM II slave IP core, you must modify the pin assignment script to allow the fitter to correctly resolve the OCT termination block name in the OCT master core.

To modify the pin assignment script for QDR II or RLDRAM II slaves, follow these steps:

1. In a text editor, open your system's Tcl timing scripts file, as follows:
 - For systems generated with the MegaWizard Plug-In Manager:
Open the `<IP core name>/<IP core name>_p0_timing.tcl` file.
 - For systems generated with Qsys or SOPC Builder:
Open the `<HDL Path>/<submodules>/<master_corename>_pin_assignments.tcl` file.
2. Search for the following line:
 - `set ::master_corename "_MASTER_CORE_"`
3. Replace `_MASTER_CORE_` with the core name of the UniPHY master to which the slave is connected. The name to use is the same as that for the **instance name** used in step 3 of [The DLL and PLL Sharing Interface](#) instructions.

UniPHY Signals

This section describes the UniPHY signals.

[Table 1-2](#) shows the clock and reset signals.

Table 1-2. Clock and Reset Signals (Part 1 of 2)

Name	Direction	Width	Description
<code>pll_ref_clk</code>	Input	1	PLL reference clock input.
<code>global_reset_n</code>	Input	1	Active low global reset for PLL and all logic in the PHY, which causes a complete reset of the whole system.

Table 1-2. Clock and Reset Signals (Part 2 of 2)

Name	Direction	Width	Description
soft_reset_n	Input	1	Holding soft_reset_n low holds the PHY in a reset state. However it does not reset the PLL, which keeps running. It also holds the afi_reset_n output low. Mainly for use by SOPC Builder.
reset_request_n	Output	1	When the PLL is locked, reset_request_n is high. When the PLL is out of lock, reset_request_n is low.

Table 1-3 shows the DDR2 and DDR3 SDRAM interface signals.

Table 1-3. DDR2 and DDR3 SDRAM Interface Signals

Name	Direction	Width	Description
mem_ck, mem_ck_n	Output	MEM_CK_WIDTH	Memory clock.
mem_cke	Output	MEM_CLK_EN_WIDTH	Clock enable.
mem_cs_n	Output	MEM_CHIP_SELECT_WIDTH	Chip select..
mem_cas_n	Output	MEM_CONTROL_WIDTH	Column address strobe.
mem_ras_n	Output	MEM_CONTROL_WIDTH	Row address strobe.
mem_we_n	Output	MEM_CONTROL_WIDTH	Write enable.
mem_a	Output	MEM_ADDRESS_WIDTH	Address.
mem_ba	Output	MEM_BANK_ADDRESS_WIDTH	Bank address.
mem_dqs, mem_dqs_n	Bidirectional	MEM_DQS_WIDTH	Data strobe.
mem_dq	Bidirectional	MEM_DQ_WIDTH	Data.
mem_dm	Output	MEM_DM_WIDTH	Data mask.
mem_odt	Output	MEM_ODT_WIDTH	On-die termination.
mem_reset_n (DDR3 only)	Output	1	Reset
mem_ac_parity (RDIMM only)	Output	MEM_CONTROL_WIDTH	Address/command parity bit.
mem_err_out_n (RDIMM only)	Input	MEM_CONTROL_WIDTH	Address/command parity error.

 For information about the AFI signals, refer to [AFI 3.0 Specification](#).

 For information about top-level HardCopy migration signals, refer to [HardCopy Design Migration Guidelines](#) in volume 2 of the *External Memory Interface Handbook*.

Table 1-4 shows parameters relating to UniPHY signals.

Table 1-4. Parameters (Part 1 of 3)

Parameter Name	Description
AFI_RATIO	AFI_RATIO is 1 in full-rate designs. AFI_RATIO is 2 for half-rate designs. AFI_RATIO is 4 for quarter-rate designs.
MEM_IF_DQS_WIDTH	The number of DQS pins in the interface.
MEM_ADDRESS_WIDTH	The address width of the specified memory device.
MEM_BANK_WIDTH	The bank width of the specified memory device.

Table 1-4. Parameters (Part 2 of 3)

Parameter Name	Description
MEM_CHIP_SELECT_WIDTH	The chip select width of the specified memory device.
MEM_CONTROL_WIDTH	The control width of the specified memory device.
MEM_DM_WIDTH	The DM width of the specified memory device.
MEM_DQ_WIDTH	The DQ width of the specified memory device.
MEM_READ_DQS_WIDTH	The READ DQS width of the specified memory device.
MEM_WRITE_DQS_WIDTH	The WRITE DQS width of the specified memory device.
OCT_SERIES_TERM_CONTROL_WIDTH	—
OCT_PARALLEL_TERM_CONTROL_WIDTH	—
AFI_ADDRESS_WIDTH	The AFI address width, derived from the corresponding memory interface width.
AFI_BANK_WIDTH	The AFI bank width, derived from the corresponding memory interface width.
AFI_CHIP_SELECT_WIDTH	The AFI chip select width, derived from the corresponding memory interface width.
AFI_DATA_MASK_WIDTH	The AFI data mask width.
AFI_CONTROL_WIDTH	The AFI control width, derived from the corresponding memory interface width.
AFI_DATA_WIDTH	The AFI data width.
AFI_DQS_WIDTH	The AFI DQS width.
DLL_DELAY_CTRL_WIDTH	The DLL delay output control width.
NUM_SUBGROUP_PER_READ_DQS	A read datapath parameter for timing purposes.
QVLD_EXTRA_FLOP_STAGES	A read datapath parameter for timing purposes.
READ_VALID_TIMEOUT_WIDTH	A read datapath parameter; calibration fails when the timeout counter expires.
READ_VALID_FIFO_WRITE_ADDR_WIDTH	A read datapath parameter; the write address width for half-rate clocks.
READ_VALID_FIFO_READ_ADDR_WIDTH	A read datapath parameter; the read address width for full-rate clocks.
MAX_LATENCY_COUNT_WIDTH	A latency calibration parameter; the maximum latency count width.
MAX_READ_LATENCY	A latency calibration parameter; the maximum read latency.
READ_FIFO_READ_ADDR_WIDTH	—
READ_FIFO_WRITE_ADDR_WIDTH	—
MAX_WRITE_LATENCY_COUNT_WIDTH	A write datapath parameter; the maximum write latency count width.
INIT_COUNT_WIDTH	An initialization sequence.
MRSC_COUNT_WIDTH	A memory-specific initialization parameter.
INIT_NOP_COUNT_WIDTH	A memory-specific initialization parameter.
MRS_CONFIGURATION	A memory-specific initialization parameter.
MRS_BURST_LENGTH	A memory-specific initialization parameter.
MRS_ADDRESS_MODE	A memory-specific initialization parameter.
MRS_DLL_RESET	A memory-specific initialization parameter.

Table 1-4. Parameters (Part 3 of 3)

Parameter Name	Description
MRS_IMP_MATCHING	A memory-specific initialization parameter.
MRS_ODT_EN	A memory-specific initialization parameter.
MRS_BURST_LENGTH	A memory-specific initialization parameter.
MEM_T_WL	A memory-specific initialization parameter.
MEM_T_RL	A memory-specific initialization parameter.
SEQ_BURST_COUNT_WIDTH	The burst count width for the sequencer.
VCALIB_COUNT_WIDTH	The width of a counter that the sequencer uses.
DOUBLE_MEM_DQ_WIDTH	—
HALF_AFI_DATA_WIDTH	—
CALIB_REG_WIDTH	The width of the calibration status register.
NUM_AFI_RESET	The number of AFI resets to generate.

PHY-to-Controller Interfaces

This section describes the typical modules that are connected to the UniPHY and the port name prefixes each module uses. This section describes using a custom controller and describes the AFI.

The AFI standardizes and simplifies the interface between controller and PHY for all Altera memory designs, thus allowing you to easily interchange your own controller code with Altera's high-performance controllers. The AFI PHY interface includes an administration block that configures the memory for calibration and performs necessary accesses to mode registers that configure the memory as required.

For half-rate designs, the address and command signals in the UniPHY are asserted for one `mem_clk` cycle (1T addressing), such that there are two input bits per address and command pin in half-rate designs. If you require a more conservative 2T addressing (where signals are asserted for two `mem_clk` cycles), drive both input bits (of the address and command signal) identically in half-rate designs.

Figure 1-10 shows the half-rate write operation.

Figure 1-10. Half-Rate Write with Word-Aligned Data

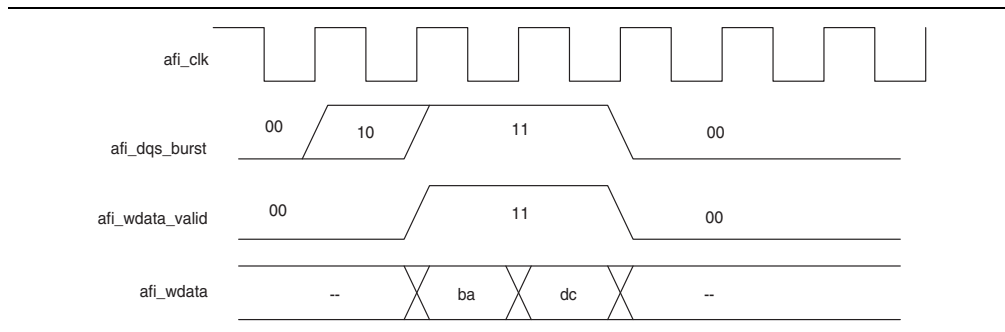
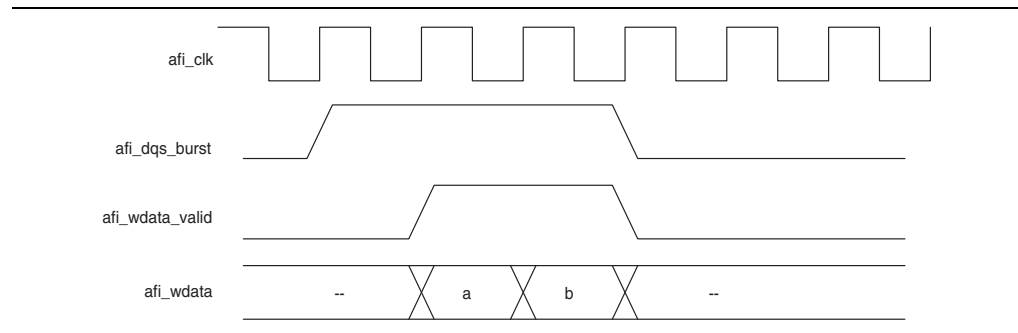


Figure 1–11 shows a full-rate write.

Figure 1–11. Full-Rate Write




After calibration is completed, the sequencer sends the write latency in number of clock cycles to the controller.

Figure 1–12 and Figure 1–13 show writes and reads, where the IP core writes data to and reads from the same address. In each example, `afi_rdata` and `afi_wdata` are aligned with controller clock (`afi_clk`) cycles. All the data in the bit vector is valid at once.

The AFI has the following conventions:

- With the AFI, high and low signals are combined in one signal, so for a single chip select (`afi_cs_n`) interface, `afi_cs_n[1:0]`, location 0 appears on the memory bus on one `mem_clk` cycle and location 1 on the next `mem_clk` cycle.

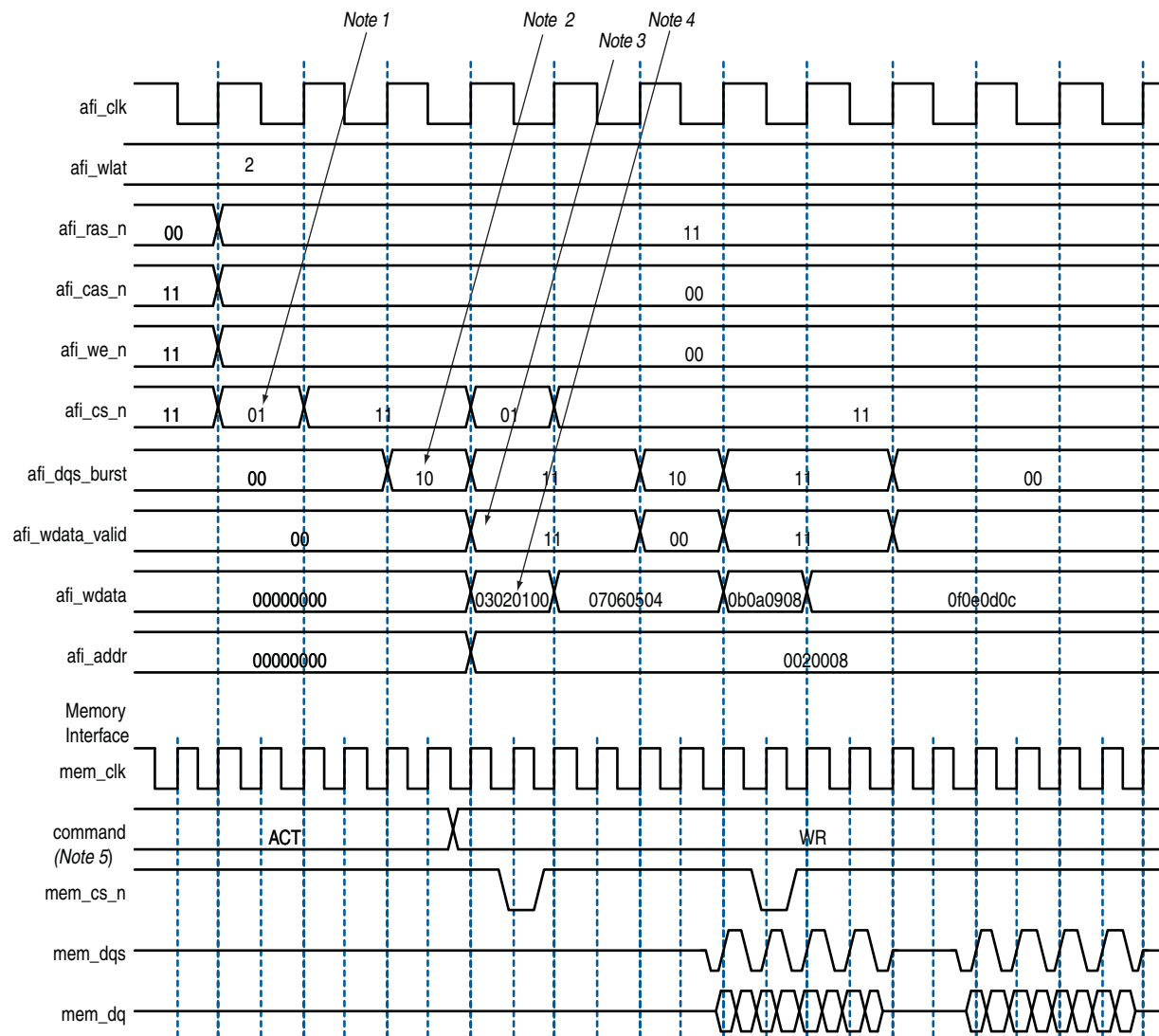
 This convention is maintained for all signals so for an 8 bit memory interface, the write data (`afi_wdata`) signal is `afi_wdata[31:0]`, where the first data on the DQ pins is `afi_wdata[7:0]`, then `afi_wdata[15:8]`, then `afi_wdata[23:16]`, then `afi_wdata[31:24]`.

- Spaced reads and writes have the following definitions:
 - Spaced writes—write commands separated by a gap of one controller clock (`afi_clk`) cycle.
 - Spaced reads—read commands separated by a gap of one controller clock (`afi_clk`) cycle.

Figure 1–12 through Figure 1–13 assume the following general points:

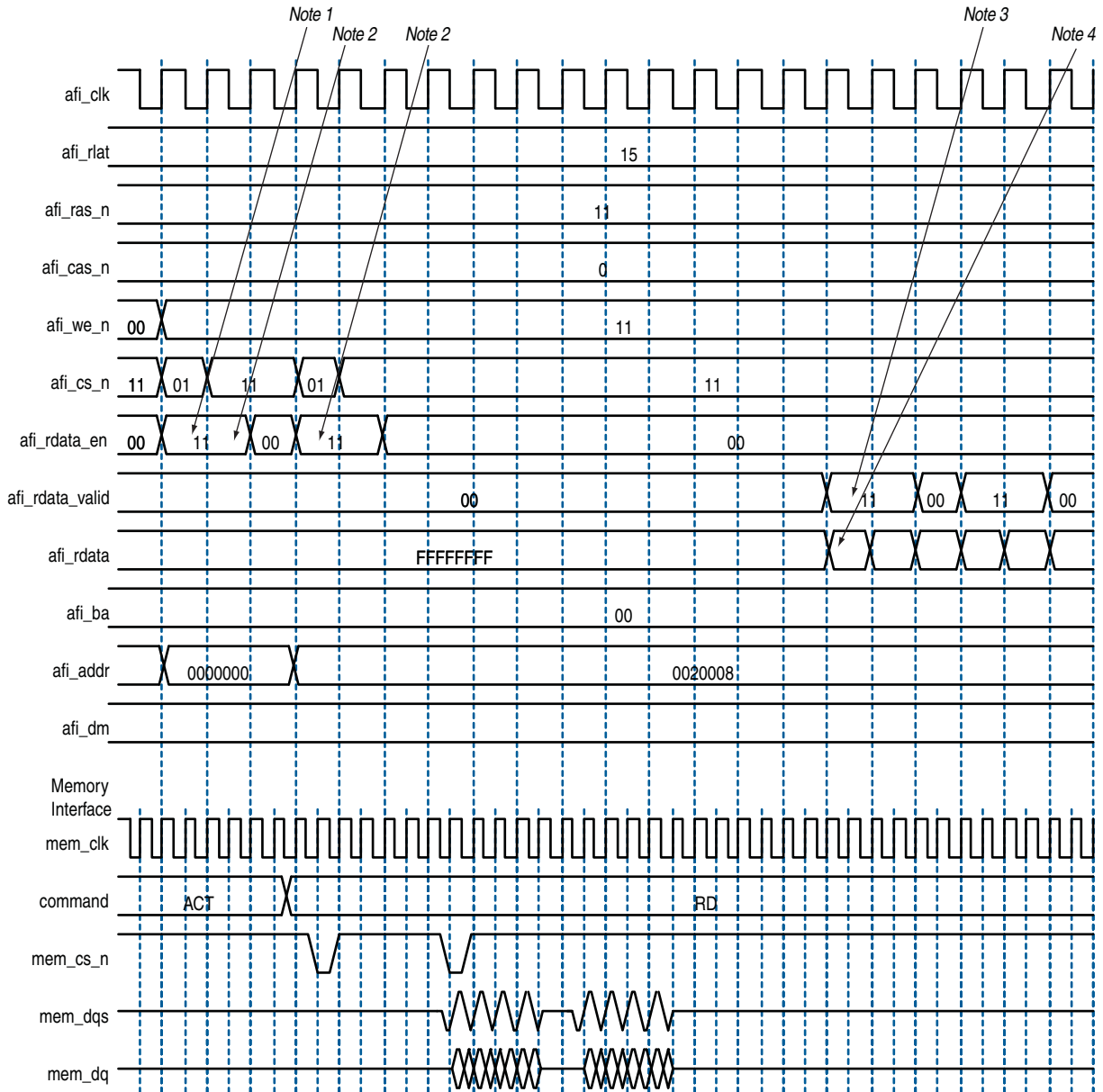
- The burst length is four.
- An 8-bit interface with one chip select.
- The data for one controller clock (`afi_clk`) cycle represents data for two memory clock (`mem_clk`) cycles (half-rate interface).

Figure 1-12. Word-Aligned Writes



Notes to Figure 1-12:

- (1) To show the even alignment of **afi_cs_n**, expand the signal (this convention applies for all other signals).
- (2) The **afi_dqs_burst** must go high one memory clock cycle before **afi_wdata_valid**. Compare with the word-unaligned case.
- (3) The **afi_wdata_valid** is asserted two **afi_wlat** controller clock (**afi_clk**) cycles after chip select (**afi_cs_n**) is asserted. The **afi_wlat** indicates the required write latency in the system. The value is determined during calibration and is dependant upon the relative delays in the address and command path and the write datapath in both the PHY and the external DDR SDRAM subsystem. The controller must drive **afi_cs_n** and then wait **afi_wlat** (two in this example) **afi_clks** before driving **afi_wdata_valid**.
- (4) Observe the ordering of write data (**afi_wdata**). Compare this to data on the **mem_dq** signal.
- (5) In all waveforms a command record is added that combines the memory pins **ras_n**, **cas_n** and **we_n** into the current command that is issued. This command is registered by the memory when chip select (**mem_cs_n**) is low. The important commands in the presented waveforms are WR = write, ACT = activate.

Figure 1-13. Word-Aligned Reads**Notes to Figure 1-13:**

- (1) For AFI, **afi_rdata_en** is required to be asserted one memory clock cycle before chip select (**afi_cs_n**) is asserted. In the half-rate **afi_clk** domain, this requirement manifests as the controller driving 11 (as opposed to the 01) on **afi_rdata_en**.
- (2) AFI requires that **afi_rdata_en** is driven for the duration of the read. In this example, it is driven to 11 for two half-rate **afi_clks**, which equates to driving to 1, for the four memory clock cycles of this four-beat burst.
- (3) The **afi_rdata_valid** returns 15 (**afi_rlat**) controller clock (**afi_clk**) cycles after **afi_rdata_en** is asserted. Returned is when the **afi_rdata_valid** signal is observed at the output of a register within the controller. A controller can use the **afi_rlat** value to determine when to register to returned data, but this is unnecessary as the **afi_rdata_valid** is provided for the controller to use as an enable when registering read data.
- (4) Observe the alignment of returned read data with respect to data on the bus.

Using a Custom Controller

By default, the UniPHY-based external memory interface IP cores are delivered with both the PHY and the memory controller integrated. If you want to use your own custom controller with the UniPHY PHY, check the **Generate PHY only** box on the **PHY Settings** tab of the parameter editor.

The AFI interface is exposed at the top-level of the generated IP core; you can connect the AFI interface to your custom controller.

When you enable **Generate PHY only**, the generated example designs include the memory controller appropriately instantiated to mediate read/write commands from the traffic generator to the PHY-only IP.



For information on the AFI protocol, refer to the [AFI 3.0 Specification](#).



For information on the example designs, refer to [Traffic Generator and BIST Engine](#) in chapter 7 of this section.

Using a Vendor-Specific Memory Model

You can replace the Altera-supplied memory model with a vendor-specific memory model. In general, you may find vendor-specific models to be standardized, thorough, and well supported, but sometimes more complex to setup and use.

If you do want to replace the Altera-supplied memory model with a vendor-supplied memory model, observe the following guidelines:

- Ensure that the vendor-supplied memory model that you have is correct for your memory device.
- Disconnect all signals from the default memory model and reconnect them to the vendor-supplied memory model.
- If you intend to run simulation from the Quartus II software, ensure that the **.qip** file points to the vendor-supplied memory model.



For related information, refer to [Simulating Memory IP](#) in volume 2 of the *External Memory Interface Handbook*.

AFI 3.0 Specification

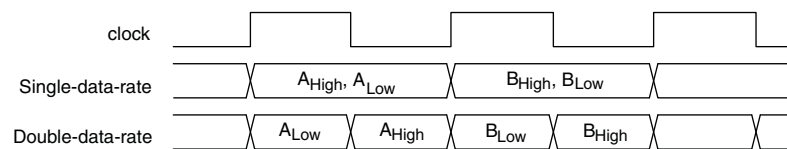
The Altera AFI interface, which is loosely based on the DDR-PHY Interface (DFI) specification, defines communication between the controller and physical layer (PHY) in the external memory interface. The following sections describe the AFI implementation and the AFI signals.

Implementation

The AFI interface is a single-data-rate interface, meaning that data is transferred on the rising edge of each clock cycle. Most memory interfaces, however, operate at double-data-rate, transferring data on both the rising and falling edges of the clock signal. If the AFI interface is to directly control a double-data-rate signal, two single-data-rate bits must be transmitted on each clock cycle; the PHY then sends out one bit on the rising edge of the clock and one bit on the falling edge.

The AFI convention is to send the low part of the data first and the high part second, as shown in [Figure 1-14](#).

Figure 1-14. Single versus Double Data Rate Transfer



Bus Width and AFI Ratio

In cases where the AFI clock frequency is one-half or one-quarter of the memory clock frequency, the AFI data must be twice or four times as wide, respectively, as the corresponding memory data. The ratio between AFI clock and memory clock frequencies is referred to as the AFI ratio. (A half-rate AFI interface has an AFI ratio of 2, while a quarter-rate interface has an AFI ratio of 4.)

In general, the width of the AFI signal depends on the following three factors:

- The size of the equivalent signal on the memory interface. For example, if a [15:0] is a DDR3 address input and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_addr` bus will be 16-bits wide.
- The data rate of the equivalent signal on the memory interface. For example, if `d[7:0]` is a double-data-rate QDR II input data bus and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_write_data` bus will be 16-bits wide.
- The AFI ratio. For example, if `cs_n` is a single-bit DDR3 chip select input and the AFI clock runs at half the speed of the memory interface, the equivalent `afi_cs_n` bus will be 2-bits wide.

The following formula summarizes the three factors described above:

$$\text{AFI_width} = \text{memory_width} * \text{signal_rate} * \text{AFI_RATE_RATIO}$$



The above formula is a general rule, but not all signals obey it. For definite signal-size information, refer to the specific table.

AFI Parameters

Table 1-5 through Table 1-13 list the AFI parameters grouped according to their functions.

Not all parameters are used for all protocols.

Parameters Affecting Bus Width

The following parameters affect the width of AFI signal buses. Parameters prefixed by MEM_IF_ refer to the signal size at the interface between the PHY and memory device.

Table 1-5. Ratio Parameters

Parameter Name	Description
AFI_RATE_RATIO	The ratio between the AFI clock frequency and the memory clock frequency. For full-rate interfaces this value is 1, for half-rate interfaces the value is 2, and for quarter-rate interfaces the value is 4.
DATA_RATE_RATIO	The number of data bits transmitted per clock cycle. For single-data rate protocols this value is 1, and for double-data rate protocols this value is 2.
ADDR_RATE_RATIO	The number of address bits transmitted per clock cycle. For single-data rate address protocols this value is 1, and for double-data rate address protocols this value is 2.

Table 1-6. Memory Interface Parameters

Parameter Name	Description
MEM_IF_ADDR_WIDTH	The width of the address bus on the memory device(s).
MEM_IF_BANKADDR_WIDTH	The width of the bank address bus on the interface to the memory device(s). Typically, the \log_2 of the number of banks.
MEM_IF_CS_WIDTH	The number of chip selects on the interface to the memory device(s).
MEM_IF_WRITE_DQS_WIDTH	The number of DQS (or write clock) signals on the write interface. For example, the number of DQS groups.
MEM_IF_CLK_PAIR_COUNT	The number of CK/CK# pairs.
MEM_IF_DQ_WIDTH	The number of DQ signals on the interface to the memory device(s). For single-ended interfaces such as QDR II, this value is the number of D or Q signals.
MEM_IF_DM_WIDTH	The number of data mask pins on the interface to the memory device(s).

Table 1-7. Derived AFI Parameters (Part 1 of 2)

Parameter Name	Derivation Equation
AFI_ADDR_WIDTH	$\text{MEM_IF_ADDR_WIDTH} * \text{AFI_RATE_RATIO} * \text{ADDR_RATE_RATIO}$
AFI_BANKADDR_WIDTH	$\text{MEM_IF_BANKADDR_WIDTH} * \text{AFI_RATE_RATIO} * \text{ADDR_RATE_RATIO}$

Table 1-7. Derived AFI Parameters (Part 2 of 2)

Parameter Name	Derivation Equation
AFI_CONTROL_WIDTH	$AFI_RATE_RATIO * ADDR_RATE_RATIO$
AFI_CS_WIDTH	$MEM_IF_CS_WIDTH * AFI_RATE_RATIO$
AFI_DM_WIDTH	$MEM_IF_DM_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO$
AFI_DQ_WIDTH	$MEM_IF_DQ_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO$
AFI_WRITE_DQS_WIDTH	$MEM_IF_WRITE_DQS_WIDTH * AFI_RATE_RATIO$
AFI_LAT_WIDTH	6
AFI_RLAT_WIDTH	AFI_LAT_WIDTH
AFI_WLAT_WIDTH	$AFI_LAT_WIDTH * MEM_IF_WRITE_DQS_WIDTH$
AFI_CLK_PAIR_COUNT	MEM_IF_CLK_PAIR_COUNT

AFI Signals

The following tables list the AFI signals grouped according to their functions.

In each table, the **direction** column denotes the direction of the signal relative to the PHY. For example, a signal defined as an output passes out of the PHY to the controller. The AFI specification does not include any bidirectional signals.

Not all signals are used for all protocols

Clock and Reset Signals

The AFI interface provides up to two clock signals and an asynchronous reset signal.

Table 1-8. Clock and Reset Signals

Signal Name	Direction	Width	Description
afi_clk	Output	1	Clock with which all data exchanged on the AFI bus is synchronized. In general, this clock is referred to as full-rate, half-rate, or quarter-rate, depending on the ratio between the frequency of this clock and the frequency of the memory device clock.
afi_half_clk	Output	1	Clock signal that runs at half the speed of the afi_clk. The controller uses this signal when the half-rate bridge feature is in use. This signal is optional.
afi_reset_n	Output	1	Asynchronous reset output signal. You must synchronize this signal to the clock domain in which you use it.

Address and Command Signals

The address and command signals encode read/write/configuration commands to send to the memory device. The address and command signals are single-data rate signals.

Table 1–9. Address and Command Signals

Signal Name	Direction	Width	Description
afi_ba	Input	AFI_BANKADDR_WIDTH	Bank address.
afi_cke	Input	AFI_CS_WIDTH	Clock enable.
afi_cs_n	Input	AFI_CS_WIDTH	Chip select signal. (The number of chip selects may not match the number of ranks; for example, RDIMMs use 2 chip select signals for both single-rank and dual-rank configurations. Consult your memory device datasheet for information about chip select signal width.)
afi_ras_n	Input	AFI_CONTROL_WIDTH	RAS# (for DDR2 and DDR3 memory devices.)
afi_we_n	Input	AFI_CONTROL_WIDTH	WE# (for DDR2, DDR3, and RLDRAM II memory devices.)
afi_cas_n	Input	AFI_CONTROL_WIDTH	CAS# (for DDR2 and DDR3 memory devices.)
afi_ref_n	Input	AFI_CONTROL_WIDTH	REF# (for RLDRAM II memory devices.)
afi_rst_n	Input	AFI_CONTROL_WIDTH	RESET# (for DDR3 memory devices.)
afi_odt	Input	AFI_CS_WIDTH	On-die termination signal for DDR2 and DDR3 memory devices. (Do not confuse this memory device signal with the FPGA's internal on-chip termination signal.)
afi_mem_clk_disable	Input	AFI_CLK_PAIR_COUNT	When this signal is asserted, mem_clk and mem_clk_n are disabled. This signal is used in low-power mode.
afi_wps_n	Output	AFI_CS_WIDTH	WPS (for QDR II/II+ memory devices.)
afi_rps_n	Output	AFI_CS_WIDTH	RPS (for QDR II/II+ memory devices.)

Write Data Signals

The signals described in this section control the data, data mask, and strobe signals passed to the memory device during write operations.

Table 1–10. Write Data Signals

Signal Name	Direction	Width	Description
afi_dqs_burst	Input	AFI_WRITE_DQS_WIDTH	Controls the enable on the strobe (DQS) pins for DDR2 and DDR3 memory devices. When this signal is asserted, mem_dqs and mem_dqsn are driven. This signal must be asserted before afi_wdata_valid to implement the write preamble, and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
afi_wdata_valid	Input	AFI_WRITE_DQS_WIDTH	Write data valid signal. This signal controls the output enable on the data and data mask pins.
afi_wdata	Input	AFI_DQ_WIDTH	Write data signal to send to the memory device at double-data rate. This signal controls the PHY's mem_dq output.
afi_dm	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_dm signal for DDR2, DDR3, and RLDRAM II memory devices.)
afi_bws_n	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_bws_n signal for QDR II/II+ memory devices.)

Read Data Signals

The signals described in this section control the data sent from the memory device during read operations.

Table 1–11. Read Data Signals

Signal Name	Direction	Width	Description
afi_rdata_en	Input	AFI_RATE_RATIO	Read data enable. Indicates that the memory controller is currently performing a read operation. This signal is held high only for cycles of relevant data (read data masking). If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata_en_full	Input	AFI_RATE_RATIO	Read data enable full. Indicates that the memory controller is currently performing a read operation. This signal is held high for the entire read burst. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata	Output	AFI_DQ_WIDTH	Read data from the memory device. This data is considered valid only when afi_rdata_valid is asserted by the PHY.
afi_rdata_valid	Output	AFI_RATE_RATIO	Read data valid. When asserted, this signal indicates that the afi_rdata bus is valid. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).

Calibration Status Signals

The PHY instantiates a sequencer which calibrates the memory interface with the memory device and some internal components such as read FIFOs and valid FIFOs. The sequencer reports the results of the calibration process to the controller through the AFI interface. This section describes the calibration status signals.

Table 1–12. Calibration Status Signals (Part 1 of 2)

Signal Name	Direction	Width	Description
afi_cal_success	Output	1	Asserted to indicate that calibration has completed successfully.
afi_cal_fail	Output	1	Asserted to indicate that calibration has failed.

Table 1–12. Calibration Status Signals (Part 2 of 2)

Signal Name	Direction	Width	Description
afi_cal_req	Input	1	Effectively a synchronous reset for the sequencer. When this signal is asserted, the sequencer returns to the reset state; when this signal is released, a new calibration sequence begins.
afi_wlat	Output	AFI_WLAT_WIDTH	The required write latency in afi_clk cycles, between address/command and write data being issued at the PHY/controller interface. The afi_wlat value can be different for different groups; each group's write latency can range from 0 to 63. If write latency is the same for all groups, only the lowest 6 bits are required.
afi_rlat	Output	AFI_RLAT_WIDTH	The required read latency in afi_clk cycles between address/command and read data being returned to the PHY/controller interface. Values can range from 0 to 63.

Tracking Management Signals

When tracking management is enabled, the sequencer can take control over the AFI interface at given intervals, and issue commands to the memory device to track the internal DQS Enable signal alignment to the DQS signal returning from the memory device. The tracking management portion of the AFI interface provides a means for the sequencer and the controller to exchange handshake signals.

Table 1–13. Tracking Management Signals

Signal Name	Direction	Width ¹	Description
afi_ctl_refresh_done	Input	MEM_IF_CS_WIDTH	Signal from controller to tracking manager, indicating that a refresh has occurred; also acts as an acknowledge signal.
afi_seq_busy	Output	MEM_IF_CS_WIDTH	Stall request and acknowledge signal from sequencer to controller when DQS tracking is needed and in progress.
afi_ctl_long_idle	Input	MEM_IF_CS_WIDTH	Signal from controller to tracking manager, indicating that it has exited low power state without a refresh; also acts as an acknowledge signal.

Register Maps

Table 1-14 shows the overall register mapping for the DDR2 and DDR3 SDRAM Controller with UniPHY.

Table 1-14. Register Map

Address	Description
UniPHY Register Map	
0x001	Reserved.
0x004	UniPHY status register 0.
0x005	UniPHY status register 1.
0x006	UniPHY status register 2.
0x007	UniPHY memory initialization parameters register 0.
Controller Register Map	
0x100	Reserved.
0x110	Controller status and configuration register.
0x120	Memory address size register 0.
0x121	Memory address size register 1.
0x122	Memory address size register 2.
0x123	Memory timing parameters register 0.
0x124	Memory timing parameters register 1.
0x125	Memory timing parameters register 2.
0x126	Memory timing parameters register 3.
0x130	ECC control register.
0x131	ECC status register.
0x132	ECC error address register.

UniPHY Register Map

The UniPHY register map allows you to control the memory components' mode register settings. Table 1-15 shows the register map for UniPHY.

Table 1-15. UniPHY Register Map (Part 1 of 2)

Address	Bit	Name	Default	Access	Description
0x001	15:0	Reserved.	0	—	Reserved for future use.
	31:16	Reserved.	0	—	Reserved for future use.
0x002	15:0	Reserved.	0	—	Reserved for future use.
	31:16	Reserved.	0	—	Reserved for future use.

Table 1–15. UniPHY Register Map (Part 2 of 2)

Address	Bit	Name	Default	Access	Description
0x004	0	SOFT_RESET	—	Write only	Initiate a soft reset of the interface. This bit is automatically deasserted after reset.
	23:1	Reserved.	0	—	Reserved for future use.
	24	AFI_CAL_SUCCESS	—	Read only	Reports the value of the UniPHY <code>afi_cal_success</code> . Writing to this bit has no effect.
	25	AFI_CAL_FAIL	—	Read only	Reports the value of the UniPHY <code>afi_cal_fail</code> . Writing to this bit has no effect.
	26	Reserved.	0	—	Reserved for future use.
	31:27	Reserved.	0	—	Reserved for future use.
0x005	7:0	Reserved.	0	—	Reserved for future use.
	15:8	Reserved.	0	—	Reserved for future use.
	23:16	Reserved.	0	—	Reserved for future use.
	31:24	Reserved.	0	—	Reserved for future use.
0x006	7:0	INIT_FAILING_STAGE	—	Read only	Initial failing error stage of calibration. Only applicable if <code>AFI_CAL_FAIL=1</code> .
	15:8	Reserved.	0	—	Reserved for future use.
	23:16	INIT_FAILING_GROUP	—	Read only	Initial failing error group of calibration. Only applicable if <code>AFI_CAL_FAIL=1</code> .
	31:24	Reserved.	0	—	Reserved for future use.
0x007	31:0	DQS_DETECT	—	Read only	Identifies if DQS edges have been identified for each of the groups. Each bit corresponds to one DQS group.
0x008 (DDR2)	1:0	RTT_NOM	—	Read only	Rtt (nominal) setting of the DDR2 Extended Mode Register used during memory initialization.
	31:2	Reserved.	0	—	Reserved for future use.
0x008 (DDR3)	2:0	RTT_NOM	—		Rtt (nominal) setting of the DDR3 MR1 mode register used during memory initialization.
	4:3	Reserved.	0		Reserved for future use.
	6:5	ODS	—		Output driver impedance control setting of the DDR3 MR1 mode register used during memory initialization.
	8:7	Reserved.	0		Reserved for future use.
	10:9	RTT_WR	—		Rtt (writes) setting of the DDR3 MR2 mode register used during memory initialization.
	31:11	Reserved.	0		Reserved for future use.

Controller Register Map

The controller register map allows you to control the memory controller settings. Table 1-16 shows the register map for the controller.

Table 1-16. Controller Register Map (Part 1 of 4)

Address	Bit	Name	Default	Access	Description
0x100	0	Reserved.	0	—	Reserved for future use.
	1	Reserved.	0	—	Reserved for future use.
	2	Reserved.	0	—	Reserved for future use.
	7:3	Reserved.	0	—	Reserved for future use.
	13:8	Reserved.	0	—	Reserved for future use.
	30:14	Reserved.	0	—	Reserved for future use.
0x110	15:0	AUTO_PD_CYCLES	0x0	Read write	The number of idle clock cycles after which the controller should place the memory into power-down mode. The controller is considered to be idle if there are no commands in the command queue. Setting this register to 0 disables the auto power-down mode. The default value of this register depends on the values set during the generation of the design.
	16	Reserved.	0	—	Reserved for future use.
	17	Reserved.	0	—	Reserved for future use.
	18	Reserved.	0	—	Reserved for future use.
	19	Reserved.	0	—	Reserved for future use.
	21:20	ADDR_ORDER	00	Read write	00 - Chip, row, bank, column. 01 - Chip, bank, row, column. 10 - reserved for future use. 11 - Reserved for future use.
	22	Reserved.	0	—	Reserved for future use.
	24:23	Reserved.	0	—	Reserved for future use.
30:24	Reserved.	0	—	Reserved for future use.	
0x120	7:0	Column address width	—	Read write	The number of column address bits for the memory devices in your memory interface. The range of legal values is 7-12.
	15:8	Row address width	—	Read write	The number of row address bits for the memory devices in your memory interface. The range of legal values is 12-16.
	19:16	Bank address width	—	Read write	The number of bank address bits for the memory devices in your memory interface. The range of legal values is 2-3.
	23:20	Chip select address width	—	Read write	The number of chip select address bits for the memory devices in your memory interface. The range of legal values is 0-2. If there is only one single chip select in the memory interface, set this bit to 0.
	31:24	Reserved.	0	—	Reserved for future use.

Table 1–16. Controller Register Map (Part 2 of 4)

Address	Bit	Name	Default	Access	Description
0x121	31:0	Data width representation (word)	—	Read only	The number of DQS bits in the memory interface. This bit can be used to derive the width of the memory interface by multiplying this value by the number of DQ pins per DQS pin (typically 8).
0x122	7:0	Chip select representation	—	Read only	The number of chip select in binary representation. For example, a design with 2 chip selects has the value of 00000011.
	31:8	Reserved.	0	—	Reserved for future use.
0x123	3:0	t_{RCD}	—	Read write	The activate to read or write a timing parameter. The range of legal values is 2-11 cycles.
	7:4	t_{RRD}	—	Read write	The activate to activate a timing parameter. The range of legal values is 2-8 cycles.
	11:8	t_{RP}	—	Read write	The precharge to activate a timing parameter. The range of legal values is 2-11 cycles.
	15:12	t_{MRD}	—	Read write	The mode register load time parameter. This value is not used by the controller, as the controller derives the correct value from the memory type setting.
	23:16	t_{RAS}	—	Read write	The activate to precharge a timing parameter. The range of legal values is 4-29 cycles.
	31:24	t_{RC}	—	Read write	The activate to activate a timing parameter. The range of legal values is 8-40 cycles.
0x124	3:0	t_{WTR}	—	Read write	The write to read a timing parameter. The range of legal values is 1-10 cycles.
	7:4	t_{RTP}	—	Read write	The read to precharge a timing parameter. The range of legal values is 2-8 cycles.
	15:8	t_{FAW}	—	Read write	The four-activate window timing parameter. The range of legal values is 6-32 cycles.
	31:16	Reserved.	0	—	Reserved for future use.
0x125	15:0	t_{REFI}	—	Read write	The refresh interval timing parameter. The range of legal values is 780-6240 cycles.
	23:16	t_{RFC}	—	Read write	The refresh cycle timing parameter. The range of legal values is 12-88 cycles.
	31:24	Reserved.	0	—	Reserved for future use.
0x126	3:0	Reserved.	0	—	Reserved for future use.
	7:4	Reserved.	0	—	Reserved for future use.
	11:8	Reserved.	0	—	Reserved for future use.
	15:12	Reserved.	0	—	Reserved for future use.
	23:16	Burst Length	—	Read write	Value must match memory burst length.
	31:24	Reserved.	0	—	Reserved for future use.

Table 1–16. Controller Register Map (Part 3 of 4)

Address	Bit	Name	Default	Access	Description
0x130	0	ENABLE_ECC	1	Read write	When this bit equals 1, it enables the generation and checking of ECC. This bit is only active if ECC was enabled during IP parameterization.
	1	ENABLE_AUTO_CORR	—	Read write	When this bit equals 1, it enables auto-correction when a single-bit error is detected.
	2	GEN_SBE	0	Read write	When this bit equals 1, it enables the deliberate insertion of single-bit errors, bit 0, in the data written to memory. This bit is used only for testing purposes.
	3	GEN_DBE	0	Read write	When this bit equals 1, it enables the deliberate insertion of double-bit errors, bits 0 and 1, in the data written to memory. This bit is used only for testing purposes.
	4	ENABLE_INTR	1	Read write	When this bit equals 1, it enables the interrupt output.
	5	MASK_SBE_INTR	0	Read write	When this bit equals 1, it masks the single-bit error interrupt.
	6	MASK_DBE_INTR	0	Read write	When this bit equals 1, it masks the double-bit error interrupt
	7	CLEAR	0	Read write	When this bit equals 1, writing to this self-clearing bit clears the interrupt signal, and the error status and error address registers.
	8	MASK_CORDROP_INTR	0	Read write	When this bit equals 1, the dropped autocorrection error interrupt is dropped.
	9	Reserved.	0	—	Reserved for future use.
0x131	0	SBE_ERROR	0	Read only	Set to 1 when any single-bit errors occur.
	1	DBE_ERROR	0	Read only	Set to 1 when any double-bit errors occur.
	2	CORDROP_ERROR	0	Read only	Value is set to 1 when any controller-scheduled autocorrections are dropped.
	7:3	Reserved.	0	—	Reserved for future use.
	15:8	SBE_COUNT	0	Read only	Reports the number of single-bit errors that have occurred since the status register counters were last cleared.
	23:16	DBE_COUNT	0	Read only	Reports the number of double-bit errors that have occurred since the status register counters were last cleared.
	31:24	CORDROP_COUNT	0	Read only	Reports the number of controller-scheduled autocorrections dropped since the status register counters were last cleared.
0x132	31:0	ERR_ADDR	0	Read only	The address of the most recent ECC error. This address is a memory burst-aligned local address.

Table 1-16. Controller Register Map (Part 4 of 4)

Address	Bit	Name	Default	Access	Description
0x133	31:0	CORDROP_ADDR	0	Read only	The address of the most recent autocorrection that was dropped. This is a memory burst-aligned local address.
0x134	0	REORDER_DATA	—	Read write	
	15:1	Reserved.	0	—	Reserved for future use.
	23:16	STARVE_LIMIT	0	Read write	Number of commands that can be served before a starved command.
	31:24	Reserved.	0	—	Reserved for future use.

Efficiency Monitor and Protocol Checker

The Efficiency Monitor and Protocol Checker is a feature available with the DDR2 and DDR3 SDRAM controllers with UniPHY and the RLDRAM II Controller with UniPHY. The Efficiency Monitor and Protocol Checker allows measurement of traffic efficiency on the Avalon-MM bus between the controller and user logic, measures read latencies, and checks the legality of Avalon commands passed from the master. The following sections describe the parts of this feature.

Efficiency Monitor

The Efficiency Monitor reports read and write throughput on the controller input, by counting command transfers and wait times, and making that information available to the External Memory Interface Toolkit via an Avalon slave port. This information may be useful to you when experimenting with advanced controller settings, such as command look ahead depth and burst merging.

Protocol Checker

The Protocol Checker checks the legality of commands on the controller's input interface against Altera's Avalon interface specification, and sets a flag in a register on an Avalon slave port if an illegal command is detected.


Read Latency Counter

The Read Latency Counter measures the minimum and maximum wait times for read commands to be serviced on the Avalon bus. Each read command is time-stamped and placed into a FIFO buffer upon arrival, and latency is determined by comparing that timestamp to the current time when the first beat of the returned read data is provided back to the master.

Using the Efficiency Monitor and Protocol Checker

To include the Efficiency Monitor and Protocol Checker when you generate your IP core, on the **Diagnostics** tab in the parameter editor, turn on **Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface**.

To see the results of the data compiled by the Efficiency Monitor and Protocol Checker, use the External Memory Interface Toolkit.

 For information on the External Memory Interface Toolkit, refer to [UniPHY External Memory Interface Debug Toolkit](#), in section 2, chapter 4 of this volume. For information about the Avalon interface, refer to [Avalon Interface Specifications](#).

Avalon CSR Slave and JTAG Memory Map

Table 1–17 summarizes the memory map of registers inside the Efficiency Monitor and Protocol Checker; this information is only of interest if you want to communicate directly with the Efficiency Monitor and Protocol Checker without using the External Memory Interface Toolkit. This CSR map is not part of the UniPHY CSR map.

Table 1–17. Avalon CSR Slave and JTAG Memory Map (Part 1 of 2)

Address	Bit	Name	Default	Access	Description
0x01	31:0	Reserved	0	—	Used internally by EMIF Toolkit to identify Efficiency Monitor type.
0x02	31:0	Reserved	0	—	Used internally by EMIF Toolkit to identify Efficiency Monitor version.
0x08	0	Efficiency Monitor reset	—	Write only	Write a 0 to reset.
	7:1	Reserved	—	—	Reserved for future use.
	8	Protocol Checker reset	—	Write only	Write a 0 to reset.
	15:9	Reserved	—	—	Reserved for future use.
	16	Start/stop Efficiency Monitor	—	Read/Write	Starting and stopping stastics gathering.
	23:17	Reserved	—	—	Reserved for future use.
0x10	31:24	Efficiency Monitor status	—	Read Only	bit 0: Efficiency Monitor stopped bit 1: Waiting for start of pattern bit 2: Running bit 3: Counter saturation
	15:0	Efficiency Monitor address width	—	Read Only	Address width of the Efficiency Monitor.
0x11	31:16	Efficiency Monitor data width	—	Read Only	Data Width of the Efficiency Monitor.
	15:0	Efficiency Monitor byte enable	—	Read Only	Byte enable width of the Efficiency Monitor.
0x14	31:16	Efficiency Monitor burst count width	—	Read Only	Burst count width of the Efficiency Monitor.
	31:0	Cycle counter	—	Read Only	Clock cycle counter for the Efficiency Monitor. Lists the number of clock cycles elapsed before the Efficiency Monitor stopped.
0x18	31:0	Transfer counter	—	Read Only	Counts any read or write data transfer cycle.
0x1C	31:0	Write counter	—	Read Only	Counts write requests, including those during bursts.

Table 1–17. Avalon CSR Slave and JTAG Memory Map (Part 2 of 2)

Address	Bit	Name	Default	Access	Description
0x20	31:0	Read counter	—	Read Only	Counts read requests.
0x24	31:0	Readtotal counter	—	Read Only	Counts read requests (total burst requests).
0x28	31:0	NTC waitrequest counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave wait request high.
0x2C	31:0	NTC noreaddatavalid counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave not having read data.
0x30	31:0	NTC master write idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command, or pause in write burst.
0x34	31:0	NTC master idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command anytime.
0x40	31:0	Read latency min	—	Read Only	The lowest read latency value.
0x44	31:0	Read latency max	—	Read Only	The highest read latency value.
0x48	31:0	Read latency total [31:0]	—	Read Only	The lower 32 bits of the total read latency.
0x49	31:0	Read latency total [63:32]	—	Read Only	The upper 32 bits of the total read latency.
0x50	7:0	Illegal command	—	Read Only	Bits used to indicate which illegal command has occurred. Each bit represents a unique error.
	31:8	Reserved	—		Reserved for future use.

UniPHY Calibration Stages

This section describes the calibration stages performed by the DDR2 and DDR3 SDRAM, QDR II and QDR II+ SRAM, and RLDRAM II Controllers with UniPHY. This information is useful in debugging calibration failures. The section includes an overview of calibration, explanation of the calibration stages, and a list of generated calibration signals. The information in this section applies only to the Nios® II-based sequencer used in the DDR2 and DDR3 SDRAM Controllers with UniPHY versions 10.0 and later, and, optionally, in the QDR II and QDR II+ SRAM and RLDRAM II Controllers with UniPHY version 11.0 and later. The information in this section applies to the Arria II GZ, Arria V, Cyclone V, Stratix III, Stratix IV, and Stratix V device families.



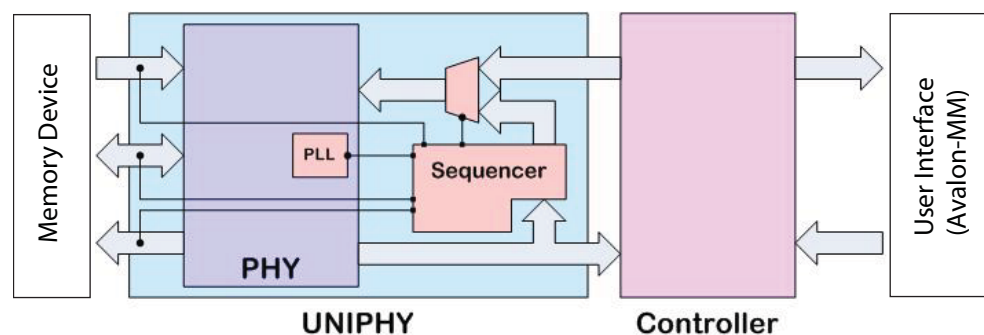
For QDR II and QDR II+ SRAM and RLDRAM II Controllers with UniPHY version 11.0 and later, you have the option to select either the RTL-based sequencer or the Nios® II-based sequencer. Generally, choose the RTL-based sequencer when area is the major consideration, and choose the Nios II-based sequencer when performance is the major consideration.

Overview

Calibration configures the memory interface (PHY and I/Os) so that data can pass reliably to and from memory. The sequencer illustrated in Figure 1-15 calibrates the PHY and the I/Os. To correctly transmit data between a memory device and the FPGA at high speed, the data must be center-aligned with the data clock.

Calibration also determines the delay settings needed to center-align the various data signals with respect to their clocks. I/O delay chains implement the required delays in accordance with the computed alignments. The Nios II-based sequencer performs two major tasks: FIFO buffer calibration and I/O calibration. FIFO buffer calibration adjusts FIFO lengths and I/O calibration adjusts any delay chain and phase settings to center-align data signals with respect to clock signals for both reads and writes. When the calibration process completes, the sequencer shuts off and passes control to the memory controller.


Figure 1-15. Sequencer in Memory Interface Logic



Calibration Stages

The calibration process begins when the PHY reset signal deasserts and the PLL and DLL lock. The following stages of calibration take place:

1. Read calibration part one—DQS enable calibration (only for DDR2 and DDR3 SDRAM Controllers with UniPHY) and DQ/DQS centering
2. Write calibration part one—Leveling
3. Write calibration part two—DQ/DQS centering
4. Read calibration part two—Read latency minimization

 For multirank calibration, the sequencer transmits every read and write command to each rank in sequence. Each read and write test is successful only if all ranks pass the test. The sequencer calibrates to the intersection of all ranks.

Assumptions

The calibration process assumes the following conditions; if either of these conditions is not true, calibration likely fails in its early stages:

- The address and command paths must be functional; calibration does not tune the address and command paths. (The Quartus II software fully analyzes the timing for the address and command paths, and the slack report is accurate, assuming the correct board timing parameters.)
- At least one bit per group must work before running per-bit-deskew calibration. (This assumption requires that DQ-to-DQS skews be within the recommended 20 ps.)

Memory Initialization

The memory is powered up according to protocol initialization specifications. All ranks power up simultaneously. Once powered, the device is ready to receive mode register load commands. This part of initialization occurs separately for each rank. The sequencer issues mode register set commands on a per-chip-select basis and initializes the memory to the user-specified settings.

Stage 1: Read Calibration Part One—DQS Enable Calibration and DQ/DQS Centering

Read calibration occurs in two parts. Part one is DQS enable calibration with DQ/DQS centering, which happens during stage 1 of the overall calibration process; part two is read latency minimization, which happens during stage 4 of the overall calibration process.

The objectives of DQS enable calibration and DQ/DQS centering are as follows:

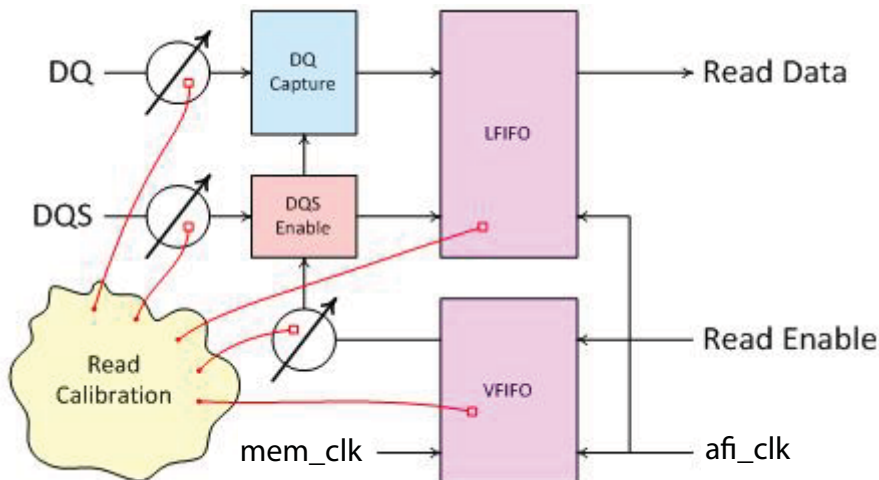
- To calculate when the read data is received after a read command is issued to setup the Data Valid Prediction FIFO (VFIFO) cycle
- To align the input data (DQ) with respect to the clock (DQS) to maximize the read margins (DDR2 and DDR3 only)

DQS enable calibration and DQ/DQS centering consists of the following actions:

- Guaranteed Write
- DQS Enable Calibration
- DQ/DQS Centering

Figure 1-16 illustrates the components in the read data path that the sequencer calibrates in this stage. (The round knobs in the figure represent configurable hardware over which the sequencer has control.)

Figure 1-16. Read Data Path Calibration Model

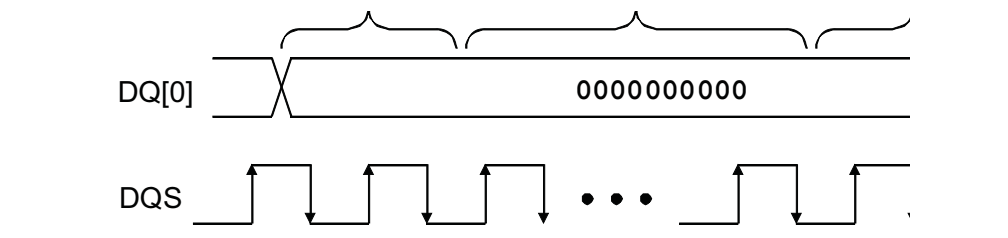


Guaranteed Write

Since initially no communication can be reliably performed with the memory device, the sequencer uses a guaranteed write mechanism to write data into the memory device. (For the QDR II protocol, guaranteed write is not necessary, a simple write mechanism is sufficient.)

The guaranteed write is a write command issued with all data pins, all address and bank pins, and all command pins (except chip select) held constant. The sequencer begins toggling DQS well before the expected latch time at memory and continues to toggle DQS well after the expected latch time at memory. DQ-to-DQS relationship is not a factor at this stage because DQ is held constant. Figure 1-17 illustrates a guaranteed write of zeros.

Figure 1-17. Guaranteed Write of Zeros



The guaranteed write consists of a series of back-to-back writes to alternating columns and banks. For example, for DQ[0] for the DDR3 protocol, the guaranteed write performs the following operations:

- a. Writes a full burst of zeros to bank 0, column 0
- b. Writes a full burst of zeros to bank 0, column 1
- c. Writes a full burst of ones to bank 3, column 0
- d. Writes a full burst of ones to bank 3, column 1

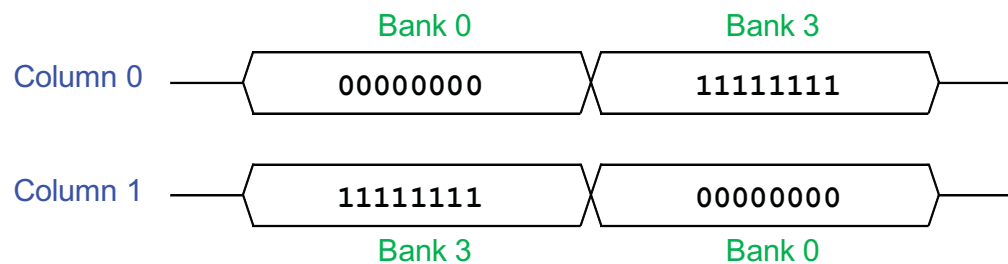
(Different protocols may use different combinations of banks and columns.)

The guaranteed write is followed by back-to-back read operations at alternating banks, effectively producing a stream of zeros followed by a stream of ones, or vice versa. The sequencer uses the zero-to-one and one-to-zero transitions in between the two bursts to identify a correct read operation, as shown in [Figure 1-18](#).


Although the approach described above for pin DQ[0] would work by writing the same pattern to all DQ pins, it is more effective and robust to write (and read) alternating ones and zeros to alternating DQ bits. The value of the DQ bit is still constant across the burst, and the back-to-back read mechanism works exactly as described above, except that odd DQ bits have ones instead of zeros, or vice versa.


The guaranteed write does not ensure a correct DQS-to-memory clock alignment at the memory device—DQS-to-memory clock alignment is performed later, in stage 2 of the calibration process. However, the process of guaranteed write followed by read calibration is repeated several times for different DQS-to-memory clock alignments, to ensure at least one correct alignment is found.

Figure 1-18. Back to Back Reads on pin DQ[0]



DQS Enable Calibration

 The full DQS enable calibration is applicable only for DDR2 and DDR3 protocols; QDR II and RLDRAM protocols use only the VFIFO-based cycle-level calibration, described below.

 Delay and phase values used in this section are examples, for illustrative purposes. Your exact values may vary depending on device and configuration.

DQS enable calibration ensures reliable capture of the DQ signal without glitches on the DQS line. At this point LFIFO is set to its maximum value to guarantee a reliable read from read capture registers to the core. Read latency is minimized later.

DQS enable calibration controls the timing of the enable signal using 3 independent controls: a cycle-based control (the VFIFO), a phase control, and a delay control. The VFIFO selects the cycle by shifting the controller-generated read data enable signal, `rdata_en`, by a number of full-rate clock cycles. The phase is controlled using the DLL, while the delays are adjusted using a sequence of individual delay taps. The resolution of the phase and delay controls varies with family and configuration, but is approximately 45° for the phase, and between 10 and 50 picoseconds for the delays.

The sequencer finds the two edges of the DQS enable window by searching the space of cycles, phases, and delays (an exhaustive search can usually be avoided by initially assuming the window is at least one phase wide). During the search, to test the current settings, the sequencer issues back-to-back reads from column 0 of bank 0 and bank 3, and column 1 of bank 0 and bank 3, as shown in Figure 1-18. Two full bursts are read and compared with the reference data for each phase and delay setting.

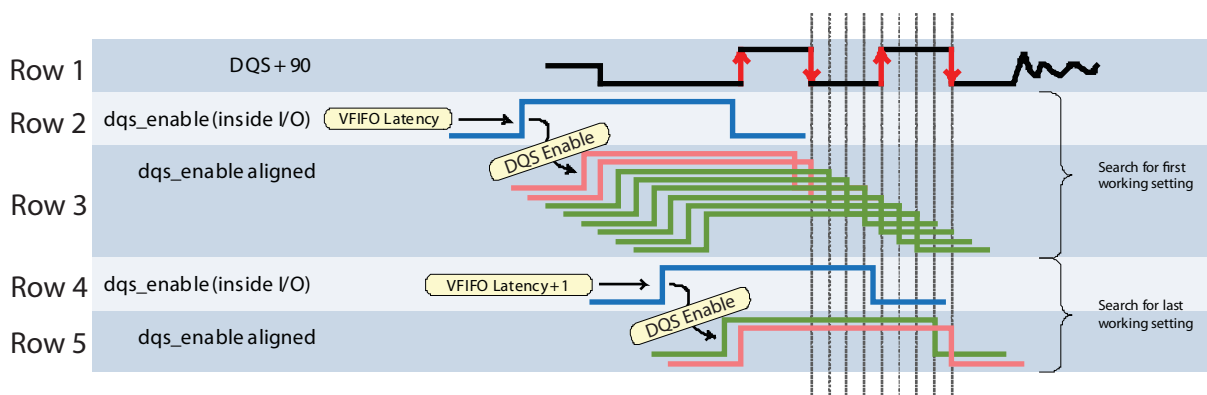
Once the sequencer identifies the two edges of the window, it center-aligns the falling edge of the DQS enable signal within the window. At this point, per-bit deskew has not yet been performed, therefore not all bits are expected to pass the read test; however, for read calibration to succeed, at least one bit per group must pass the read test.

Figure 1-19 shows the DQS and DQS enable signal relationship. The goal of DQS enable calibration is to find settings that satisfy the following conditions:

- The DQS enable signal rises before the first rising edge of DQS.
- The DQS enable signal is at one after the second-last falling edge of DQS.
- The DQS enable signal falls before the last falling edge of DQS.

The ideal position for the falling edge of the DQS enable signal is centered between the second-last and last falling edges of DQS.

Figure 1-19. DQS and DQS Enable Signal Relationships



The following points describe each row of Figure 1-19:

- Row 1 shows the DQS signal shifted by 90° to center-align it to the DQ data.
- Row 2 shows the raw DQS enable signal from the VFIFO.

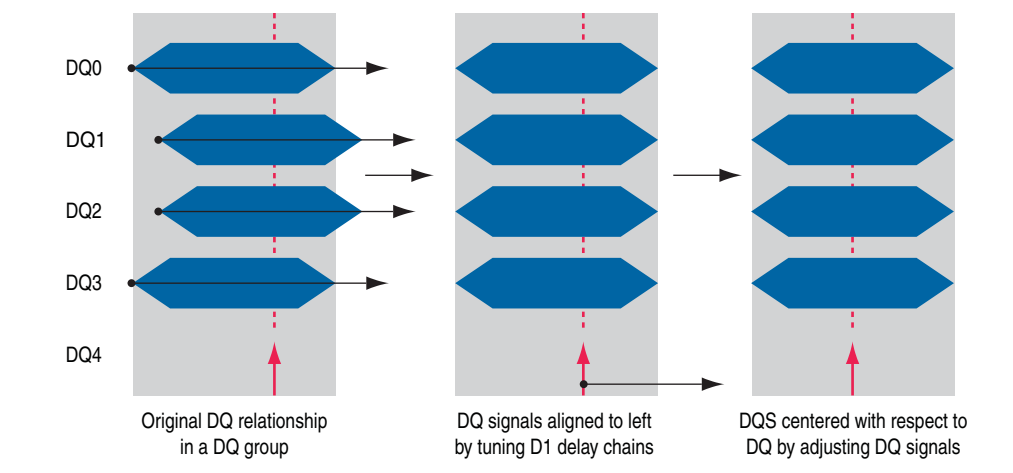
- Row 3 shows the effect of sweeping DQS enable phases. The first two settings (shown in red) fail to properly gate the DQS signal because the enable signal turns off before the second-last falling edge of DQS. The next six settings (shown in green) gate the DQS signal successfully, with the DQS signal covering DQS from the first rising edge to the second-last falling edge.
- Row 4 shows the raw DQS enable signal from the VFIFO, increased by one clock cycle relative to Row 2.
- Row 5 shows the effect of sweeping DQS enable, beginning from the initial DQS enable of Row 4. The first setting (shown in green) successfully gates DQS, with the signal covering DQS from the first rising edge to the second-last falling edge. The second signal (shown in red), does not gate DQS successfully because the enable signal extends past the last falling edge of DQS. Any further adjustment would show the same failure.

Centering DQ/DQS

The centering DQ/DQS stage attempts to align DQ and DQS signals on reads within a group. In reality, each DQ signal within a DQS group might be skewed and consequently arrive at the FPGA at a different time. At this point, the sequencer sweeps each DQ signal in a DQ group to align them, by adjusting DQ input delay chains (D1).


Figure 1-20 illustrates a four DQ/DQS group per-bit-deskew and centering.

Figure 1-20. Per-bit Deskew




To align and center DQ and DQS, the sequencer finds the right edge of DQ signals with respect to DQS by sweeping DQ signals within a DQ group to the right until a failure occurs. In Figure 1-20, DQ0 and DQ3 fail after six taps to the right; DQ1 and DQ2 fail after 5 taps to the right. To align the DQ signals, DQ0 and DQ3 are shifted to the right by 1 tap.

To find the center of DVW, the DQS signal is shifted to the right until a failure occurs. In Figure 1-20, a failure occurs after 3 taps, meaning that there are 5 taps to the right edge and 3 taps to the left edge. To center-align DQ and DQS, the sequencer shifts the aligned DQ signal by 1 more tap to the right.

 The sequencer does not adjust DQS directly; instead, the sequencer center-aligns DQS with respect to DQ by delaying the DQ signals.

Stage 2: Write Calibration Part One

The objectives of the write calibration stage are to align DQS to the memory clock at each memory device, and to compensate for address, command, and memory clock skew at each memory device. This stage is important because the address, command, and clock signals for each memory component arrive at different times.

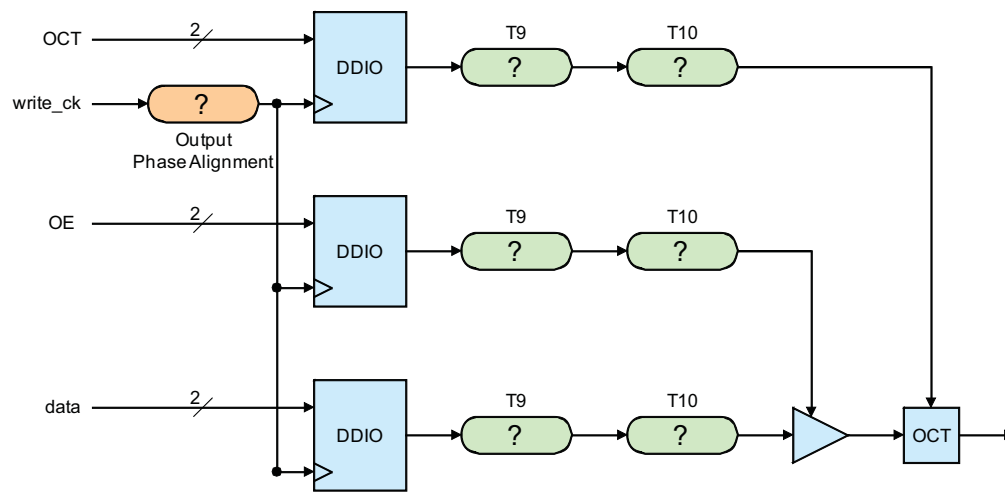
 This stage applies only to DDR2, DDR3, and RLDRAM II protocols; it does not apply to the QDR II and QDR II+ protocols.

Memory clock signals and DQ/DM and DQS signals have specific relationships mandated by the memory device. The PHY must ensure that these relationships are met by skewing DQ/DM and DQS signals. The relationships between DQ/DM and DQS and memory clock signals must meet the tDQSS, tDSS, and tDSH timing constraints.



The sequencer calibrates the write data path using a variety of random burst patterns to compensate for the jitter on the output data path. Simple write patterns are insufficient to ensure a reliable write operation because they might cause imprecise DQS-to-CK alignments, depending on the actual capture circuitry on a memory device. The write patterns in the write leveling stage have a burst length of 8, and are generated by a linear feedback shift register in the form of a pseudo-random binary sequence.

The write data path architecture is the same for DQ, DM, and DQS pins. [Figure 1-21](#) illustrates the write data path for a DQ signal. The phase coming out of the Output Phase Alignment block can be set to different values to center-align DQS with respect to DQ, and it is the same for data, OE, and OCT of a given output.

Figure 1-21. Write Data Path



In write leveling, the sequencer performs write operations with different delay and phase settings, followed by a read. The sequencer can implement any phase shift between 0° and 720° (depending on device and configuration). The sequencer uses the Output Phase Alignment for coarse delays and D5 and D6 for fine delays; D5 has 15 taps of 50 ps each, and D6 has 7 taps of 50 ps each. The DQS signal phase is held at $+90^\circ$ with respect to DQ signal phase (Stratix IV example).

-  Coarse delays are called *phases*, and fine delays are called *delays*; phases are process, voltage, and temperature (PVT) compensated, delays are not (depending on family).
-  Delay and phase values used in this section are examples, for illustrative purposes. Your exact values may vary depending on device and configuration.

The sequencer writes and reads back several burst-length-8 patterns. Because the sequencer has not performed per-bit deskew on the write data path, not all bits are expected to pass the write test. However, for write calibration to succeed, at least one bit per group must pass the write test. The test begins by shifting the DQ/DQS phase until the first write operation completes successfully. The DQ/DQS signals are then delayed to the left by D5 and D6 to find the left edge for that working phase. Then DQ/DQS phase continues the shift to find the last working phase. For the last working phase, DQ/DQS is delayed in 50 ps steps to find the right edge of the last working phase.

The sequencer sweeps through all possible phase and delay settings for each DQ group where the data read back is correct, to define a window within which the PHY can reliably perform write operations. The sequencer picks the closest value to the center of that window as the phase/delay setting for the write data path.

Stage 3: Write Calibration Part Two—DQ/DQS Centering

The process of DQ/DQS centering in write calibration is similar to that performed in read calibration, except that write calibration is performed on the output path, using D5 and D6 delay chains.

Stage 4: Read Calibration Part Two—Read Latency Minimization

At this stage of calibration the sequencer adjusts LFIFO latency to determine the minimum read latency that guarantees correct reads.

Read Latency Tuning

In general, DQ signals from different DQ groups may arrive at the FPGA in a staggered fashion. In a DIMM or multiple memory device system, the DQ/DQS signals from the first memory device arrive sooner, while the DQ/DQS signals from the last memory device arrive the latest at the FPGA.

LFIFO transfers data from the capture registers in IOE to the core and aligns read data to the AFI clock. Up to this point in the calibration process, the read latency has been a maximum value set initially by LFIFO; now, the sequencer progressively lowers the read latency until the data can no longer be transferred reliably. The sequencer then increases the latency by one cycle to return to a working value and adds an additional cycle of margin to assure reliable reads.

Calibration Signals

Table 1-18 lists signals produced by the calibration process.

Table 1-18. Calibration Signals

Signal	Description
afi_cal_fail	Asserts high if calibration fails.
afi_cal_success	Asserts high if calibration is successful.

Document Revision History

Table 1-19 lists the revision history for this document.

Table 1-19. Document Revision History

Date	Version	Changes
November 2011	2.1	<ul style="list-style-type: none"> ■ Consolidated UniPHY information from 11.0 DDR2 and DDR3 SDRAM Controller with UniPHY User Guide, QDR II and QDR II+ SRAM Controller with UniPHY User Guide, and RLDRAM II Controller with UniPHY IP User Guide. ■ Revised Reset and Clock Generation and Dedicated Clock Networks sections. ■ Revised Figure 1-3 and Figure 1-5. ■ Added Tracking Manager to Sequencer section. ■ Revised Interfaces section for DLL, PLL, and OCT sharing interfaces. ■ Revised Using a Custom Controller section. ■ Added UniPHY Calibration Stages section; reordered stages 3 and 4, removed stage 5.

