LogiCORE IP UltraScale Architecture-Based FPGAs Memory Interface Solutions v4.2

Product Guide for Vivado Design Suite

PG150 December 18, 2013

XILINX®

XILINX

Table of Contents

SECTION I: SUMMARY

IP Facts

Chapter 1: Overview

Feature Summary 9
Licensing and Ordering Information 10

SECTION II: DDR3/DDR4

Chapter 2: Product Specification

Standards 12
Performance 12
Resource Utilization 12
Port Descriptions 13

Chapter 3: Core Architecture

Overview 14
Memory Controller 15
PHY 18

Chapter 4: Designing with the Core

Clocking 23
Resets 23
PCB Guidelines for DDR3 23
PCB Guidelines for DDR4 35
Pin and Bank Rules 46
Protocol Description 55

Chapter 5: Customizing and Generating the Core

Vivado Integrated Design Environment 71

Output Generation 79

Chapter 6: Constraining the Core

Required Constraints 80
I/O Standard and Placement 80

Chapter 7: Simulation

Chapter 8: Synthesis and Implementation

Chapter 9: Example Design

Simulating the Example Design (Designs with Standard User Interface) 84

Chapter 10: Test Bench

Stimulus Pattern 87
Bus Utilization 88
Example Patterns 89
Simulating the Performance Traffic Generator 92

SECTION III: RLDRAM 3

Chapter 11: Product Specification

Standards 94
Performance 94
Resource Utilization 94
Port Descriptions 95

Chapter 12: Core Architecture

Overview 96
Memory Controller 99
PHY 101

Chapter 13: Designing with the Core

Clocking 106
Resets 106
PCB Guidelines for RLDRAM 3 107
Pin and Bank Rules 107
Protocol Description 111

Chapter 14: Customizing and Generating the Core

Vivado Integrated Design Environment 119
Output Generation. 119

Chapter 15: Constraining the Core

Chapter 16: Simulation

Chapter 17: Synthesis and Implementation

Chapter 18: Example Design

Chapter 19: Test Bench

SECTION IV: APPENDICES

Appendix A: Debugging

Finding Help on Xilinx.com 126
Debug Tools 128

Appendix B: Additional Resources

Xilinx Resources 129
References 129
Revision History 130
Notice of Disclaimer. 130

SECTION I: SUMMARY

IP Facts

Overview

Introduction

The Xilinx UltraScale Architecture-based FPGAs memory interface solutions core is a combined pre-engineered controller and physical layer (PHY) for interfacing UltraScale Architecture-based FPGA user designs to DDR3 and DDR4 SDRAM and RLDRAM 3 devices. This product guide provides information about using, customizing, and simulating a LogiCORE™ IP DDR3 or DDR4 SDRAM or a RLDRAM 3 interface core for UltraScale Architecture-based FPGAs. The product guide describes the core architecture and provides details on customizing and interfacing to the core.

Features

For feature information on the DDR3/DDR4 SDRAM and RLDRAM 3 interfaces, see the Feature Summary, page 9.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Kintex® UltraScale Family
Supported User Interfaces	User, Native
Resources	See Table 2-1.
Provided with Core	
Design Files	RTL
Example Design	Verilog
Test Bench	Not Provided
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows(2)	
Design Entry	Vivado Design Suite IP Integrator
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete listing of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The Xilinx UltraScale™ architecture DDR3, DDR4, and RLDRAM 3 memory interface cores provide solutions for interfacing with these SDRAM memory types. Both a complete Memory Controller and a physical (PHY) layer only solution are supported. The UltraScale architecture DDR3, DDR4, and RLDRAM 3 memory interface cores are organized in the following high-level blocks.

- **Controller** – The controller accepts burst transactions from the User Interface and generates transactions to and from the SDRAM. The controller takes care of the SDRAM timing parameters and refresh. It coalesces write and read transactions in order to reduce the dead cycles involved in turning the bus around. The controller also reorders commands to improve the utilization of the data bus to the SDRAM.
- **Physical Layer** – The physical layer provides a high speed interface to the SDRAM. This layer includes the hard blocks inside the FPGA and the soft blocks calibration logic necessary to ensure optimal timing of the hard blocks interfacing to the SDRAM.

  The new hard blocks in the UltraScale architecture allow interface rates of up to 2,400 Mb/s to be achieved. The application logic is responsible for all SDRAM transactions, timing, and refresh.

  - These hard blocks include:
    - Data serialization and transmission
    - Data capture and deserialization
    - High-speed clock generation and synchronization
    - Coarse and fine delay elements per pin with voltage and temperature tracking
  - The soft blocks include:
    - **Memory Initialization** – The calibration modules provide a JEDEC®-compliant initialization routine for the particular memory type. The delays in the initialization process may be bypassed to speed up simulation time if desired.

- **Calibration** – The calibration modules provide a complete method to set all delays in the hard blocks and soft IP to work with the memory interface. Each bit is individually trained and then combined in order to ensure optimal interface performance. Results of the calibration process will be available through the Xilinx debug tools. After completion of calibration, the PHY layer presents raw interface to the SDRAM.

- **Application Interface** – The "User Interface" layer provides a simple FIFO-like interface to the application. Data is buffered and read data is presented in request order.

  The above User Interface is layered on top of the Native Interface to the controller. The native interface is accessible by removing the User Interface. The Native Interface has no buffering and presents return data to the application as it is received from the SDRAM which is not necessarily in the original request order. The application must buffer the read and write data as needed and reorder the data if the native interface is used. The native interface does provide the lowest possible latency and the least amount of logic utilization.

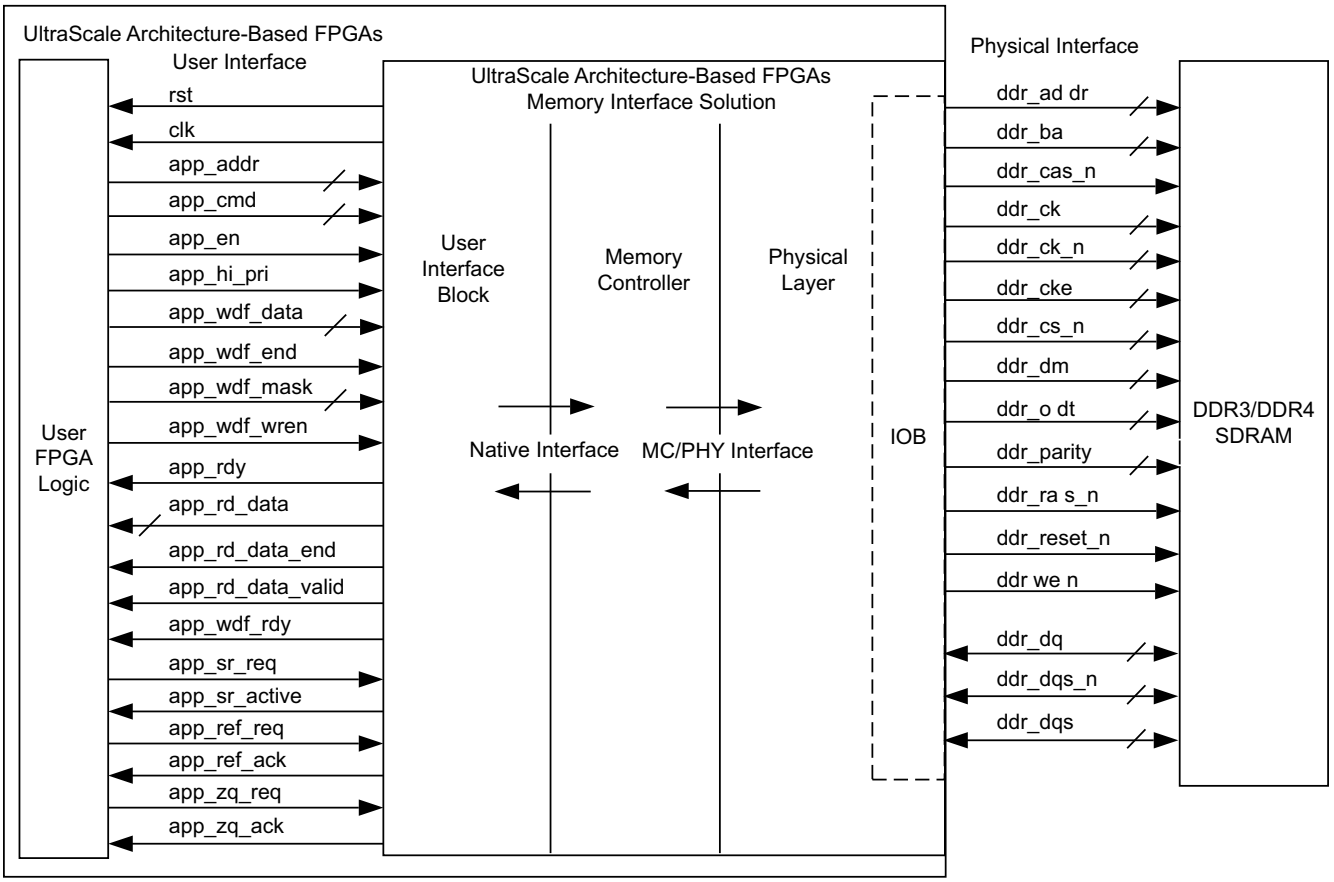

Figure 1-1: **UltraScale Architecture-Based FPGAs Memory Interface Solution**

Feature Summary

DDR3 SDRAM

- Component support for interface width of 16 bits
- DDR3 (1.5V)
- 4 Gb density device support
- 8-bank support
- x8 and x16 device support
- 8:1 DQ:DQS ratio support
- 8-word burst support
- Support for 5 to 14 cycles of column-address strobe (CAS) latency (CL)
- On-die termination (ODT) support
- Support for 5 to 10 cycles of CAS write latency
- Write leveling support for DDR3 (fly-by routing topology required component designs)
- JEDEC®-compliant DDR3 initialization support
- Source code delivery in Verilog
- 4:1 memory to FPGA logic interface clock ratio
- Open, closed, and transaction based pre-charge controller policy

DDR4 SDRAM

- Component support for interface width of 16 bits
- 4 Gb density device support
- x8 and x16 device support
- 8:1 DQ:DQS ratio support
- 8-word burst support
- Support for 9 to 24 cycles of column-address strobe (CAS) latency (CL)
- ODT support
- Support for 9 to 18 cycles of CAS write latency
- Write leveling support for DDR4 (fly-by routing topology required component designs)
- JEDEC-compliant DDR4 initialization support

- Source code delivery in Verilog
- 4:1 memory to FPGA logic interface clock ratio
- Open, closed, and transaction based pre-charge controller policy

RLDRAM 3

- Component support for interface width of 36 bits
- x18 and x36 memory device support
- 8-word burst support
- Support for 5 to 16 cycles of Read Latency
- Address Multiplexing Mode support
- ODT support
- JEDEC-compliant RLDRAM 3 initialization support
- Source code delivery in Verilog
- 4:1 memory to FPGA logic interface clock ratio

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

License Checkers

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license check points for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following tools:

- Vivado design tools: Vivado Synthesis, Vivado Implementation, write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

SECTION II: DDR3/DDR4

Product Specification

Core Architecture

Designing with the Core

Customizing and Generating the Core

Constraining the Core

Simulation

Synthesis and Implementation

Example Design

Test Bench

Product Specification

Standards

This core complies to the JESD79-3F, *DDR3 SDRAM Standard* and JESD79-4, *DDR4 SDRAM Standard,* JEDEC® Solid State Technology Association [Ref 1].

For more information on UltraScale™ architecture documents, see References, page 129.

Performance

Maximum Frequencies

For more information on the maximum frequencies, see *Kintex UltraScale Architecture Data Sheet, DC and AC Switching Characteristics* (DS892) [Ref 2].

Resource Utilization

Kintex UltraScale Devices

Table 2-1 provides approximate resource counts on Kintex® UltraScale™ devices.

Table 2-1: **Device Utilization – Kintex UltraScale FPGAs**

Parameter Values	Device Resources						
Interface Width	Slices	LUTs	FFs	CLB LUTs	CLB Registers	RAMB36E2	PLLE3_ADV
72	413	732	722	10,489	12,534	16	3

Resources required for the UltraScale architecture-based FPGAs MIS core have been estimated for the Kintex UltraScale devices (Table 2-1). These values were generated using Vivado® IP catalog. They are derived from post-synthesis reports, and might change during implementation.

Port Descriptions

There are three port categories at the top-level of the memory interface core called the "user design."

- The first category are the memory interface signals that directly interfaces with the SDRAM. These are defined by the JEDEC specification.
- The second category are the application interface signals which can be either the "native interface" or the simpler "user interface." These are described in the Protocol Description, page 55.
- The third category includes other signals necessary for proper operation of the core. These include the clocks, reset, and status signals from the core. The clocking and reset signals are described in their respective sections.

The active high `init_calib_complete` signal indicates that the initialization and calibration are complete and that the interface is now ready to accept commands for the interface.

Core Architecture

This chapter describes the UltraScale™ architecture-based FPGAs Memory Interface Solutions core with an overview of the modules and interfaces.

Overview

The UltraScale architecture-based FPGAs Memory Interface Solutions is shown in Figure 3-1.

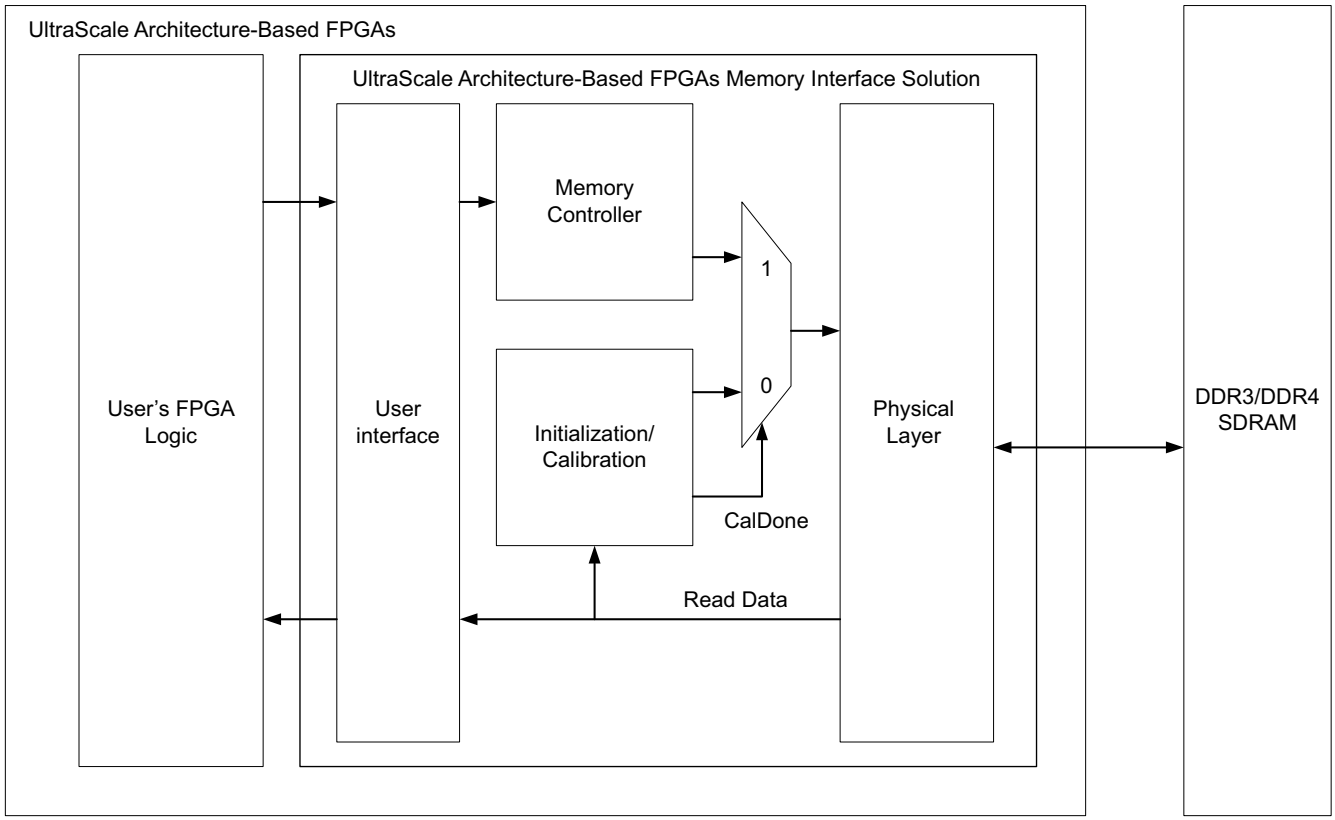

Figure 3-1: **UltraScale Architecture-Based FPGAs Memory Interface Solution Core**

Memory Controller

The Memory Controller (MC) is a general purpose design that is suitable for many applications. The MC balances logic utilization, throughput, latency, and efficiency for typical situations.

The design of the MC is bounded on one side by the Native Interface and on the other side by the PHY. These interfaces result in certain design constraints on the Memory Controller.

Figure 3-2 shows the Memory Controller block diagram.

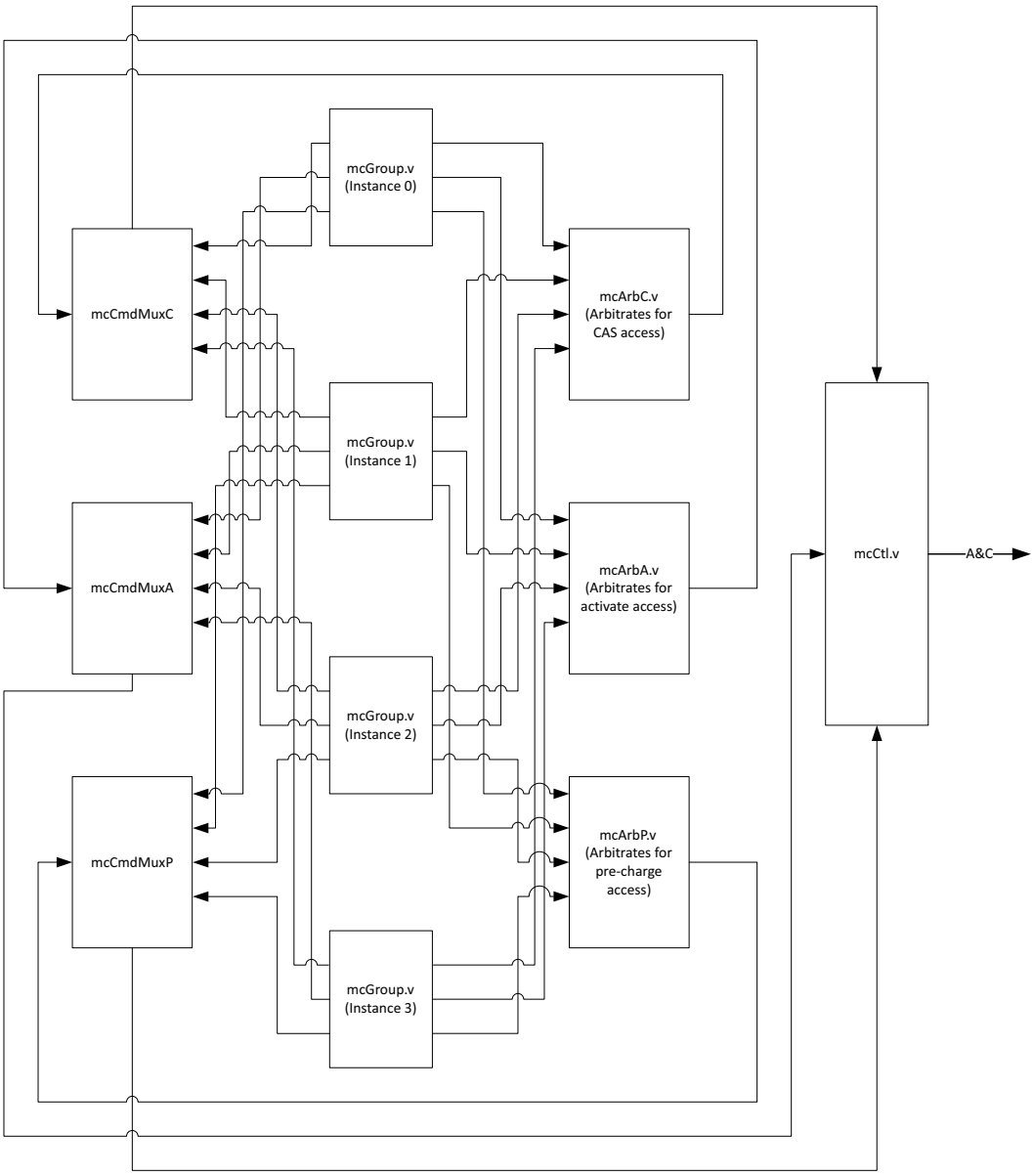

Figure 3-2: **Memory Controller Block Diagram**

Native Interface

The Native Interface does not offer any opportunity of pipelining data, either read or write. On writes data is requested one cycle before it is needed by presenting the data buffer address and the data is expected to be supplied on the next cycle. Hence there is no buffering of any kind for data (except due to the barrel shifting to place the data on a particular DDR clock).

On reads, the data is offered by the MC on the cycle it is available. Read data, along with a buffer address is presented on the Native Interface as soon as it is ready. The data has to be accepted by the Native Interface master.

The number of requests that can be outstanding is dictated by the amount of command buffering provided in the mcGroup module. Although there are no groups in DDR3, the name group notionally represents either a real group in DDR4 x4 and x8 devices (which serves four banks of that group). For DDR3, each mcGroup module would service two banks. In case of DDR4 x16 interface, the mcGroup represents 1-bit of group (there are only one group bit in x16) and 1-bit of bank, whereby the mcGroup serves two banks.

Control and Datapaths

Control Path

The control path starts at mcGroup which can buffer at least two and possibly four commands and dispatch them taking care of address collisions. If each mcGroup can hold "n" commands, the total number of commands that are pending can be up to a maximum of n × 4 or as few as n, if the addresses supplied are all aimed at a single group.

Datapath

The read and write data do not pass through the Memory Controller at all, but are directly connected to the mcCal module. The MC generates the requisite control signals to the mcRead and mcWrite modules telling them the timing of read and write data. The two modules acquire or provide the data as required at the right time.

Read and Write Coalescing

Central to the description is the notion of coalescing reads and writes. As the turnaround times from read to write and write to read are expensive in terms of bus occupancy, the goal of the controller is to put operations of same type together. Two parameters control how many transactions are bunched together.

The parameters are RDCYCLES (default 256) and WRCYCLES (default 128). To prevent starvation, counters (10-bit each) are maintained and are controlled by these parameter specifications to switch between two modes of operation (read mode versus write mode). The MC is either in read mode or write mode at any given instance and entertains only the requests of that type (read or write). It switches over to the other mode if all the pending requests are of "other type" or the counter expires.

The number of read and write cycles can be changed through the RDCYCLES and WRCYCLES parameters. It should be observed that very small parameter values might result in too many switchovers between read and write modes, thus resulting in poor efficiency. Values that are too large result in higher bus efficiency, but at the expense of read latency.

Reordering

The mcGroup module reorders the transactions in a limited manner. The module has a queue for the commands. The queues are reordered based on any address collisions and whether they are reads or writes. To achieve high-speed operation, very complex reordering is not implemented. The address collisions are checked only between banks and not pages.

Reads and writes can pass each other depending on the mode of operation (either read or write). Reads (writes) can bypass reads (writes) depending on the page status. For instance, if a read (write) can bypass another read (write) if the earlier operation is waiting for the page to be opened.

To prevent starvation, a transaction is dispatched even if dispatching another transaction might yield more efficient operation. An aging mechanism is implemented for this purpose.

Group Machines

In the Memory Controller, there are four group state machines. These state machines are allocated depending on technology (DDR3 or DDR4) and width (x4, x8, and x16). The following summarizes the allocation to each group machine. In this description, GM refers to the Group Machine (0 to 3), BG refers to group address, and BA refers to bank address. Note that group in the context of a group state machine denotes a notional group and does not necessarily refer to a real group (except in case of DDR4, part x4 and x8).

- DDR3, any part – Total of eight banks
  - GM 0: BA[2:1] == 2'b00; services banks 0 and 1
  - GM 1: BA[2:1] == 2'b01; services banks 2 and 3
  - GM 2: BA[2:1] == 2'b10; services banks 4 and 5
  - GM 3: BA[2:1] == 2'b11; services banks 6 and 7

- DDR4, x4 and x8 parts – Total of 16 banks
  - GM 0: services BG 0; four banks per group
  - GM 1: services BG 1; four banks per group
  - GM 2: services BG 2; four banks per group
  - GM 3: services BG 3; four banks per group
- DDR4, x16 parts – Total of eight banks
  - GM 0: services BG 0, BA[1] == 0; 2 banks per group
  - GM 1: services BG 0, BA[1] == 1; 2 banks per group
  - GM 0: services BG 1, BA[1] == 0; 2 banks per group
  - GM 1: services BG 1, BA[1] == 1; 2 banks per group

Aging Mechanism (For Reads Only)

Transactions can coalesce to improve performance and one of the considerations for prioritizing is open banks, some transactions can inordinately be delayed. To alleviate this issue, an AGING parameter is provided. If a transaction that has reached the specified wait cycles value, the transaction is pushed ahead of the queue. If this is not possible (due to a conflict), all the transaction ahead is committed for prioritization.

PHY

PHY is considered the low-level physical interface to an external DDR3 or DDR4 SDRAM device as well as all calibration logic for ensuring reliable operation of the physical interface itself. PHY generates the signal timing and sequencing required to interface to the memory device.

PHY contains the following features:

- Clock/address/control-generation logics
- Write and read datapaths
- Logic for initializing the SDRAM after power-up

In addition, PHY contains calibration logic to perform timing training of the read and write datapaths to account for system static and dynamic delays.

Overall PHY Architecture

The UltraScale architecture PHY is composed of dedicated blocks and soft calibration logic. The dedicated blocks are structured adjacent to one another with back-to-back interconnects to minimize the clock and datapath routing necessary to build high performance physical layers.

The Memory Controller and calibration logic communicate with this dedicated PHY in the slow frequency clock domain, which is either divided by 4 or divided by 2. This depends on the DDR3 or DDR4 memory clock. A more detailed block diagram of the PHY design is shown in Figure 3-3.

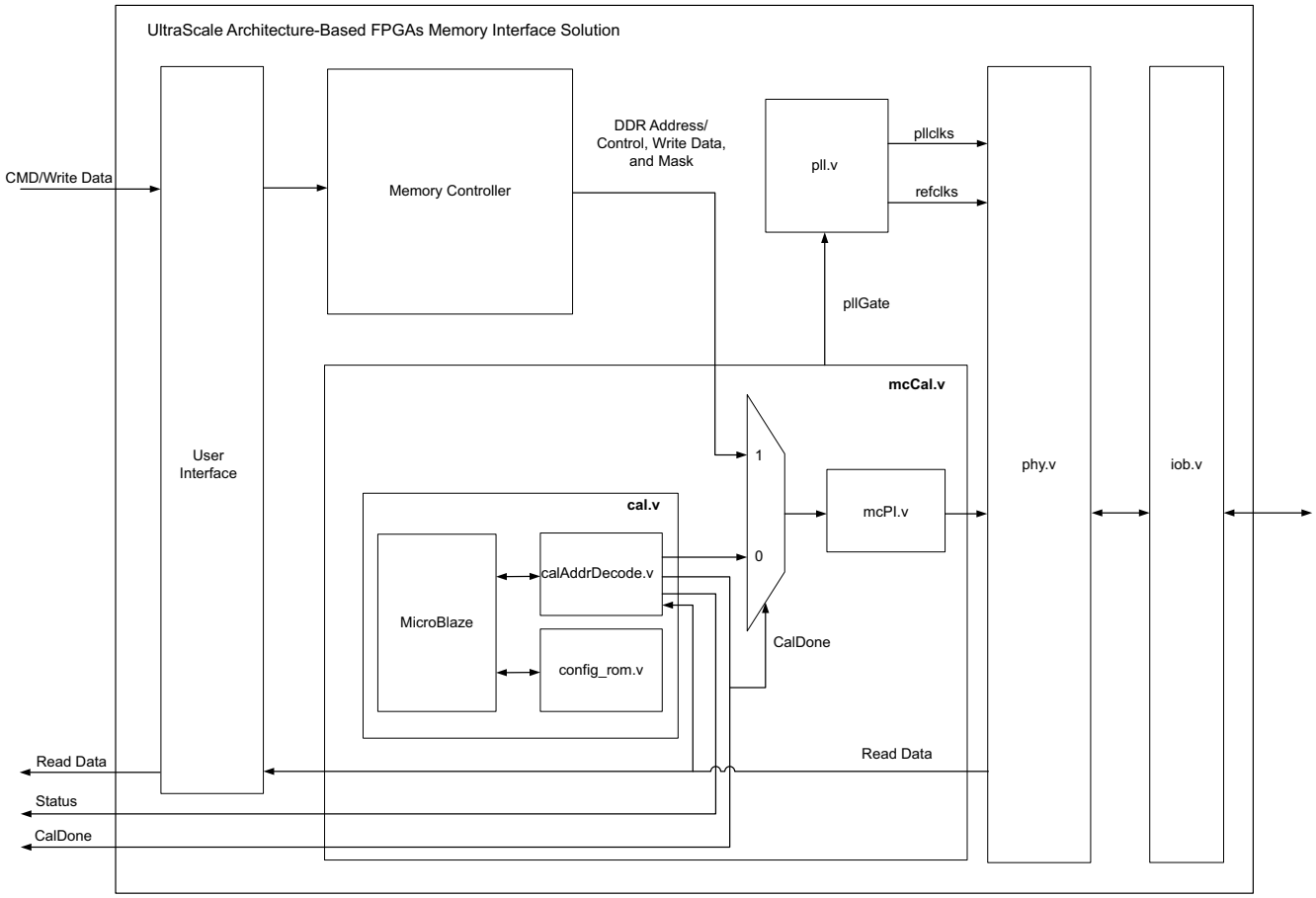

Figure 3-3: **PHY Block Diagram**

The Memory Controller is designed to separate out the command processing from the low-level PHY requirements to ensure a clean separation between the controller and physical layer. The command processing can be replaced with custom logic if desired, while the logic for interacting with the PHY stays the same and can still be used by the calibration logic.

Table 3-1: **PHY Modules**

Module Name	Description
mcCal.v	Contains cal.v, mcPI.v, and MUXes between the calibration and the Memory Controller.
cal.v	Contains the MicroBlaze processing system and associated logic.
mcPI.v	Adjusts signal timing for the PHY for reads and writes.
calAddrDecode.v	FPGA logic interface for the MicroBlaze processor.
Config_rom	Configuration storage for calibration options.
microblaze	MicroBlaze processor
iob.v	Instantiates all byte IOB modules
iobByte.v	Generates the I/O buffers for all the signals in a given byte lane.

The PHY architecture encompasses all of the logic contained in `phy.v`. The PHY contains wrappers around dedicated hard blocks to build up the memory interface from smaller components. A byte lane contains all of the clocks, resets, and datapaths for a given subset of I/O. Multiple byte lanes are grouped together, along with dedicated clocking resources, to make up a single bank memory interface. For more information on the hard silicon physical layer architecture, see the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3].

The memory initialization and calibration are implemented in C programming on a small soft core processor. The MicroBlaze™ Controller System (MCS) is configured with an I/O Module, MicroBlaze Debug Module (MDM), and block RAM. The `calAddrDecode.v` module provides the interface for the processor to the rest of the system and implements helper logic. The `config_rom.v` module stores settings that control the operation of initialization and calibration, providing run time options that can be adjusted without having to recompile the source code.

The address unit connects the MCS to the local register set and the PHY by performing address decode and control translation on the I/O module bus from spaces in the memory map and MUXing return data (`calAddrDecode.v`). In addition, it provides address translation (also known as "mapping") from a logical conceptualization of the DRAM interface to the appropriate pinout-dependent location of the delay control in the PHY address space.

Although the calibration architecture presents a simple and organized address map for manipulating the delay elements for individual data, control and command bits, there is flexibility in how those I/O pins are placed. For a given I/O placement, the path to the FPGA logic is locked to a given pin. To enable a single binary software file to work with any memory interface pinout, a translation block converts the simplified RIU addressing into the pinout-specific RIU address for the target design. The specific address translation is written by MIG after a pinout is selected. The code shows an example of the RTL structure that supports this.

```
Casez(io_address)// MicroBlaze I/O module address
  // … static address decoding skipped
  //=======================================//
  //===========DQ ODELAYS==================//
  //=======================================//
  //Byte0
  28'h0004100: begin //dq2
    riu_addr_cal =  /* MIG Generated */ 6'hd;
    riu_nibble =    /* MIG Generated */ 'h0;
  end
  // … additional dynamic addressing follows
```

In this example, `DQ0` is pinned out on Bit[0] of nibble 0 (nibble 0 according to instantiation order). The RIU address for the ODELAY for Bit[0] is 0x0D (for more details on the RIU address map, see the RIU specification). When `DQ0` is addressed — indicated by address 0x000_4100), this snippet of code is active. It enables nibble 0 (decoded to one-hot downstream) and forwards the address 0x0D to the RIU address bus.

The MicroBlaze I/O module interface updates at a maximum rate of once every three clock cycles, which is not always fast enough for implementing all of the functions required in calibration. A helper circuit implemented in `calAddrDecode.v` is required to obtain commands from the registers and translate at least a portion into single-cycle accuracy for submission to the PHY. In addition, it supports command repetition to enable back-to-back read transactions and read data comparison.

Memory Initialization and Calibration Sequence

After deassertion of the system reset, PHY performs some required internal calibration steps first.

1. Start an I/O offset calibration. This performs an internal adjustment of each single-ended pin to cancel any inherent offsets between single-ended and differential buffers.
2. After I/O offset calibration is completed, the built-in self-check of the PHY (BISC) is run.
3. BISC is used in the PHY to compute internal skews for use in voltage and temperature tracking after calibration is completed.
4. After BISC is completed, calibration logic performs the required power-on initialization sequence for the memory. This is followed by several stages of timing calibration for the write and read datapaths.
5. After calibration is completed, PHY calculates internal offsets to be used in voltage and temperature tracking.
6. PHY indicates calibration is finished and the controller begins issuing commands to the memory.

Figure 3-4 shows the overall flow of memory initialization and the different stages of calibration.

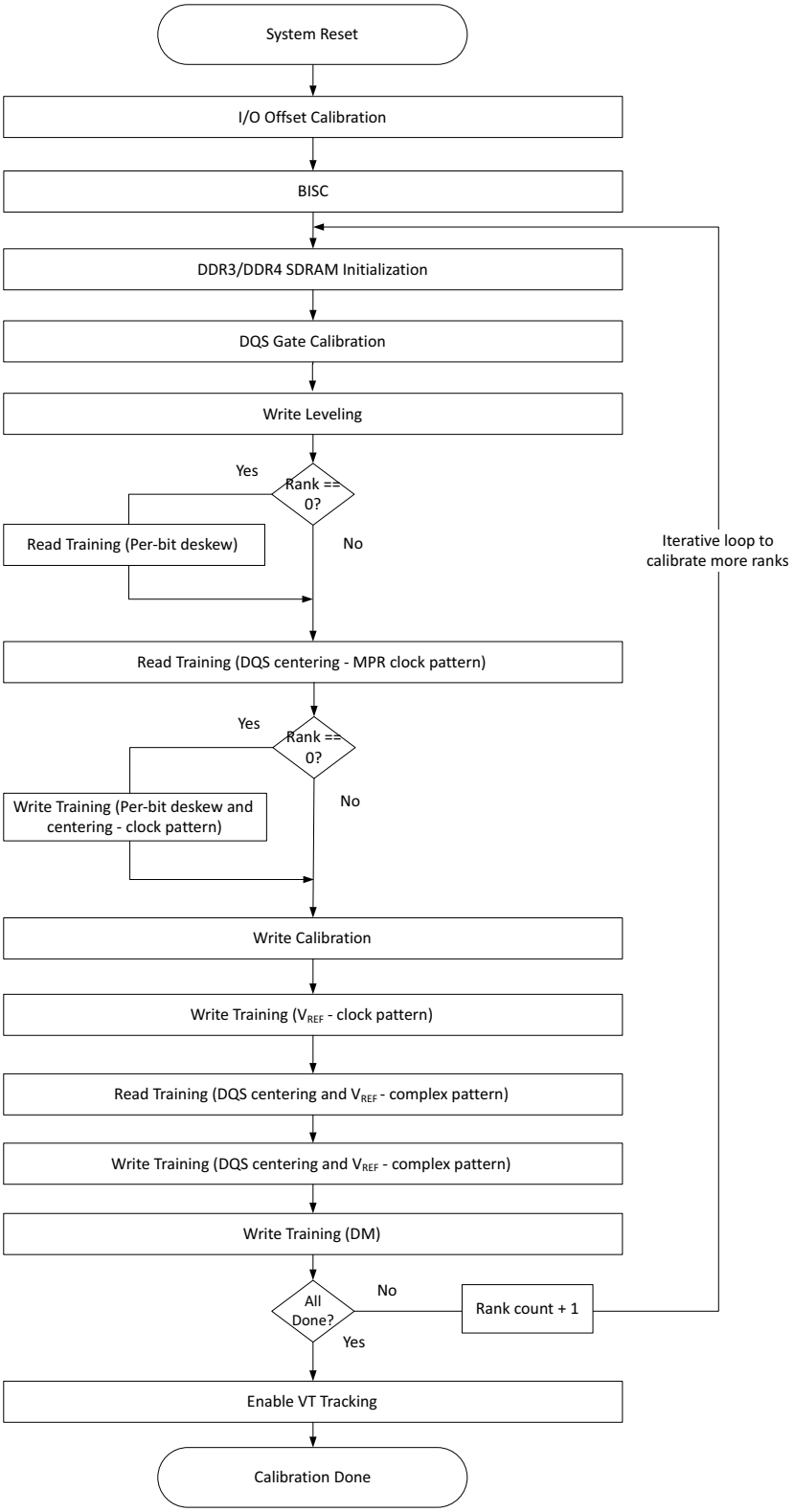

Figure 3-4: **PHY Overall Initialization and Calibration Sequence**

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Clocking

The memory interface requires one PLL in each bank that is occupied by the interface. There are two PLLs per bank. If a bank is shared by two interfaces, both PLLs in that bank are used.

The memory interface requires a high quality reference clock. This clock should come from a differential pair on the same column of the FPGA that the memory interface occupies. The memory interface PLLs generate the appropriate frequencies and phases necessary for proper operation. An output clock is provided for the FPGA logic which includes the controller and the interface logic. This output clock is 1/4 of the memory interface clock in the 4:1 memory clock to FPGA logic clock mode.

Resets

An asynchronous reset input is provided. This active-High reset must assert for a minimum of 20 cycles of the controller clock.

PCB Guidelines for DDR3

Overview

This section provides electrical design guidelines for typical DDR3 interfaces from system-level signal integrity simulations for UltraScale™ architecture. A four component DDR3 32-bit wide interface using x8 devices is used as a typical design example in this document.

As the x16 DRAM component has the same physical width as the x8 DRAM component, the same design rules apply to a four component (x16 DRAM) 64-bit interface. Other DDR3 memory interface design guidelines are included in this document in the future.

IMPORTANT: *All guidelines in this section must be followed in to achieve the maximum data rates specified for the DDR3 interface.*

DDR3 Memory Interface Signal Description

The DDR3 DRAM memory interface consists of clock, control, address, command, and data signals as shown in Table 4-1.

Table 4-1: **DDR3 Memory Interface Signal Description**

Signal Name	Description
Clock Signals	
ck_p/n[1:0]	Differential Clock
Control Signals	
cke[1:0]	Clock Enable
cs_n[1:0]	Chip Select
odt[1:0]	On-Die Termination Enable
Address Signals	
a[15:0]	Memory Address Bus
ba[2:0]	Bank Address
Command Signals	
ras_n	Row Address Select
cas_n	Column Address Select
we_n	Write Enable
Data Signals	
dq[63:0](1)	Data Bus
dqs_p/n[3:0]	Differential Data Strobe

1. The data group includes dq and dm.

Reference Stack-Up

This design guide refers to the stack-up in Table 4-2. All electrical routing constraints are defined upon the reference stack-up. The actual stack-up might be different from this reference stack-up. The related constraints such as spacing should be adjusted accordingly.

IMPORTANT: *To achieve highest memory interface performance, all DDR3 signals must be routed on the top signal layers, that is, L3/L5, as shown in Figure 4-1, to minimize FPGA pin field via crosstalk impact.*
All differential signals, clocks, and strobes must be routed as closely coupled differential pairs from FPGA pins to DRAM pins. Routing signals on bottom layers might degrade the timing margin, as shown in Figure 4-2.

To determine system timing margins in this design following the Xilinx memory simulation guidelines, system designers should use I/O Buffer Information Specification (IBIS) or other simulation tools.

Table 4-2: **Reference Stack-Up**

Layer	Thickness (mil)	Description	
L1	2.5	0.5 oz	Top
	2.9		
L2	0.6	0.5 oz	P/G
	4.5		
L3	0.6	0.5 oz	SIG
	4.5		
L4	0.6	0.5 oz	P/G
	4.5		
L5	0.6	0.5 oz	SIG
	4.5		
L6	1.2	1.0 oz	P/G
	8.0		
L7	1.2	1.0 oz	P/G
	8.0		
L8	1.2	1.0 oz	P/G
	8.0		
L9	1.2	1.0 oz	P/G
	8.0		
L10	1.2	1.0 oz	P/G
	8.0		
L11	1.2	1.0 oz	P/G
	4.5		
L12	0.6	0.5 oz	SIG
	4.5		
L13	0.6	0.5 oz	P/G
	4.5		

Table 4-2: **Reference Stack-Up *(Cont'd)***

Layer	Thickness (mil)	Description	
L14	0.6	0.5 oz	SIG
	4.5		
L15	0.6	0.5 oz	P/G
	2.9		
L16	2.5	0.5 oz	Bottom

Note: Material for this reference stack-up is Isola High-Tg FR-4, 370H.

Figure 4-1: **Layer 3 Routing Example**

Figure 4-2: **Layer 14 Routing Example**

Topology and Routing Guidelines for x4 Component DRAM Configuration

IMPORTANT: *To achieve the specified performance, all rules in the tables must be followed. This includes but not limited to the trace type, width, spacing, impedance, and via count.*

addr/cmd/ctrl Fly-By Termination

With high-speed signaling in DDR3, fly-by topology is used for `addr/cmd/ctrl` signals to achieve the best signal integrity. Each address, command, and control pin on each DRAM is connected to a single trace and terminated at far-end.

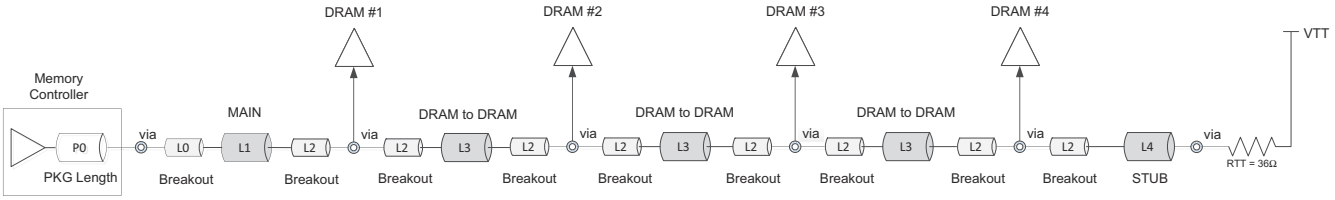

Figure 4-3: **addr/cmd/ctrl Fly-By Termination for x4 DRAMs**

Table 4-3: **Impedance, Length, and Spacing Guidelines for addr/cmd/ctrl Signals**

Parameter	L0 (FPGA Breakout)	L1 (Main PCB)	L2 (DRAM Breakout)	L3	L4 (To RTT)	Units
Trace type	Stripline	Stripline	Stripline	Stripline	Stripline	–
Single-ended impedance Z_0	50±10%	36±10%	50±10%	50±10%	39±10%	Ω
Trace width	4.0	7.0	4.0	4.0	6.0	mil
Trace length	0.0~0.5	1.0~3.0	0.0~0.1	0.35~0.55	0.6~1.0	inch
Spacing in addr/cmd/ctrl (minimum)	4.0	8.0	4.0	8.0	8.0	mil
Spacing to clock signals (minimum)	8.0	20	8.0	20	20	mil
Spacing to other group signals (minimum)	8.0	30	30	30	30	mil
Maximum PCB via count	6.0, see Figure 4-3 for location.					–

Clock Fly-By Termination

Inherent to fly-by topology, the timing skew between the clock and `dqs` signals is de-skewed by the write-leveling feature on DDR3.

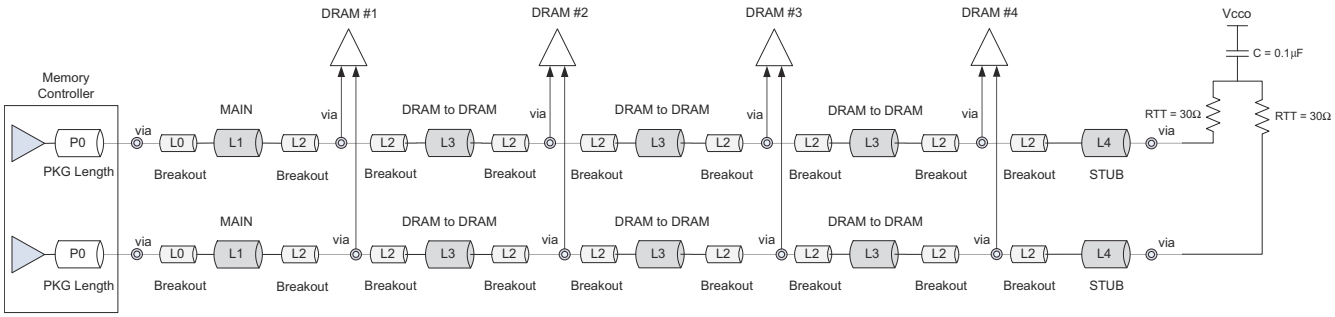

Figure 4-4: **Clock Fly-By Termination for x4 DRAMs**

Table 4-4: **Impedance, Length, and Spacing Guidelines for Clock Signals**

Parameter	L0 (FPGA Breakout)	L1 (Main PCB)	L2 (DRAM Breakout)	L3	L4 (To RTT)	Units
Trace type	Stripline	Stripline	Stripline	Stripline	Stripline	–
Clock differential impedance Z_{diff}	86±10%	76±10%	86±10%	90±10%	76±10%	Ω
Trace width/space/width	4.0/4.0/4.0	6.0/6.0/6.0	4.0/4.0/4.0	4.0/5.0/4.0	6.0/6.0/6.0	mil
Trace length	0.0~0.5	1.0~3.0	0.0~0.1	0.35~0.55	0.6~1.0	inch
Spacing in addr/cmd/ctrl (minimum)	8.0	20	8.0	20	20	mil
Spacing to other group signals (minimum)	8.0	30	30	30	30	mil
Maximum PCB via count per signal	6.0, see Figure 4-3 and Figure 4-4 for location.					–

Data Signals Point-to-Point

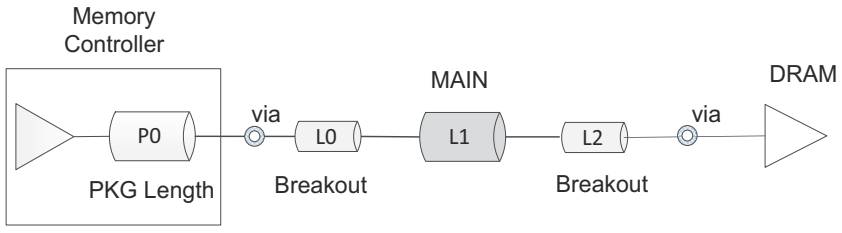

Figure 4-5: **Data Signals Point-to-Point for x4 DRAM Configuration**

Table 4-5: **Impedance, Length, and Spacing Guidelines for Data Signals**

Parameter	L0 (FPGA Breakout)	L1 (Main PCB)	L2 (DRAM Breakout)	Units
Trace type	Stripline	Stripline	Stripline	–
dq single-ended impedance Z_0	50±10%	39±10%	50±10%	Ω
dqs differential impedance Z_{diff}	86±10%	76±10%	86±10%	Ω
Trace width (nominal)	4.0	6.0	4.0	mil
Differential trace width/space/width	4.0/4.0/4.0	6.0/6.0/6.0	4.0/4.0/4.0	mil
Trace length (nominal)	0.0~0.5	1.0~5.0	0.0~0.1	inch
Spacing in byte (minimum)	4.0	8.0	4.0	mil
Spacing byte to byte (minimum)	4.0	20	4.0	mil
dq to strobe spacing (minimum)	4.0	20	8.0	mil
Spacing to other group signals (minimum)	8.0	30	30	mil
Maximum PCB via count	2.0, see Figure 4-5 for location.			–

DDR3 Routing Constraints

There are two parts of constraints requirement for each signal group in the DDR3 memory interface:

- Total length constraints
- Length matching constraints

The total length constraints are shown in Table 4-6.

Table 4-6: **Total Length Constraints**

Signal Group	Reference Figure	Total Length Constraints (inch)
addr/cmd/ctrl P0+L0+L1+L2+…+L5	Figure 4-3	4.0~7.4
Data Signals P0+L0+L1+L2	Figure 4-5	4.2~7.0

The length matching constraints are listed in Table 4-7.

Table 4-7: **Data Group Length Matching Constraints**

Signal Group	Length Matching Constraints (mil)
Data to Strobe	Strobe ±20
Data(1)	±15
dqs_p and dqs_n	±5.0

1. The data group includes dq and dm.

The data group length matching constraints are listed in Table 4-8.

Table 4-8: **addr/cmd/ctrl Length Matching Constraints**

Signal	Signal Segment	Length Matching Constraints
addr/cmd/ctrl, clk	FPGA to DRAM #1	±50 mil
	FPGA to DRAM #2	
	FPGA to DRAM #3	
	FPGA to DRAM #4	
	DRAM #1 to DRAM #2	
	DRAM #2 to DRAM #3	
	DRAM #3 to DRAM #4	
	DRAM #4 to RTT	
ck_p and ck_n	FPGA to DRAM #1	±5.0 mil
	FPGA to DRAM #2	
	FPGA to DRAM #3	
	FPGA to DRAM #4	
	DRAM #1 to DRAM #2	
	DRAM #2 to DRAM #3	
	DRAM #3 to DRAM #4	
	DRAM #4 to RTT	

IMPORTANT: *Package routing length should be included in both total length constraints and length matching constraints.*

Generic Routing Guideline

1. Keep the same byte signals on the same layer.
2. Signal lines are routed over solid reference plane. Avoid routing over void as shown in Figure 4-6.

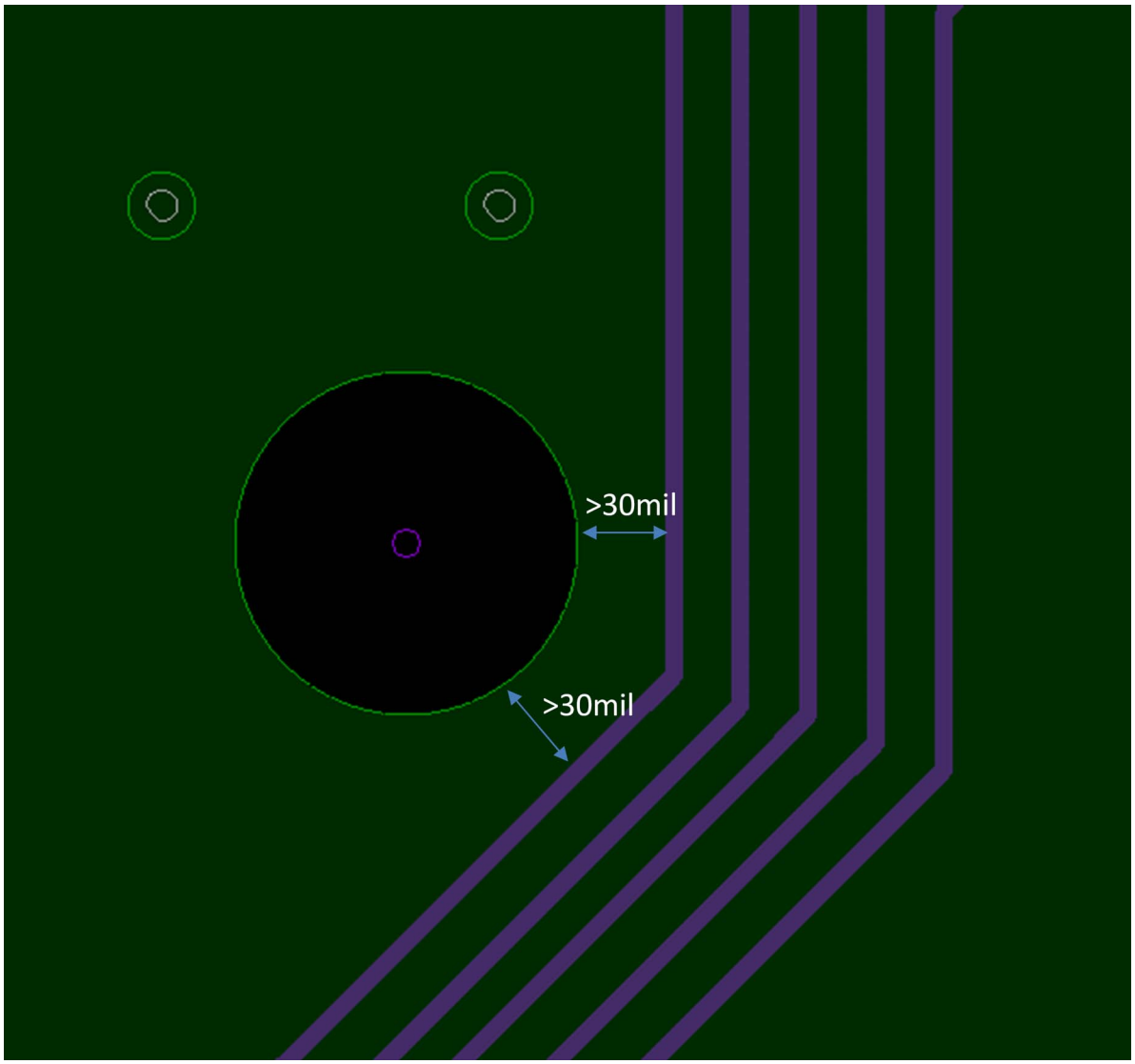

Figure 4-6: **Signal Routing Over Solid Reference Plane**

3. Avoid routing over reference plane splits, see Figure 4-7.

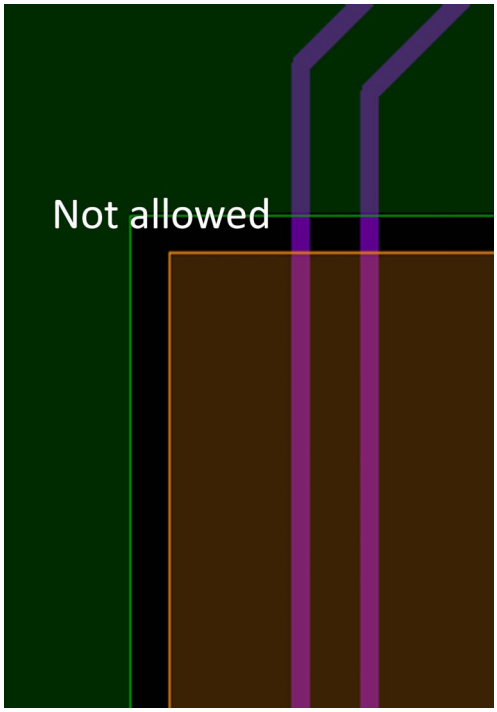

Figure 4-7: **Signal Routing Over Reference Plane Split**

4. Keep the routing at least 30 mils away from the reference plane and void edges with the exception of breakout region, as shown in Figure 4-6.
5. In the breakout region, route signal lines in the middle of the via void aperture. Avoid routing at the edge of via void, as shown in Figure 4-8.

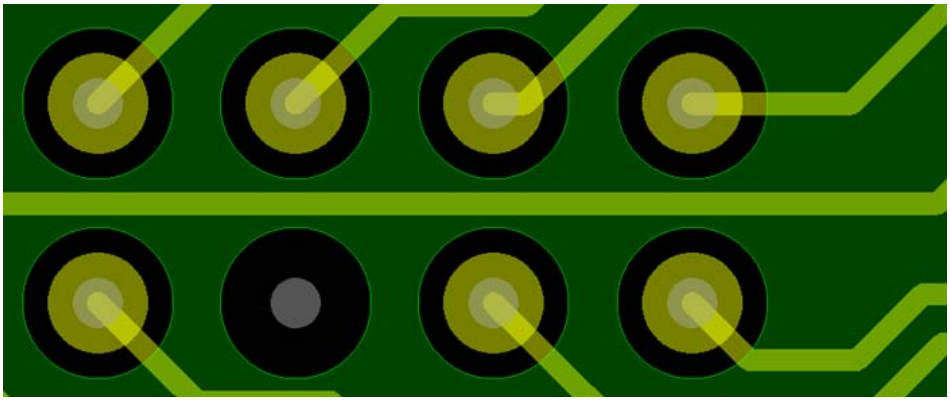

Figure 4-8: **Breakout Region Routing**

6. Add stitch via close to signal via.
7. To optimize the component placement, the recommendations for one component is shown in Figure 4-9. For five components with fly-by topology, it is shown in Figure 4-10.

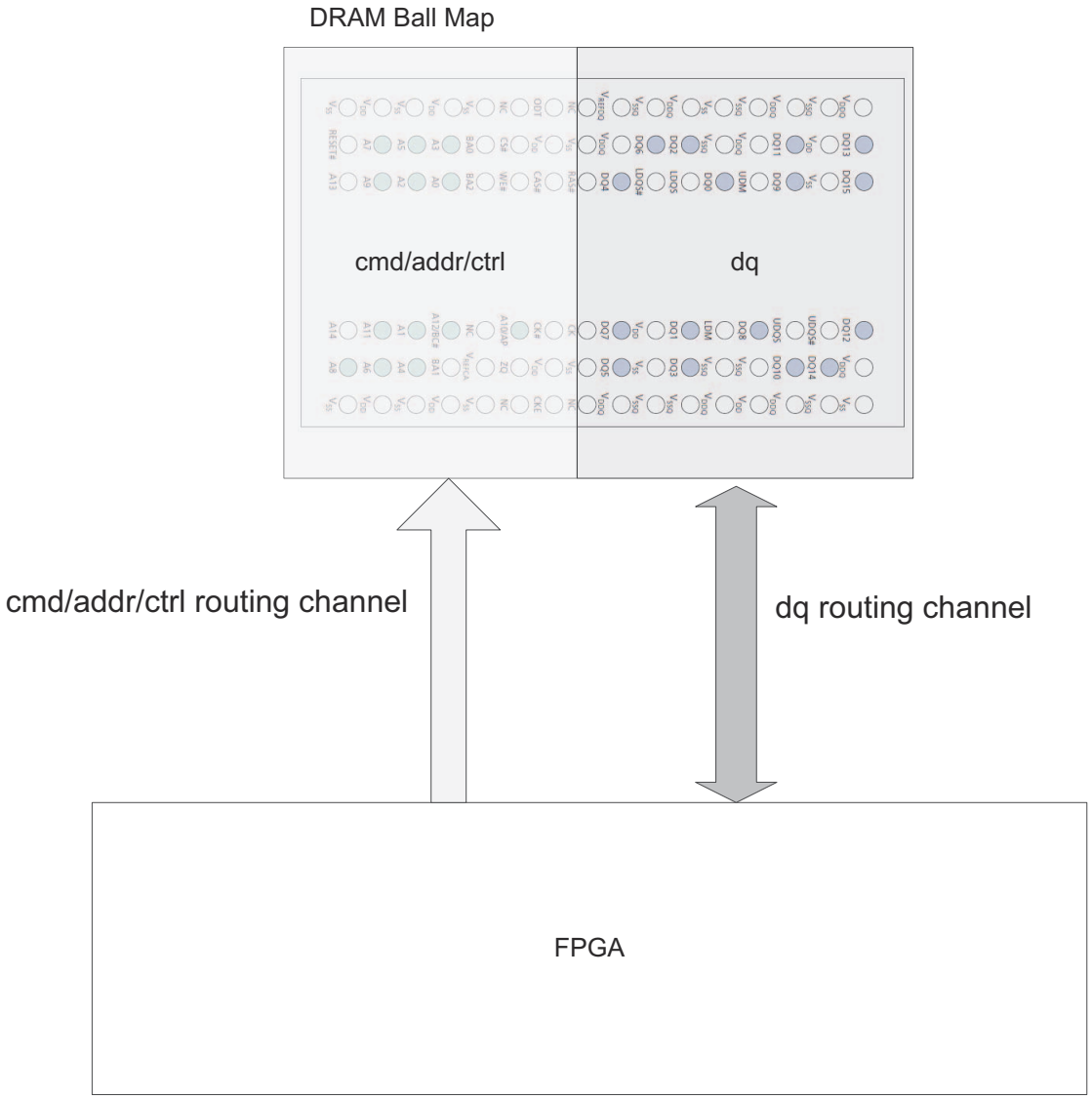

Figure 4-9: **Component Placement Recommendations for One Component**

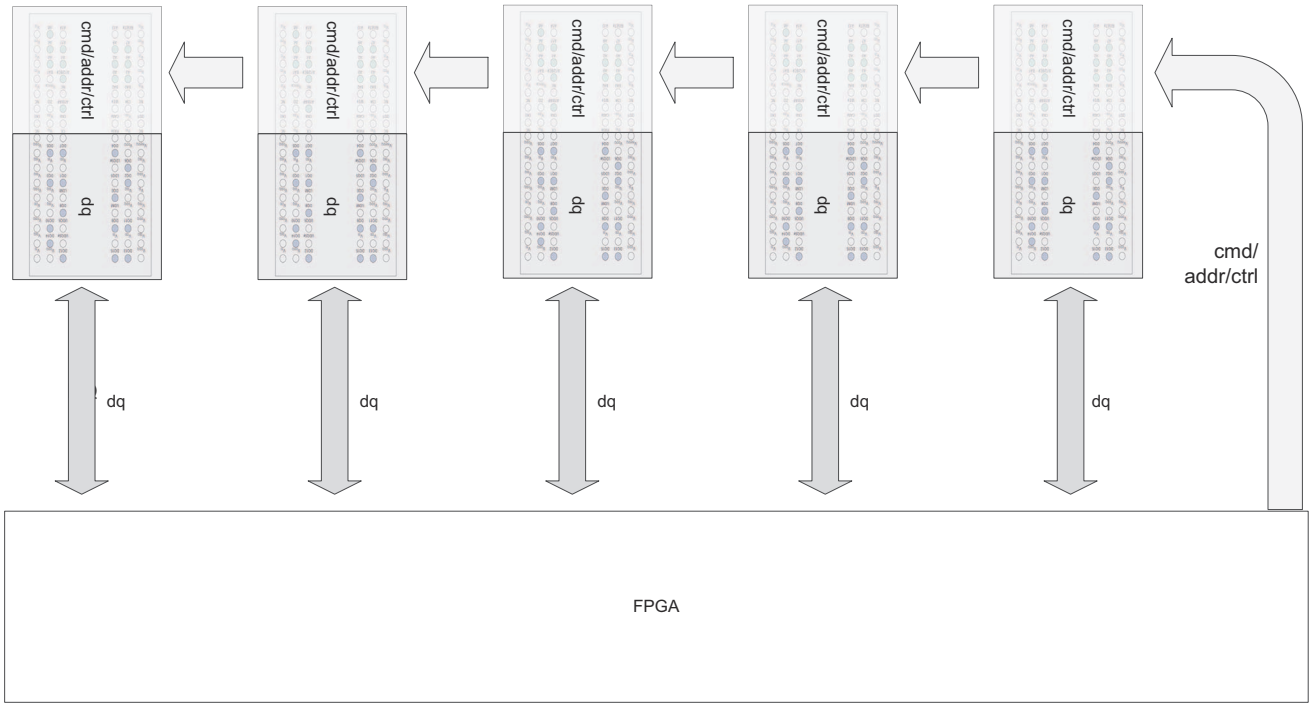

Figure 4-10: **Component Placement Recommendations for Five Components with Fly-By Topology**

ODT Settings

The recommended ODT settings for a four component DDR3 32-bit x8 DRAM or 64-bit x16 DRAM single rank are listed in Table 4-9.

Table 4-9: **ODT Settings**

Item	Setting
ODT_NOM	40Ω
ODT_WR	Disabled

PCB Guidelines for DDR4

Overview

This section provides electrical design guidelines for typical DDR4 interfaces from system-level signal integrity simulations for UltraScale architecture. A four component DDR4 32-bit wide interface using x8 devices is used as a typical design example in this document.

As the x16 DRAM component has the same physical width as the x8 DRAM component, the same design rules apply to a four component (x16 DRAM) 64-bit interface. Other DDR4 memory interface design guidelines are included in this document in the future.

IMPORTANT: *All guidelines in this section must be followed in to achieve the maximum data rates specified for the DDR4 interface.*

DDR4 Memory Interface Signal Description

The DDR4 DRAM memory interface consists of clock, control, address, and data signals as shown in Table 4-10.

Table 4-10: **DDR4 Memory I/O Signal Description**

Signal Name	Description
Clock Signals	
ck_t, ck_c	Differential Clock
Control Signals	
cke	Clock Enable
cs_n	Chip Select
odt	On-Die Termination Enable
Address Signals	
a[17:0]	Address Inputs
bg[1:0]	Bank Group Inputs
ba[1:0]	Bank Address Inputs
act_n	Activation Command Input
par	Command and Address Parity Input
Data Signals	
dq[63:0]	Data Input/Output
dqs_t,_c[3:0]	Data Strobe (Differential)

Table 4-10: **DDR4 Memory I/O Signal Description *(Cont'd)***

Signal Name	Description
dm_n/dbi_n[1]	Data Mask and Data Bus Inversion

1. The data group includes dq and dm_n/dbi_n.

Reference Stack-Up

This design guide refers to the stack-up in Table 4-11. All electrical routing constraints are defined upon the reference stack-up. The actual stack-up might be different from this reference stack-up. The related constraints such as spacing should be adjusted accordingly.

IMPORTANT: *To achieve highest memory interface performance, all DDR4 signals must be routed on the top signal layers, that is, L3/L5, as shown in Figure 4-11, to minimize FPGA pin field via crosstalk impact.*
All differential signals, clocks, and strobes must be routed as closely coupled differential pairs from FPGA pins to DRAM pins. Routing signals on bottom layers might degrade the timing margin, as shown in Figure 4-12.

To determine system timing margins in this design following the Xilinx memory simulation guidelines, system designers should use IBIS or other simulation tools.

Table 4-11: **Reference Stack-Up**

Layer	Thickness (mil)	Description	
L1	2.5	0.5 oz	Top
	2.9		
L2	0.6	0.5 oz	P/G
	4.5		
L3	0.6	0.5 oz	SIG
	4.5		
L4	0.6	0.5 oz	P/G
	4.5		
L5	0.6	0.5 oz	SIG
	4.5		
L6	1.2	1.0 oz	P/G
	8.0		
L7	1.2	1.0 oz	P/G
	8.0		
L8	1.2	1.0 oz	P/G
	8.0		

Table 4-11: **Reference Stack-Up *(Cont'd)***

Layer	Thickness (mil)	Description	
L9	1.2	1.0 oz	P/G
	8.0		
L10	1.2	1.0 oz	P/G
	8.0		
L11	1.2	1.0 oz	P/G
	4.5		
L12	0.6	0.5 oz	SIG
	4.5		
L13	0.6	0.5 oz	P/G
	4.5		
L14	0.6	0.5 oz	SIG
	4.5		
L15	0.6	0.5 oz	P/G
	2.9		
L16	2.5	0.5 oz	Bottom

Note: Material for this reference stack-up is Isola High-Tg FR-4, 370H.

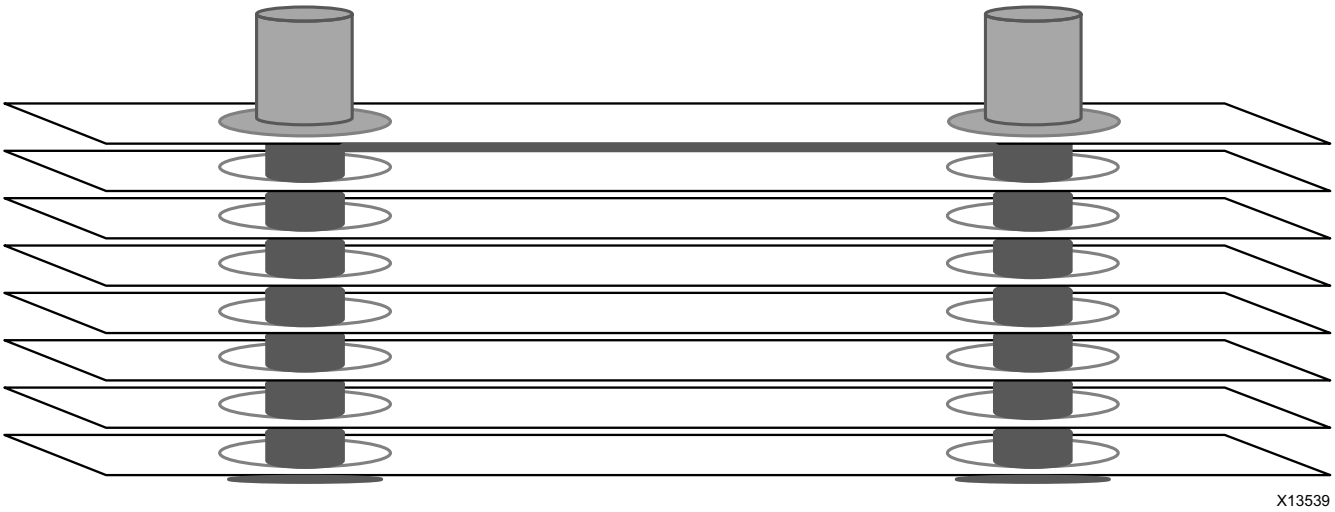

Figure 4-11: **Layer 3 Routing Example**

Figure 4-12: **Layer 14 Routing Example**

Topology and Routing Guidelines for x4 Component DRAM Configuration

IMPORTANT: *To achieve the specified performance, all rules in the tables must be followed. This includes but not limited to the trace type, width, spacing, impedance, and via count.*

addr/cmd/ctrl Fly-By Termination

With high-speed signaling in DDR4, fly-by topology is used for `addr/cmd/ctrl` signals to achieve the best signal integrity. Each clock, address, command, and control pin on each DRAM is connected to a single trace and terminated at far-end. Inherent to fly-by topology, the timing skew between the clock and `dqs` signals is de-skewed by the write-leveling feature on DDR4.

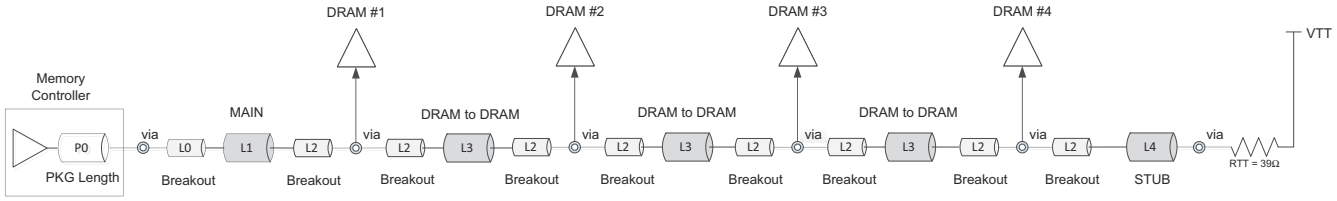

Figure 4-13: **addr/cmd/ctrl Fly-By Termination for x4 DRAMs**

Table 4-12: **Impedance, Length, and Spacing Guidelines for addr/cmd/ctrl Signals**

Parameter	L0 (FPGA Breakout)	L1 (Main PCB)	L2 (DRAM Breakout)	L3	L4 (To RTT)	Units
Trace type	Stripline	Stripline	Stripline	Stripline	Stripline	–
Single-ended impedance Z_0	50±10%	36±10%	50±10%	50±10%	39±10%	Ω
Trace width	4.0	7.0	4.0	4.0	6.0	mil

Table 4-12: **Impedance, Length, and Spacing Guidelines for addr/cmd/ctrl Signals *(Cont'd)***

Parameter	L0 (FPGA Breakout)	L1 (Main PCB)	L2 (DRAM Breakout)	L3	L4 (To RTT)	Units
Trace length	0.0~0.5	1.0~3.0	0.0~0.1	0.35~0.55	0.6~1	inch
Spacing in addr/cmd/ctrl (minimum)	4.0	8.0	4.0	8.0	8.0	mil
Spacing to clock signals (minimum)	8.0	20	8.0	20	20	mil
Spacing to other group signals (minimum)	8.0	30	30	30	30	mil
Maximum PCB via count	6.0, see Figure 4-13 for location.					–

Clock Fly-By Termination

Inherent to fly-by topology, the timing skew between the clock and `dqs` signals is de-skewed by the write-leveling feature on DDR4.

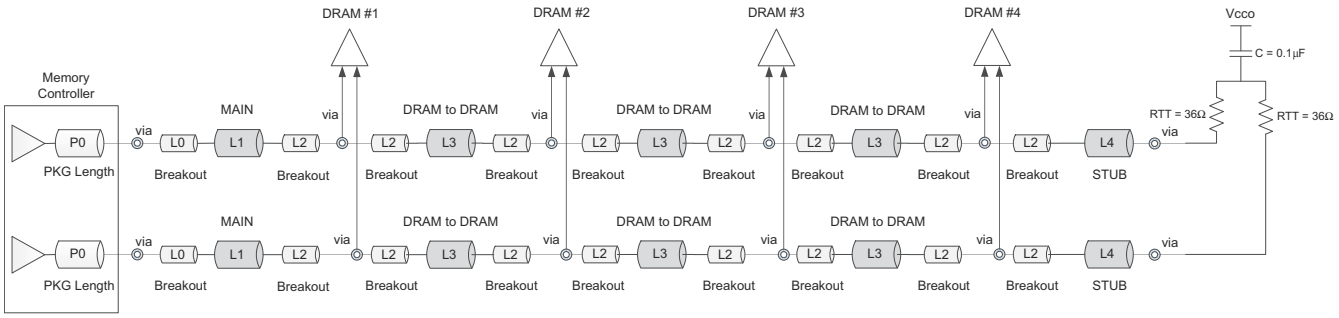

Figure 4-14: **Clock Fly-By Termination for x4 DRAMs**

Table 4-13: **Impedance, Length, and Spacing Guidelines for Clock Signals**

Parameter	L0 (FPGA Breakout)	L1 (Main PCB)	L2 (DRAM Breakout)	L3	L4 (To RTT)	Units
Trace type	Stripline	Stripline	Stripline	Stripline	Stripline	–
Clock differential impedance Z_{diff}	86	76	86	90	76	Ω
Trace width/space/width	4.0/4.0/4.0	6.0/6.0/6.0	4.0/4.0/4.0	4.0/5.0/4.0	6.0/6.0/6.0	mil
Trace length	0.0~0.5	L1 (Figure 4-13) +0.09	0.0~0.1	0.35~0.55	0.6~1.0	inch
Spacing in addr/cmd/ctrl (minimum)	8.0	20	8.0	20	20	mil
Spacing to other group signals (minimum)	8.0	30	30	30	30	mil
Maximum PCB via count per signal	6.0, see Figure 4-14 for location.					–

Data Signals Point-to-Point

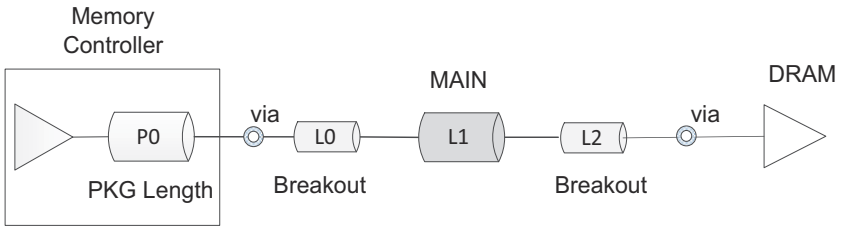

Figure 4-15: **Data Signals Point-to-Point for x4 DRAM Configuration**

Table 4-14: **Impedance, Length, and Spacing Guidelines for Data Signals**

Parameter	L0 (FPGA Breakout)	L1 (Main PCB)	L2 (DRAM Breakout)	Units
Trace type	Stripline	Stripline	Stripline	–
dq single-ended impedance Z_0	50±10%	39±10%	50±10%	Ω
dqs differential impedance Z_{diff}	86	76	86	Ω
Trace width (nominal)	4.0	6.0	4.0	mil
Differential trace width/space/width	4.0/4.0/4.0	6.0/6.0/6.0	4.0/4.0/4.0	mil
Trace length	0.0~0.5	1.0~4.0	0.0~0.1	inch
Spacing in byte (minimum)	4.0	8.0	4.0	mil
Spacing byte to byte (minimum)	4.0	20	4.0	mil
dq to strobe spacing (minimum)	4.0	20	8.0	mil
Spacing to other group signals (minimum)	8.0	30	30	mil
Maximum PCB via count	2.0, see Figure 4-15 for location.			–

DDR4 Routing Constraints

There are two parts of constraints requirement for each signal group in the DDR4 memory interface:

- Total length constraints
- Length matching constraints

The total length constraints are shown in Table 4-15.

Table 4-15: **Total Length Constraints**

Signal Group	Reference Figure	Total Length Constraints (inch)
addr/cmd/ctrl P0+L0+L1+L2+L4	Figure 4-13	4.0~7.4
Clock to addr/cmd/ctrl	Figure 4-14	+0.09
Data Signals P0+L0+L1+L2	Figure 4-15	4.2~6.0

The length matching constraints are listed in Table 4-16.

Table 4-16: **Data Group Length Matching Constraints**

Signal Group	Length Matching Constraints (mil)
Data to Strobe	Strobe ±20
Data[1]	±15
dqs_p and dqs_n	±5.0

1. The data group includes dq and dm_n/dbi_n.

The data group length matching constraints are listed in Table 4-17.

Table 4-17: **addr/cmd/ctrl Length Matching Constraints**

Signal	Signal Segment	Length Matching Constraints
addr/cmd/ctrl	FPGA to DRAM #1	±50 mil
	FPGA to DRAM #2	
	FPGA to DRAM #3	
	FPGA to DRAM #4	
	DRAM #1 to DRAM #2	
	DRAM #2 to DRAM #3	
	DRAM #3 to DRAM #4	
	DRAM #4 to RTT	
ck_t and ck_c	FPGA to DRAM #1	±5.0 mil
	FPGA to DRAM #2	
	FPGA to DRAM #3	
	FPGA to DRAM #4	
	DRAM #1 to DRAM #2	
	DRAM #2 to DRAM #3	
	DRAM #3 to DRAM #4	
	DRAM #4 to RTT	

IMPORTANT: *Package routing length should be included in both total length constraints and length matching constraints.*

Generic Routing Guideline

1. Keep the same byte signals on the same layer.
2. Signal lines are routed over solid reference plane. Avoid routing over void as shown in Figure 4-16.

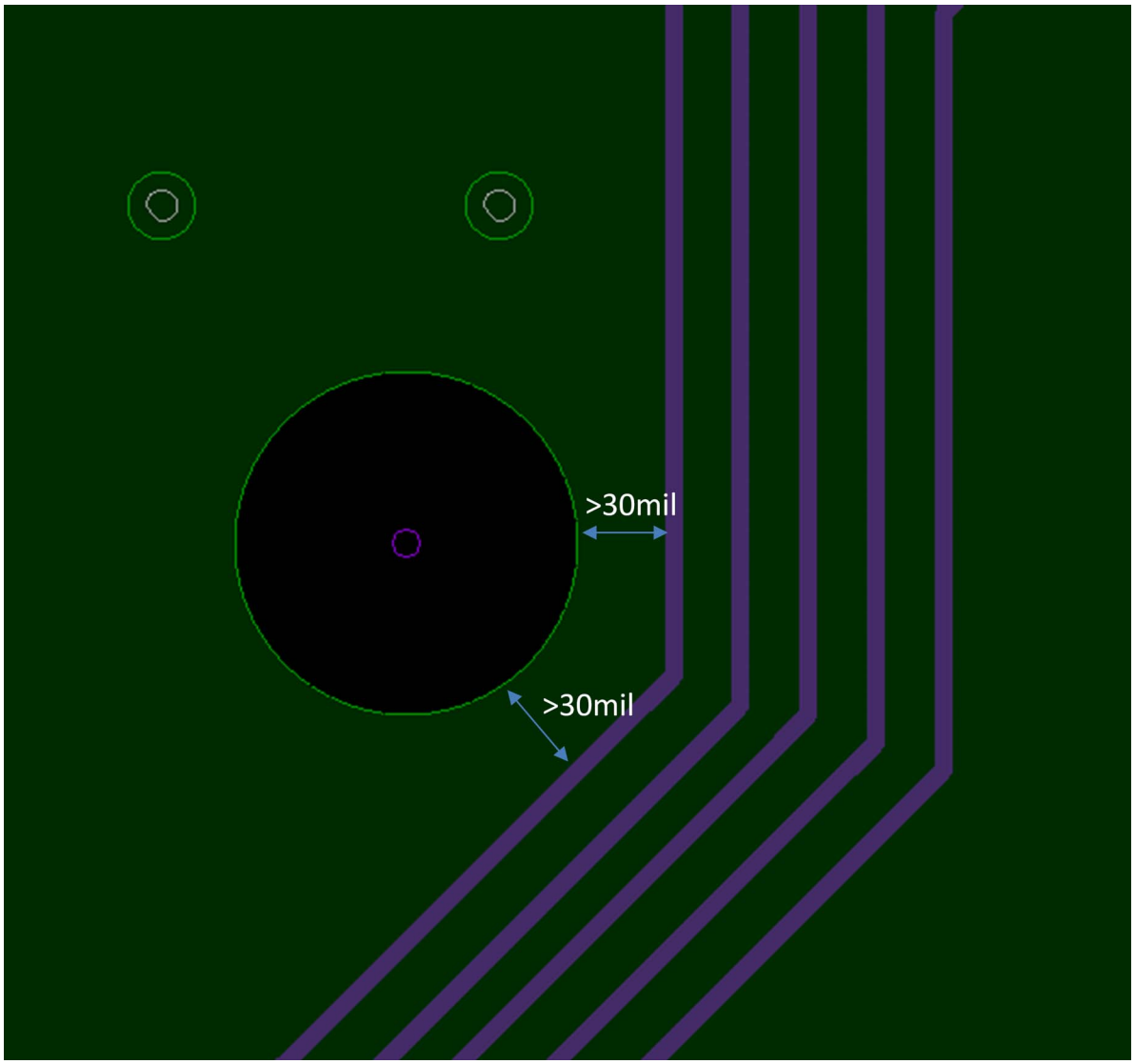

Figure 4-16: **Signal Routing Over Solid Reference Plane**

3. Avoid routing over reference plane splits, see Figure 4-17.

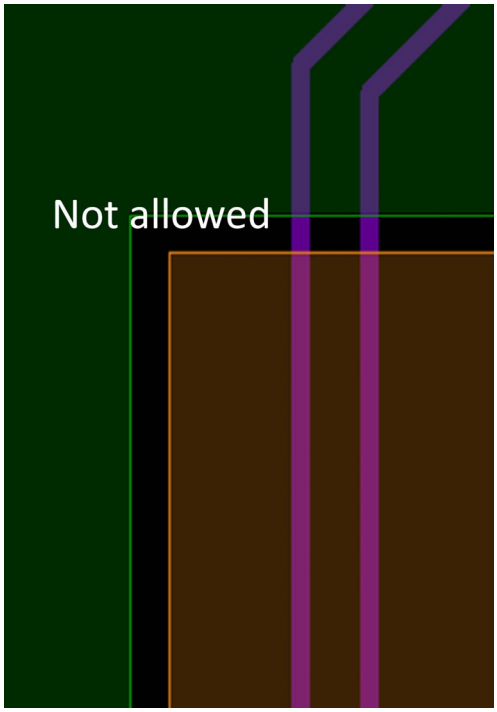

Figure 4-17: **Signal Routing Over Reference Plane Split**

4. Keep the routing at least 30 mils away from the reference plane and void edges with the exception of breakout region, as shown in Figure 4-16.
5. In the breakout region, route signal lines in the middle of the via void aperture. Avoid routing at the edge of via void, as shown in Figure 4-18.

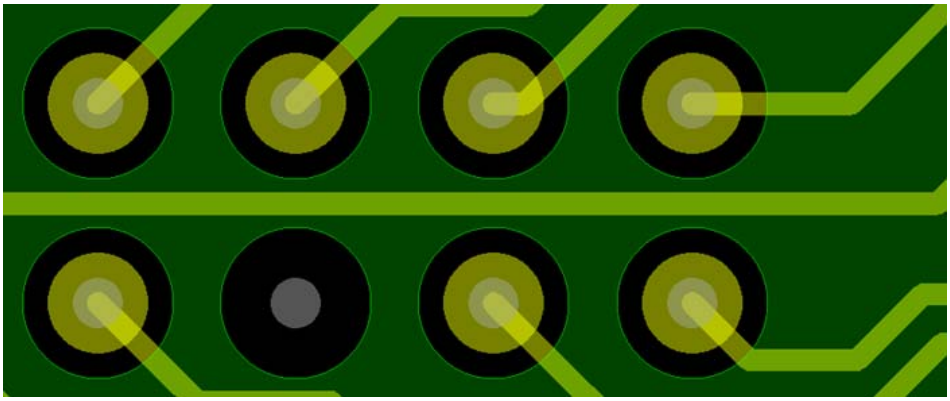

Figure 4-18: **Breakout Region Routing**

6. Add stitch via close to signal via.
7. To optimize the component placement, the recommendations for one component is shown in Figure 4-19. For five components with fly-by topology, it is shown in Figure 4-20.

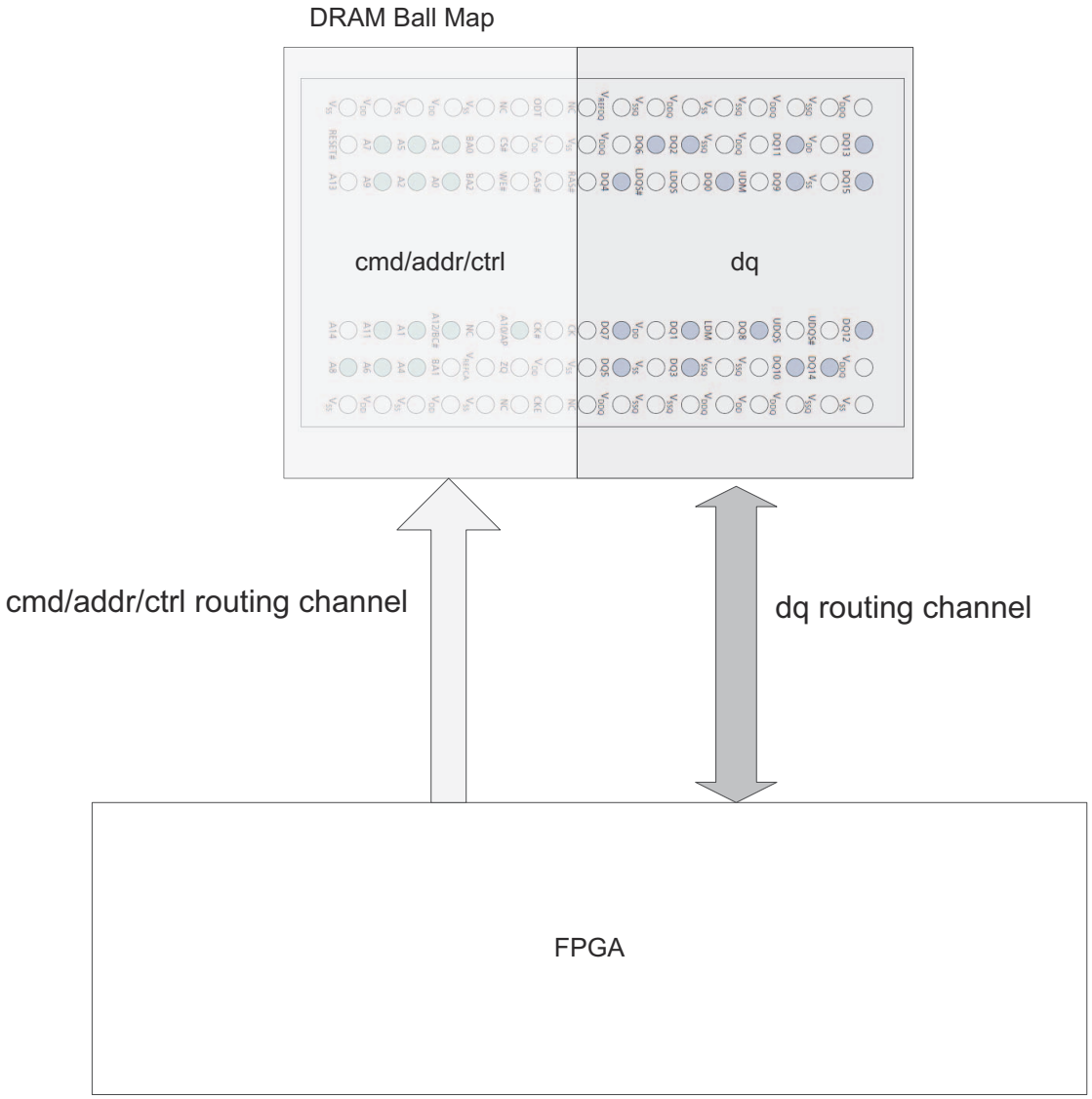

Figure 4-19: **Component Placement Recommendations for One Component**

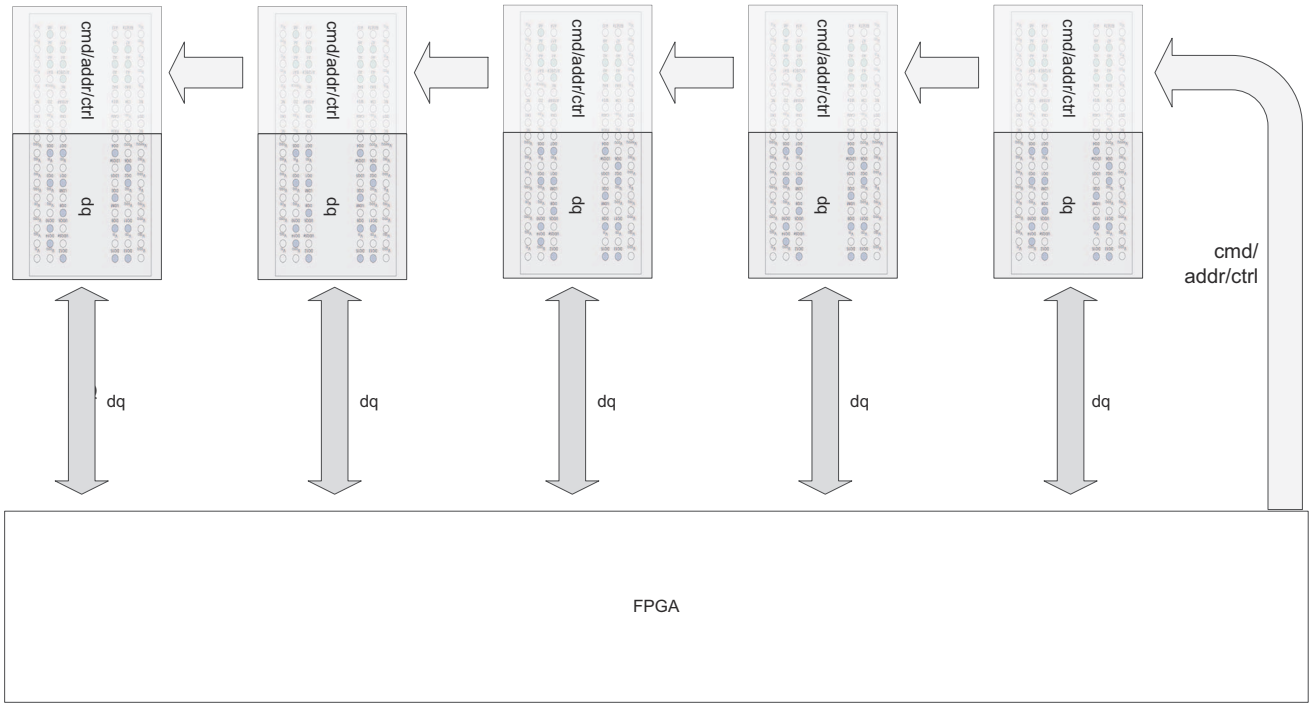

Figure 4-20: **Component Placement Recommendations for Five Components with Fly-By Topology**

ODT Settings

The recommended ODT settings for a four component DDR4 32-bit x8 DRAM or 64-bit x16 DRAM single rank are listed in Table 4-18.

Table 4-18: **ODT Settings**

Item	Setting
ODT_NOM	40Ω
ODT_PARK	Disabled
ODT_WR	Disabled

Pin and Bank Rules

DDR3 Pin Rules

The rules are for single rank memory interfaces. For multi-rank information, contact Xilinx.

- Address/control means `cs_n`, `ras_n`, `cas_n`, `we_n`, `ba`, `ck`, `cke`, `a`, and `odt`.
- Pins in a byte lane are numbered N0 to N12.
- Byte lanes in a bank are designed by T0, T1, T2, or T3. Nibbles within a byte lane are distinguished by a "U" or "L" designator added to the byte lane designator (T0, T1, T2, or T3). Thus they are T0L, T0U, T1L, T1U, T2L, T2U, T3L, and T3U.

Note: There are two PLLs per bank and a controller uses one PLL in every bank that is being used by the interface.

1. `dqs`, `dq`, and `dm` location.
   a. Designs using x8 or x16 components – `dqs` must be located on a dedicated byte clock pair in the upper nibble designated with "U." `dq` associated with a `dqs` must be in same byte lane on any of the other pins except pins 1 and 12.
   b. Designs using x4 components – `dqs` must be located on a dedicated byte clock pair in the nibble. `dq` associated with a `dqs` must be in same nibble on any of the other pins except pins N1 (lower nibble) and pin N12 (upper nibble).
   c. `dm` (if used) must be located on pin N0 in the byte lane with the corresponding `dqs`.
2. Byte lanes are configured as either data or address/control.
   a. Pins N1 and N12 can be used for address/control in a data byte lane.
   b. No data signals (`dqs`, `dq`, `dm`) can be placed in an address/control byte lane.
3. Address/control can be on any of the 13 pins in the address/control byte lanes. Address/control must be contained within the same bank. Address/control must be in the centermost bank.
4. There is one `vr` pin per bank and DCI is required. DCI cascade is not permitted. All rules for the DCI in the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3] must be followed.
5. `ck` must be on the PN pair in the center of the byte lane which is designated as the Upper byte clock pair.
6. `reset_n` can be on any pin as long as FPGA logic timing is met and I/O standard can be accommodated for the chosen bank (LVCMOS15 or LVCMOS135).
7. Banks can be shared between two controllers.
   a. Each byte lane is dedicated to a specific controller (except for `reset_n`).

b. Byte lanes from one controller cannot be placed inside the other. For example, with controllers A and B, "AABB" is allowed, while "ABAB" is not.

8. All I/O banks used by the memory interface must be in the same column.
9. Maximum height of interface is five contiguous banks for 144-bit wide interface.
10. Bank skipping is not allowed.
11. The input clock for the master PLL in the interface must come from the a clock capable pair in the I/O column used for the memory interface.
12. There are dedicated V_{REF} pins (not included in the rules above). If an external V_{REF} is not used, the V_{REF} pins should be pulled to ground by a 500Ω resistor. For more information, see the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3]. These pins must be connected appropriately for the standard in use.
13. The interface must be contained within the same I/O bank type (High Range or High Performance). Mixing bank types is not permitted with the exceptions of the `reset_n` in step 6 above and the input clock mentioned in step 11 above.

DDR3 Pinout Examples

Table 4-19 shows an example of a 16-bit DDR3 interface contained within one bank. This example is for a component interface using 2 × 8 DDR3 components.

Table 4-19: **16-Bit DDR3 Interface Contained in One Bank**

Bank	Signal Name	Byte Group	I/O Type	Special Designation
1	a0	T3U_12	–	–
1	a1	T3U_11	N	–
1	a2	T3U_10	P	–
1	a3	T3U_9	N	–
1	a4	T3U_8	P	–
1	a5	T3U_7	N	DBC-N
1	a6	T3U_6	P	DBC-P
1	a7	T3L_5	N	–
1	a8	T3L_4	P	–
1	a9	T3L_3	N	–
1	a10	T3L_2	P	–
1	a11	T3L_1	N	DBC-N
1	a12	T3L_0	P	DBC-P
1	a13	T2U_12	–	–

Table 4-19: **16-Bit DDR3 Interface Contained in One Bank *(Cont'd)***

Bank	Signal Name	Byte Group	I/O Type	Special Designation
1	a14	T2U_11	N	–
1	we	T2U_10	P	–
1	cas_n	T2U_9	N	–
1	ras_n	T2U_8	P	–
1	ck_n	T2U_7	N	QBC-N
1	ck_p	T2U_6	P	QBC-P
1	cs_n	T2L_5	N	–
1	ba0	T2L_4	P	–
1	ba1	T2L_3	N	–
1	ba2	T2L_2	P	–
1	pll refclk_n	T2L_1	N	QBC-N
1	pll refclk_p	T2L_0	P	QBC-P
1	cke	T1U_12	–	–
1	dq15	T1U_11	N	–
1	dq14	T1U_10	P	–
1	dq13	T1U_9	N	–
1	dq12	T1U_8	P	–
1	dqs1_n	T1U_7	N	QBC-N
1	dqs1_p	T1U_6	P	QBC-P
1	dq11	T1L_5	N	–
1	dq10	T1L_4	P	–
1	dq9	T1L_3	N	–
1	dq8	T1L_2	P	–
1	odt	T1L_1	N	QBC-N
1	dm1	T1L_0	P	QBC-P
1	vr	T0U_12	–	–
1	dq7	T0U_11	N	–
1	dq6	T0U_10	P	–
1	dq5	T0U_9	N	–
1	dq4	T0U_8	P	–
1	dqs0_n	T0U_7	N	DBC-N

Table 4-19: **16-Bit DDR3 Interface Contained in One Bank *(Cont'd)***

Bank	Signal Name	Byte Group	I/O Type	Special Designation
1	dqs0_p	T0U_6	P	DBC-P
1	dq3	T0L_5	N	–
1	dq2	T0L_4	P	–
1	dq1	T0L_3	N	–
1	dq0	T0L_2	P	–
1	reset_n	T0L_1	N	DBC-N
1	dm0	T0L_0	P	DBC-P

DDR4 Pin Rules

The rules are for single rank memory interfaces. For multi-rank information, contact Xilinx.

- Address/control means `cs_n`, `ras_n`, `cas_n`, `we_n`, `ba`, `bg`, `ck`, `cke`, `a`, `odt`, `act_n`, and `par`.
- Pins in a byte lane are numbered N0 to N12.
- Byte lanes in a bank are designed by T0, T1, T2, or T3. Nibbles within a byte lane are distinguished by a "U" or "L" designator added to the byte lane designator (T0, T1, T2, or T3). Thus they are T0L, T0U, T1L, T1U, T2L, T2U, T3L, and T3U.

Note: There are two PLLs per bank and a controller uses one PLL in every bank that is being used by the interface.

1. `dqs`, `dq`, and `dm/dbi` location.
   a. Designs using x8 or x16 components – `dqs` must be located on a dedicated byte clock pair in the upper nibble designated with "U." `dq` associated with a `dqs` must be in same byte lane on any of the other pins except pins N1 and N12.
   b. Designs using x4 components – `dqs` must be located on a dedicated byte clock pair in the nibble. `dq` associated with a `dqs` must be in same nibble on any of the other pins except pins N1 (lower nibble) and pin N12 (upper nibble).
   c. `dm/dbi` must be on pin N0 in the byte lane with the associated `dqs`.
2. Byte lanes are configured as either data or address/control.
   a. Pins N1 and N12 can be used for address/control in a data byte lane.
   b. No data signals (`dqs`, `dq`, `dm/dbi`) can be placed in an address/control byte lane.
3. Address/control can be on any of the 13 pins in the address/control byte lanes. Address/control must be contained within the same bank. Address/control must be in the centermost bank.

4. There is one `vr` pin per bank and DCI is required. DCI Cascade is not permitted. All rules for the DCI in the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3] must be followed.
5. `ck` must be on the PN pair in the center of the byte lane which is designated as the Upper byte clock pair.
6. `reset_n` can be on any pin as long as FPGA logic timing is met and I/O standard can be accommodated for the chosen bank.
7. Banks can be shared between two controllers.
   a. Each byte lane is dedicated to a specific controller (except for `reset_n`).
   b. Byte lanes from one controller cannot be placed inside the other. For example, with controllers A and B, "AABB" is allowed, while "ABAB" is not.
8. All I/O banks used by the memory interface must be in the same column.
9. Maximum height of interface is five contiguous banks for 144-bit wide interface.
10. Bank skipping is not allowed.
11. The input clock for the master PLL in the interface must come from the a clock capable pair in the I/O column used for the memory interface.
12. The dedicated V_{REF} pins in the banks used for DDR4 must be tied to ground with a 500Ω resistor. For more information, see the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3].
13. The interface must be contained within the same I/O bank type (High Range or High Performance). Mixing bank types is not permitted with the exceptions of the `reset_n` in step 6 above and the input clock mentioned in step 11 above.
14. The `par` input for command and address parity and the `alert_n` input/output are not supported by this interface. Consult the memory vendor for information on the proper connection for these pins when not used.

IMPORTANT: *Component interfaces should be created with the same component for all components in the interface. x16 components have a different number of bank groups than the x8 components. For example, a 72-bit wide component interface should be created by using nine x8 components or five x16 components where half of one component is not used. Four x16 components and one x8 component is not permissible.*

DDR4 Pinout Examples

Table 4-20 shows an example of a 32-bit DDR4 interface contained within two banks. This example is for a component interface using 4 × 8 DDR4 components.

Table 4-20: **32-Bit DDR4 Interface Contained in Two Banks**

Bank	Signal Name	Byte Group	I/O Type	Special Designation
Bank 1				
1	–	T3U_12	–	–
1	–	T3U_11	N	–
1	–	T3U_10	P	–
1	–	T3U_9	N	–
1	–	T3U_8	P	–
1	–	T3U_7	N	DBC-N
1	–	T3U_6	P	DBC-P
1	–	T3L_5	N	–
1	–	T3L_4	P	–
1	–	T3L_3	N	–
1	–	T3L_2	P	–
1	–	T3L_1	N	DBC-N
1	–	T3L_0	P	DBC-P
1	–	T2U_12	–	–
1	–	T2U_11	N	–
1	–	T2U_10	P	–
1	–	T2U_9	N	–
1	–	T2U_8	P	–
1	–	T2U_7	N	QBC-N
1	–	T2U_6	P	QBC-P
1	–	T2L_5	N	–
1	–	T2L_4	P	–
1	–	T2L_3	N	–
1	–	T2L_2	P	–
1	–	T2L_1	N	QBC-N
1	–	T2L_0	P	QBC-P
1	reset_n	T1U_12	–	–
1	dq31	T1U_11	N	–
1	dq30	T1U_10	P	–
1	dq29	T1U_9	N	–

Table 4-20: **32-Bit DDR4 Interface Contained in Two Banks *(Cont'd)***

Bank	Signal Name	Byte Group	I/O Type	Special Designation
1	dq28	T1U_8	P	–
1	dqs3_n	T1U_7	N	QBC-N
1	dqs3_p	T1U_6	P	QBC-P
1	dq27	T1L_5	N	–
1	dq26	T1L_4	P	–
1	dq25	T1L_3	N	–
1	dq24	T1L_2	P	–
1	unused	T1L_1	N	QBC-N
1	dm3/dbi3	T1L_0	P	QBC-P
1	vr	T0U_12	–	–
1	dq23	T0U_11	N	–
1	dq22	T0U_10	P	–
1	dq21	T0U_9	N	–
1	dq20	T0U_8	P	–
1	dqs2_n	T0U_7	N	DBC-N
1	dqs2_p	T0U_6	P	DBC-P
1	dq19	T0L_5	N	–
1	dq18	T0L_4	P	–
1	dq17	T0L_3	N	–
1	dq16	T0L_2	P	–
1	–	T0L_1	N	DBC-N
1	dm2/dbi2	T0L_0	P	DBC-P
Bank 2				
2	a0	T3U_12	–	–
2	a1	T3U_11	N	–
2	a2	T3U_10	P	–
2	a3	T3U_9	N	–
2	a4	T3U_8	P	–
2	a5	T3U_7	N	DBC-N
2	a6	T3U_6	P	DBC-P
2	a7	T3L_5	N	–
2	a8	T3L_4	P	–

Table 4-20: **32-Bit DDR4 Interface Contained in Two Banks *(Cont'd)***

Bank	Signal Name	Byte Group	I/O Type	Special Designation
2	a9	T3L_3	N	–
2	a10	T3L_2	P	–
2	a11	T3L_1	N	DBC-N
2	a12	T3L_0	P	DBC-P
2	a13	T2U_12	–	–
2	we_n/a14	T2U_11	N	–
2	cas_n/a15	T2U_10	P	–
2	ras_n/a16	T2U_9	N	–
2	act_n	T2U_8	P	–
2	ck_n	T2U_7	N	QBC-N
2	ck_p	T2U_6	P	QBC-P
2	ba0	T2L_5	N	–
2	ba1	T2L_4	P	–
2	bg0	T2L_3	N	–
2	bg1	T2L_2	P	–
2	pll refclk_n	T2L_1	N	QBC-N
2	pll refclk	T2L_0	P	QBC-P
2	cs_n	T1U_12	–	–
2	dq15	T1U_11	N	–
2	dq14	T1U_10	P	–
2	dq13	T1U_9	N	–
2	dq12	T1U_8	P	–
2	dqs1_n	T1U_7	N	QBC-N
2	dqs1_p	T1U_6	P	QBC-P
2	dq11	T1L_5	N	–
2	dq10	T1L_4	P	–
2	dq9	T1L_3	N	–
2	dq8	T1L_2	P	–
2	odt	T1L_1	N	QBC-N
2	dm1/dbi1	T1L_0	P	QBC-P

Table 4-20: **32-Bit DDR4 Interface Contained in Two Banks *(Cont'd)***

Bank	Signal Name	Byte Group	I/O Type	Special Designation
2	vr	T0U_12	–	–
2	dq7	T0U_11	N	–
2	dq6	T0U_10	P	–
2	dq5	T0U_9	N	–
2	dq4	T0U_8	P	–
2	dqs0_n	T0U_7	N	DBC-N
2	dqs0_p	T0U_6	P	DBC-P
2	dq3	T0L_5	N	–
2	dq2	T0L_4	P	–
2	dq1	T0L_3	N	–
2	dq0	T0L_2	P	–
2	cke	T0L_1	N	DBC-N
2	dm0/dbi0	T0L_0	P	DBC-P

Protocol Description

This core has the following interfaces:

- User Interface
- Native Interface

User Interface

The user interface is shown in Table 4-21 and connects to an FPGA user design to allow access to an external memory device.

Table 4-21: **User Interface**

Signal	Direction	Description
app_addr[ADDR_WIDTH – 1:0]	Input	This input indicates the address for the current request.
app_cmd[2:0]	Input	This input selects the command for the current request.
app_en	Input	This is the active-High strobe for the `app_addr[]`, `app_cmd[2:0]`, `app_sz`, and `app_hi_pri` inputs.
app_rdy	Output	This output indicates that the user interface is ready to accept commands. If the signal is deasserted when `app_en` is enabled, the current `app_cmd` and `app_addr` must be retried until `app_rdy` is asserted.
app_hi_pri	Input	This input is reserved and should be tied to 0.
app_rd_data [APP_DATA_WIDTH – 1:0]	Output	This provides the output data from read commands.
app_rd_data_end	Output	This active-High output indicates that the current clock cycle is the last cycle of output data on `app_rd_data[]`.
app_rd_data_valid	Output	This active-High output indicates that `app_rd_data[]` is valid.
app_sz	Input	This input is reserved and should be tied to `0`.
app_wdf_data [APP_DATA_WIDTH – 1:0]	Input	This provides the data for write commands.
app_wdf_end	Input	This active-High input indicates that the current clock cycle is the last cycle of input data on `app_wdf_data[]`.
app_wdf_mask [APP_MASK_WIDTH – 1:0]	Input	This provides the mask for `app_wdf_data[]`.
app_wdf_rdy	Output	This output indicates that the write data FIFO is ready to receive data. Write data is accepted when `app_wdf_rdy` = 1'b1 and `app_wdf_wren` = 1'b1.
app_wdf_wren	Input	This is the active-High strobe for `app_wdf_data[]`.

Table 4-21: **User Interface *(Cont'd)***

Signal	Direction	Description
app_correct_en_i	Input	When asserted, this active-High signal corrects single bit data errors. This input is valid only when ECC is enabled in the Vivado Integrated Design Environment (IDE). In the example design, this signal is always tied to 1.
app_sr_req	Input	This input is reserved and should be tied to 0.
app_sr_active	Output	This output is reserved.
app_ref_req	Input	This input is reserved and should be tied to 0.
app_ref_ack	Output	This output is reserved.
app_zq_req	Input	This input is reserved and should be tied to 0.
app_zq_ack	Output	This output is reserved.
ui_clk	Output	This user interface clock must be a half or quarter of the DRAM clock.
init_calib_complete	Output	PHY asserts `init_calib_complete` when calibration is finished.
ui_clk_sync_rst	Output	This is the active-High user interface reset.

app_addr[ADDR_WIDTH – 1:0]

This input indicates the address for the request currently being submitted to the user interface. The user interface aggregates all the address fields of the external SDRAM and presents a flat address space.

app_cmd[2:0]

This input specifies the command for the request currently being submitted to the user interface. The available commands are shown in Table 4-22.

Table 4-22: **Commands for app_cmd[2:0]**

Operation	app_cmd[2:0] Code
Write	000
Read	001

app_en

This input strobes in a request. Apply the desired values to `app_addr`[], `app_cmd`[2:0], and `app_hi_pri`, and then assert `app_en` to submit the request to the user interface. This initiates a handshake that the user interface acknowledges by asserting `app_rdy`.

app_wdf_data[APP_DATA_WIDTH – 1:0]

This bus provides the data currently being written to the external memory.

app_wdf_end

This input indicates that the data on the app_wdf_data[] bus in the current cycle is the last data for the current request.

app_wdf_mask[APP_MASK_WIDTH – 1:0]

This bus indicates which bits of app_wdf_data[] are written to the external memory and which bits remain in their current state.

app_wdf_wren

This input indicates that the data on the app_wdf_data[] bus is valid.

app_rdy

This output indicates whether the request currently being submitted to the user interface is accepted. If the user interface does not assert this signal after app_en is asserted, the current request must be retried. The app_rdy output is not asserted if:

- PHY/Memory initialization is not yet completed.
- All the bank machines are occupied (can be viewed as the command buffer being full).
  - A read is requested and the read buffer is full.
  - A write is requested and no write buffer pointers are available.
- A periodic read is being inserted.

app_rd_data[APP_DATA_WIDTH – 1:0]

This output contains the data read from the external memory.

app_rd_data_end

This output indicates that the data on the app_rd_data[] bus in the current cycle is the last data for the current request.

app_rd_data_valid

This output indicates that the data on the app_rd_data[] bus is valid.

app_wdf_rdy

This output indicates that the write data FIFO is ready to receive data. Write data is accepted when both app_wdf_rdy and app_wdf_wren are asserted.

ui_clk_sync_rst

This is the reset from the user interface which is in synchronous with `ui_clk`.

ui_clk

This is the output clock from the user interface. It must be a half or quarter the frequency of the clock going out to the external SDRAM, which depends on 2:1 or 4:1 mode selected in Vivado IDE.

init_calib_complete

PHY asserts init_calib_complete when calibration is finished. The application has no need to wait for `init_calib_complete` before sending commands to the Memory Controller.

Command Path

When the user logic `app_en` signal is asserted and the `app_rdy` signal is asserted from the user interface, a command is accepted and written to the FIFO by the user interface. The command is ignored by the user interface whenever `app_rdy` is deasserted. The user logic needs to hold `app_en` High along with the valid command and address values until `app_rdy` is asserted as shown in Figure 4-21.

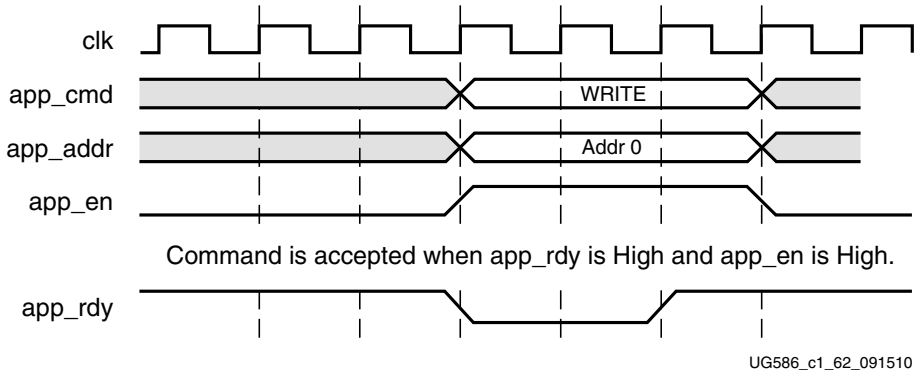

Figure 4-21: **User Interface Command Timing Diagram with app_rdy Asserted**

A non back-to-back write command can be issued as shown in Figure 4-22. This figure depicts three scenarios for the `app_wdf_data`, `app_wdf_wren`, and `app_wdf_end` signals, as follows:

1. Write data is presented along with the corresponding write command (second half of BL8).
2. Write data is presented before the corresponding write command.
3. Write data is presented after the corresponding write command, but should not exceed the limitation of two clock cycles.

For write data that is output after the write command has been registered, as shown in Note 3 (Figure 4-22), the maximum delay is two clock cycles.

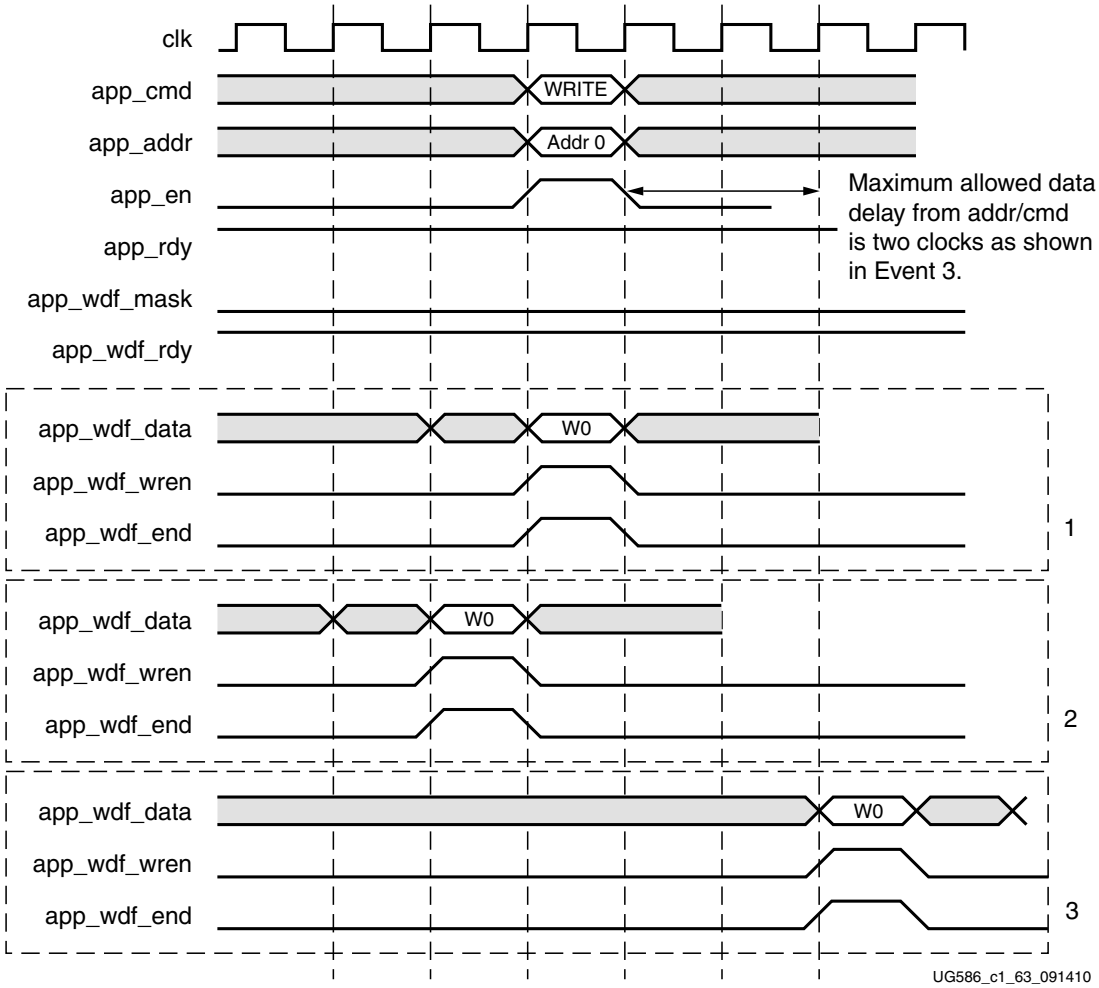

Figure 4-22: **4:1 Mode User Interface Write Timing Diagram (Memory Burst Type = BL8)**

Write Path

The write data is registered in the write FIFO when `app_wdf_wren` is asserted and `app_wdf_rdy` is High (Figure 4-23). If `app_wdf_rdy` is deasserted, the user logic needs to hold `app_wdf_wren` and `app_wdf_end` High along with the valid `app_wdf_data` value until `app_wdf_rdy` is asserted. The `app_wdf_mask` signal can be used to mask out the bytes to write to external memory.

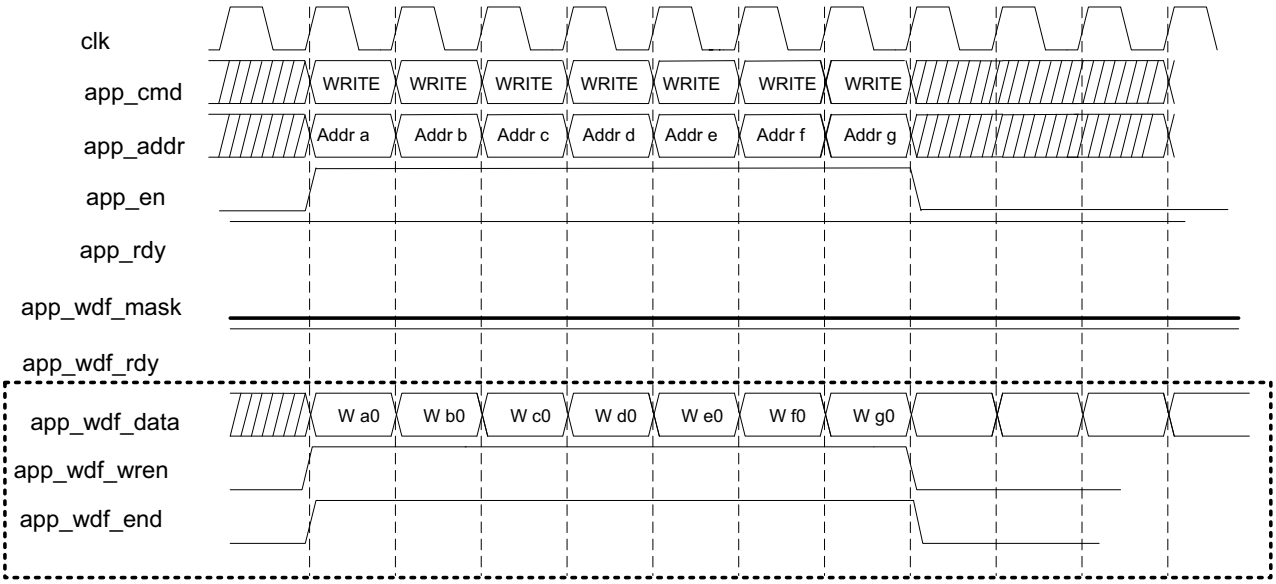

Figure 4-23: **4:1 Mode User Interface Back-to-Back Write Commands Timing Diagram (Memory Burst Type = BL8)**

As shown in Figure 4-21, page 58, the maximum delay for a single write between the write data and the associated write command is two clock cycles. When issuing back-to-back write commands, there is no maximum delay between the write data and the associated back-to-back write command, as shown in Figure 4-24.

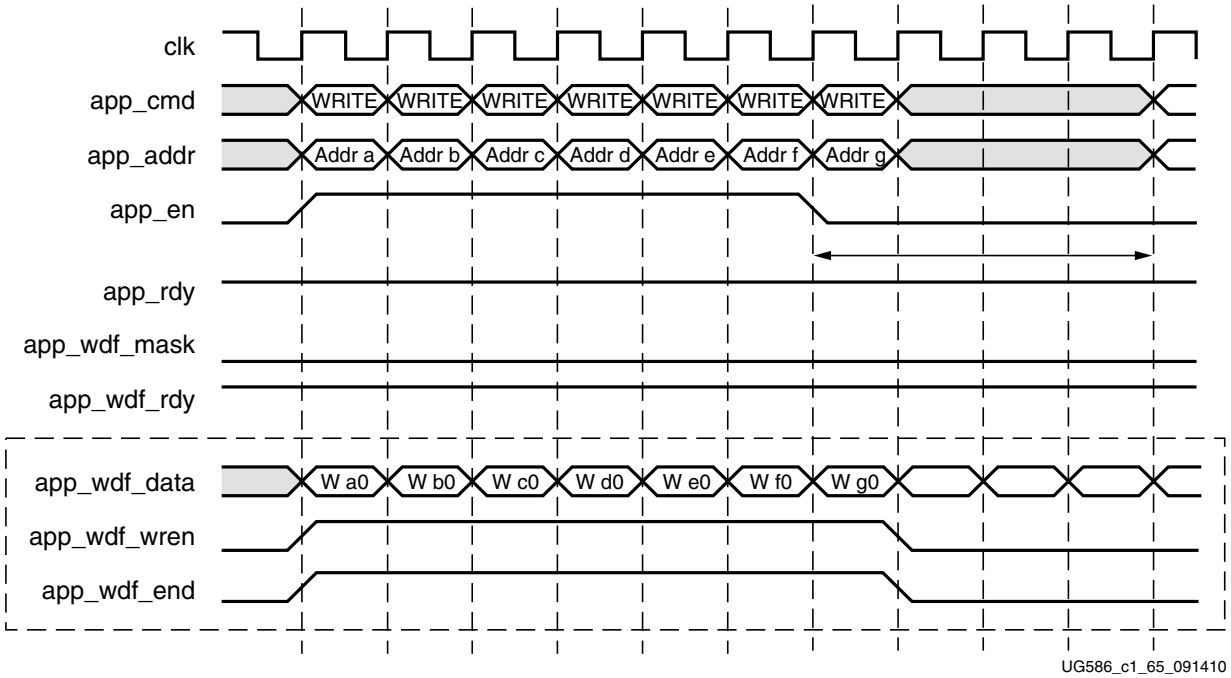

Figure 4-24: **4:1 Mode User Interface Back-to-Back Write Commands Timing Diagram (Memory Burst Type = BL8)**

The app_wdf_end signal must be used to indicate the end of a memory write burst. For memory burst types of eight in 2:1 mode, the app_wdf_end signal must be asserted on the second write data word.

The map of the application interface data to the DRAM output data can be explained with an example.

For a 4:1 Memory Controller to DRAM clock ratio with an 8-bit memory, at the application interface, if the 64-bit data driven is 0000_0806_0000_0805 (Hex), the data at the DRAM interface is as shown in Figure 4-25. This is for a BL8 (Burst Length 8) transaction.

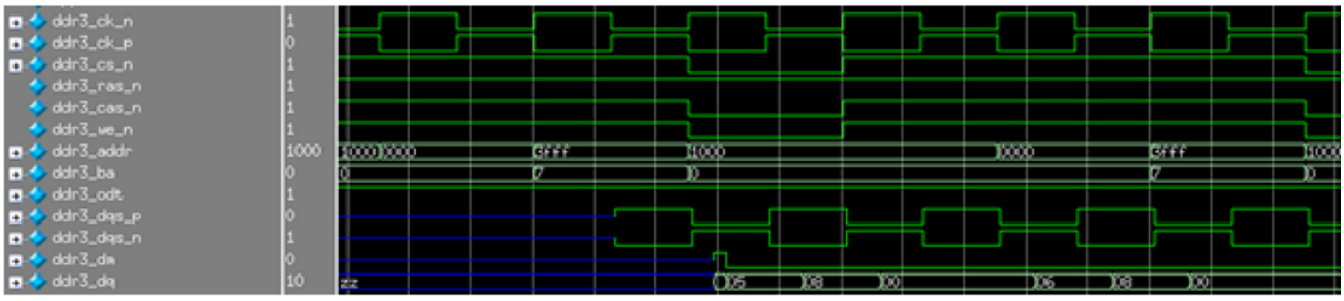

Figure 4-25: **Data at the DRAM Interface for 4:1 Mode**

The data values at different clock edges are as shown in Table 4-23.

Table 4-23: **Data Values at Different Clock Edges**

Rise0	Fall0	Rise1	Fall1	Rise2	Fall2	Rise3	Fall3
05	08	00	00	06	08	00	00

For a 2:1 Memory Controller to DRAM clock ratio, the application data width is 32 bits. Hence for BL8 transactions, the data at the application interface must be provided in two clock cycles. The app_wdf_end signal is asserted for the second data as shown in Figure 4-26. In this case, the application data provided in the first cycle is 0000_0405 (Hex), and the data provided in the last cycle is 0000_080A (Hex). This is for a BL8 transaction.

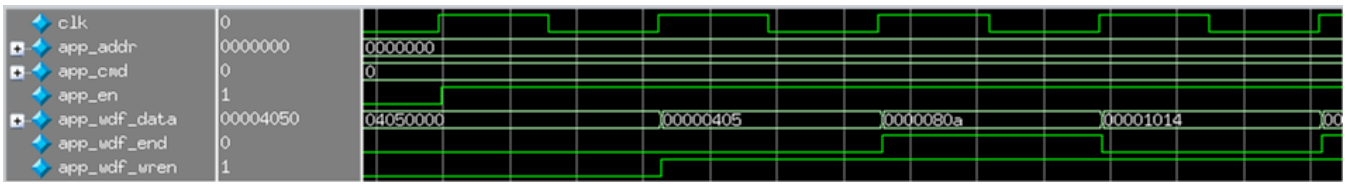

Figure 4-26: **Data at the Application Interface for 2:1 Mode**

Figure 4-27 shows the corresponding data at the DRAM interface.

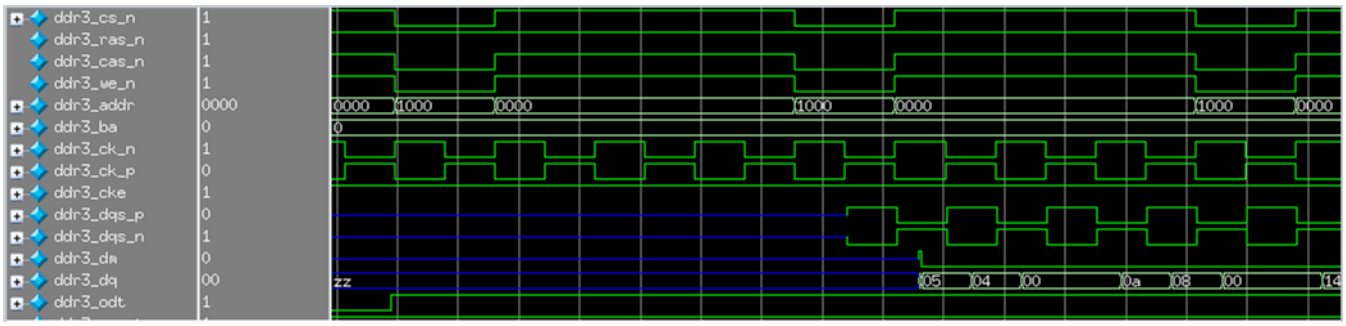

Figure 4-27: **Data at the DRAM Interface for 2:1 Mode**

Read Path

The read data is returned by the user interface in the requested order and is valid when `app_rd_data_valid` is asserted (Figure 4-28 and Figure 4-29). The `app_rd_data_end` signal indicates the end of each read command burst and is not needed in user logic.

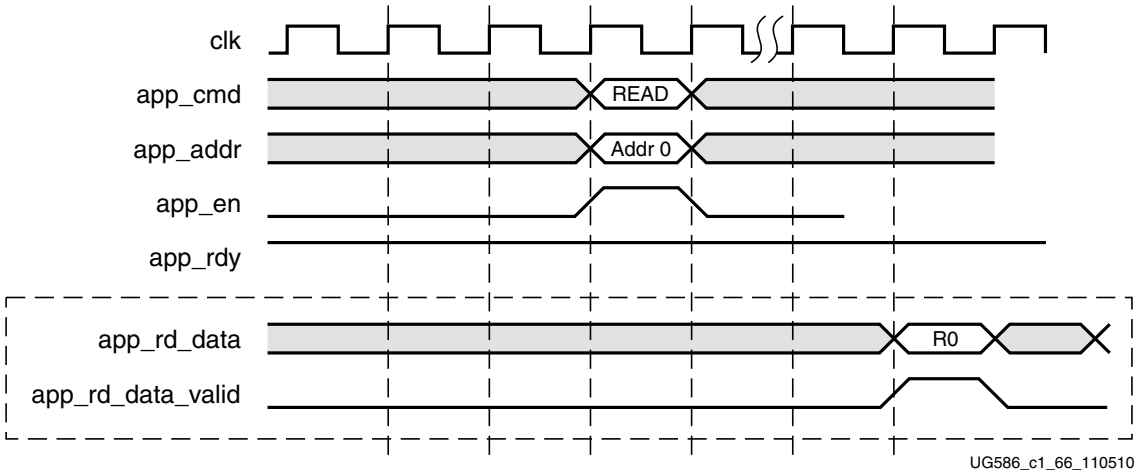

Figure 4-28: **4:1 Mode User Interface Read Timing Diagram (Memory Burst Type = BL8)**

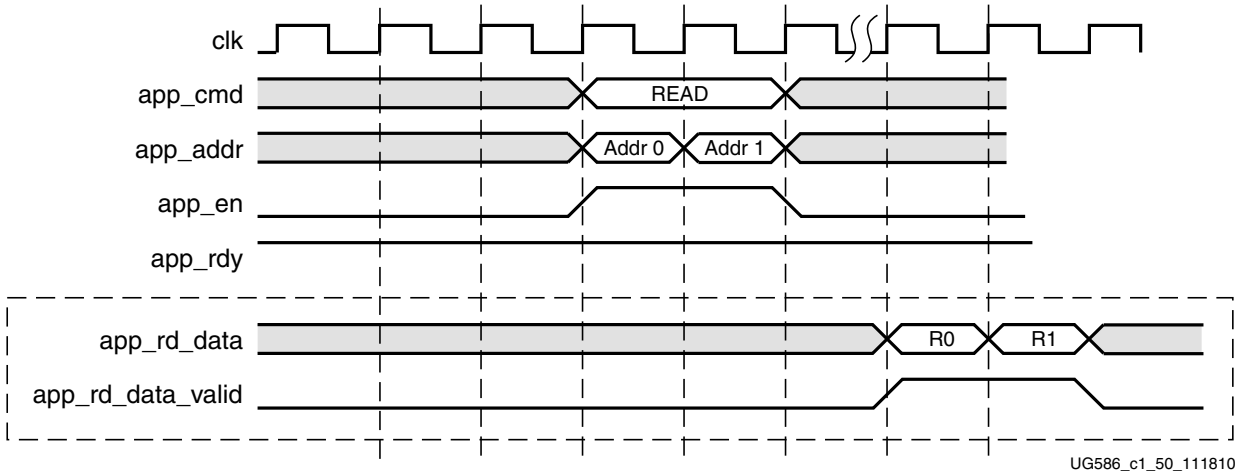

Figure 4-29: **4:1 Mode User Interface Read Timing Diagram (Memory Burst Type = BL4 or BL8)**

In Figure 4-29, the read data returned is always in the same order as the requests made on the address/control bus.

Native Interface

The native interface connects to an FPGA user design to allow access to an external memory device.

Command Request Signals

The native interface provides a set of signals that request a read or write command from the Memory Controller to the memory device. These signals are summarized in Table 4-24.

Table 4-24: **Native Interface Command Signals**

Signal	Direction	Description
accept	Output	This output indicates that the memory interface accepts the request driven on the last cycle.
bank[2:0] (DDR3), bank[1:0] (DDR4)	Input	This input selects the bank for the current request.
group (DDR4)	Input	This input selects the bank group for the current request.
cmd[2:0]	Input	This input selects the command for the current request.
col[COL_WIDTH – 1:0]	Input	This input selects the column address for the current request.
data_buf_addr[7:0]	Input	This input indicates the data buffer address where the Memory Controller: • Locates data while processing write commands. • Places data while processing read commands.
hi_priority	Input	This input is reserved and should be connected to logic 0.
precharge	Input	This input sets the precharge mode of operation in the controller.

Table 4-24: **Native Interface Command Signals *(Cont'd)***

Signal	Direction	Description
rank[]	Input	This input is reserved and should be connected to logic 0.
row[ROW_WIDTH – 1:0]	Input	This input selects the row address for the current request.
use_addr	Input	The user design strobes this input to indicate that the request information driven on the previous state is valid.

The bank, row, and column comprise a target address on the memory device for read and write operations. Commands are specified using the cmd[2:0] input to the core. The available read and write commands are shown in Table 4-25.

Table 4-25: **Memory Interface Commands**

Operation	cmd[2:0] Code
Memory write	000
Memory read	001
Reserved	All other codes

accept

This signal indicates to the user design whether or not a request is accepted by the core. When the accept signal is asserted, the request submitted on the last cycle is accepted, and the user design can either continue to submit more requests or go idle. When the accept signal is deasserted, the request submitted on the last cycle was not accepted and must be retried.

use_addr

The user design asserts the `use_addr` signal to strobe the request that was submitted to the native interface on the previous cycle.

data_buf_addr

The user design must contain a buffer for data used during read and write commands. When a request is submitted to the native interface, the user design must designate a location in the buffer for when the request is processed. For write commands, data_buf_addr is an address in the buffer containing the source data to be written to the external memory. For read commands, data_buf_addr is an address in the buffer that receives read data from the external memory. The core echoes this address back when the requests are processed.

Precharge

The precharge signal provides a transaction by transaction control over the auto-precharge feature of the Memory Controller.

There are three modes of operation:

- **precharge = 0** – The controller behaves in "keep open" mode. Pages are kept open until a different page is opened in the same bank.
- **precharge = 1** – The controller behaves in "keep closed" mode. This mode is useful for certain special access patterns.
- **precharge = By Transaction** – A page is closed after the transaction if precharge = 1 and it is kept open if precharge = 0. The user of the native interface has transaction by transaction control.

Write Command Signals

The native interface has signals that are used when the Memory Controller is processing a write command (Table 4-26). These signals connect to the control, address, and data signals of a buffer in the user design.

Table 4-26: **Native Interface Write Command Signals**

Signal	Direction	Description
wr_data[2 × nCK_PER_CLK × PAYLOAD_WIDTH – 1:0]	Input	This is the input data for write commands.
wr_data_addr [DATA_BUF_ADDR_WIDTH – 1:0]	Output	This output provides the base address for the source data buffer for write commands.
wr_data_mask[2 × nCK_PER_CLK × DATA_WIDTH/8 – 1:0]	Input	This input provides the byte enable for the write data.
wr_data_en	Output	This output indicates that the memory interface is reading data from a data buffer for a write command.
wr_data_offset[0:0]	Output	This output provides the offset for the source data buffer for write commands.

wr_data

This bus is the data that needs to be written to the external memory. This bus can be connected to the data output of a buffer in the user design.

wr_data_addr

This bus is an echo of `data_buf_addr` when the current write request is submitted. The `wr_data_addr` bus can be combined with the `wr_data_offset` signal and applied to the address input of a buffer in the user design.

wr_data_mask

This bus is the byte enable (data mask) for the data currently being written to the external memory. The byte to the memory is written when the corresponding `wr_data_mask` signal is deasserted.

wr_data_en

When asserted, this signal indicates that the core is reading data from the user design for a write command. This signal can be tied to the chip select of a buffer in the user design.

wr_data_offset

This bus is used to step through the data buffer when the burst length requires more than a single cycle to complete. This bus, in combination with `wr_data_addr`, can be applied to the address input of a buffer in the user design.

Read Command Signals

The native interface provides a set of signals used when the Memory Controller is processing a read command (Table 4-27). These signals are similar to those for processing write commands, except that they transfer data from the memory device to a buffer in the user design.

Table 4-27: **Native Interface Read Command Signals**

Signal	Direction	Description
rd_data[2 × nCK_PER_CLK × PAYLOAD_WIDTH – 1:0]	Output	This is the output data from read commands.
rd_data_addr[DATA_BUF_ADDR_WIDTH – 1:0]	Output	This output provides the base address of the destination buffer for read commands.
rd_data_en	Output	This output indicates that valid read data is available on the `rd_data` bus.
rd_data_offset[1:0]	Output	This output provides the offset for the destination buffer for read commands.

rd_data

This bus is the data that was read from the external memory. It can be connected to the data input of a buffer in the user design.

rd_data_addr

This bus is an echo of `data_buf_addr` when the current read request is submitted. This bus can be combined with the `rd_data_offset` signal and applied to the address input of a buffer in the user design.

rd_data_en

This signal indicates when valid read data is available on `rd_data` for a read request. It can be tied to the chip select and write enable of a buffer in the user design.

rd_data_offset

This bus is used to step through the data buffer when the burst length requires more than a single cycle to complete. This bus can be combined with `rd_data_addr` and applied to the address input of a buffer in the user design.

Native Interface Maintenance Command Signals

Table 4-28 lists the native interface maintenance command signals.

Table 4-28: **Native Interface Maintenance Command Signals**

Signal	Direction	Description
app_sr_req	Input	This input is reserved and should be tied to 0.
app_sr_active	Output	This output is reserved.
app_ref_req	Input	This input is reserved and should be tied to 0.
app_ref_ack	Output	This output is reserved.
app_zq_req	Input	This input is reserved and should be tied to 0.
app_zq_ack	Output	This output is reserved.

app_ref_req

This input is reserved and should be tied to 0.

app_ref_ack

This output is reserved.

app_zq_req

This input is reserved and should be tied to 0.

app_zq_ack

This output is reserved.

Native Interface Protocol

The native interface protocol is shown in Figure 4-30.

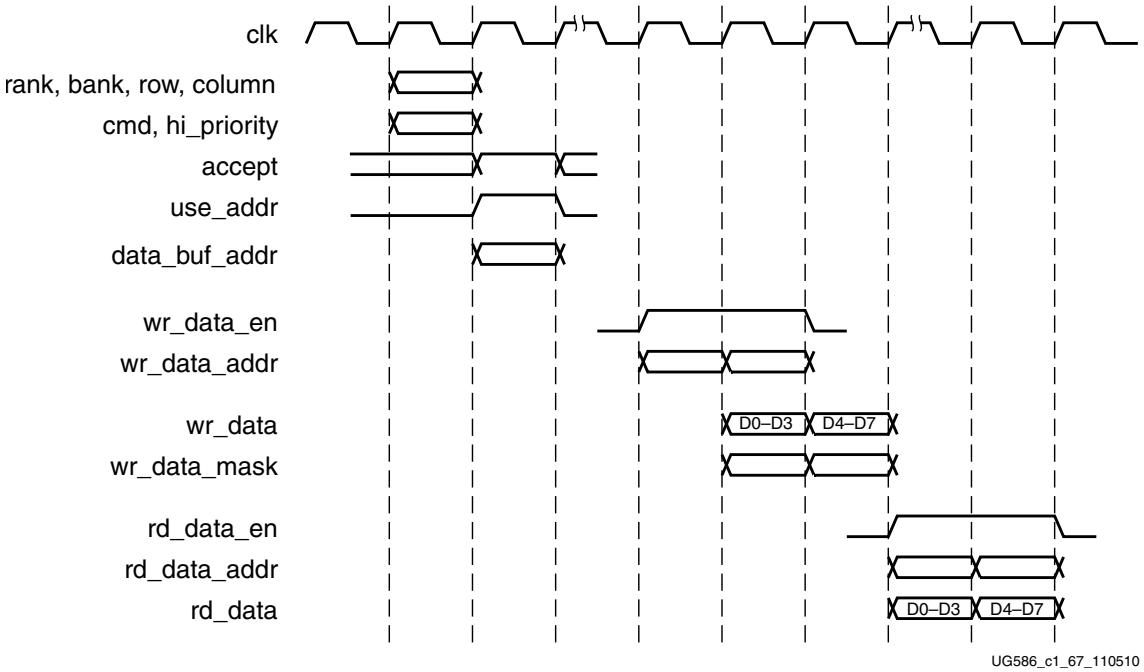

Figure 4-30: **Native Interface Protocol**

Requests are presented to the native interface as an address and a command. The address is composed of the bank, row, and column inputs. The command is encoded on the cmd input.

The address and command are presented to the native interface one state before they are validated with the `use_addr` signal. The memory interface indicates that it can accept the request by asserting the accept signal. Requests are confirmed as accepted when `use_addr` and accept are both asserted in the same clock cycle. If `use_addr` is asserted but accept is not, the request is not accepted and must be repeated. This behavior is shown in Figure 4-31.

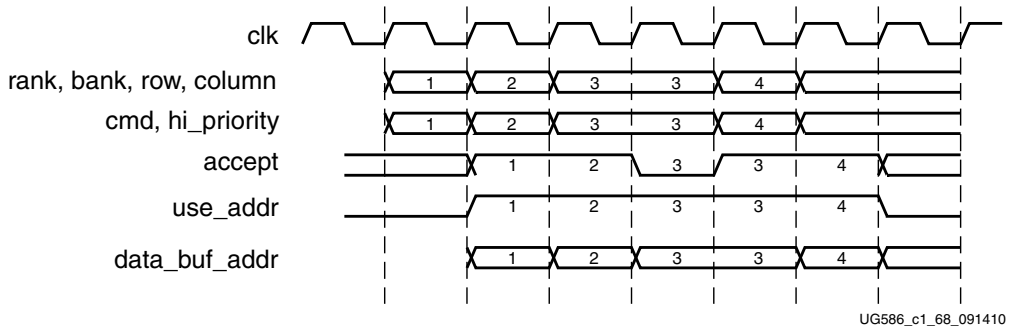

Figure 4-31: **Native Interface Flow Control**

In Figure 4-31, requests 1 and 2 are accepted normally. The first time request 3 is presented, accept is driven Low, and the request is not accepted. The user design retries request 3, which is accepted on the next attempt. Request 4 is subsequently accepted on the first attempt.

The data_buf_addr bus must be supplied with requests. This bus is an address pointer into a buffer that exists in the user design. It tells the core where to locate data when processing write commands and where to place data when processing read commands. When the core processes a command, the core echoes `data_buf_addr` back to the user design by `wr_data_addr` for write commands and rd_data_addr for read commands. This behavior is shown in Figure 4-32. Write data must be supplied in the same clock cycle that `wr_data_en` is asserted.

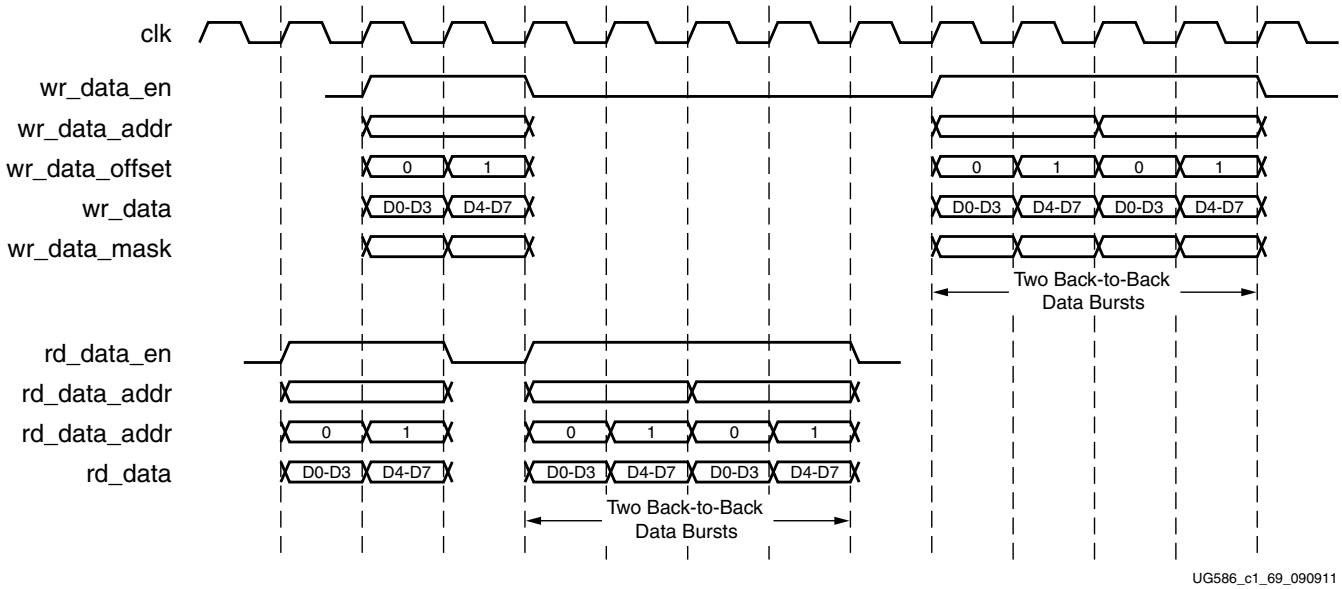

Figure 4-32: **Command Processing**

Transfers can be isolated with gaps of non-activity, or there can be long bursts with no gaps. The user design can identify when a request is being processed and when it finishes by monitoring the `rd_data_en` and `wr_data_en` signals. When the `rd_data_en` signal is asserted, the Memory Controller has completed processing a read command request. Similarly, when the `wr_data_en` signal is asserted, the Memory Controller is processing a write command request.

When NORM ordering mode is enabled, the Memory Controller reorders received requests to optimize throughput between the FPGA and memory device. The data is returned to the user design in the order processed, not the order received. The user design can identify the specific request being processed by monitoring `rd_data_addr` and `wr_data_addr`. These fields correspond to the `data_buf_addr` supplied when the user design submits the request to the native interface. Both of these scenarios are depicted in Figure 4-32.

The native interface is implemented such that the user design must submit one request at a time and, thus, multiple requests must be submitted in a serial fashion. Similarly, the core must execute multiple commands to the memory device one at a time. However, due to pipelining in the core implementation, read and write requests can be processed in parallel at the native interface.

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl Console.

Vivado Integrated Design Environment

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or popup menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

Controller Options

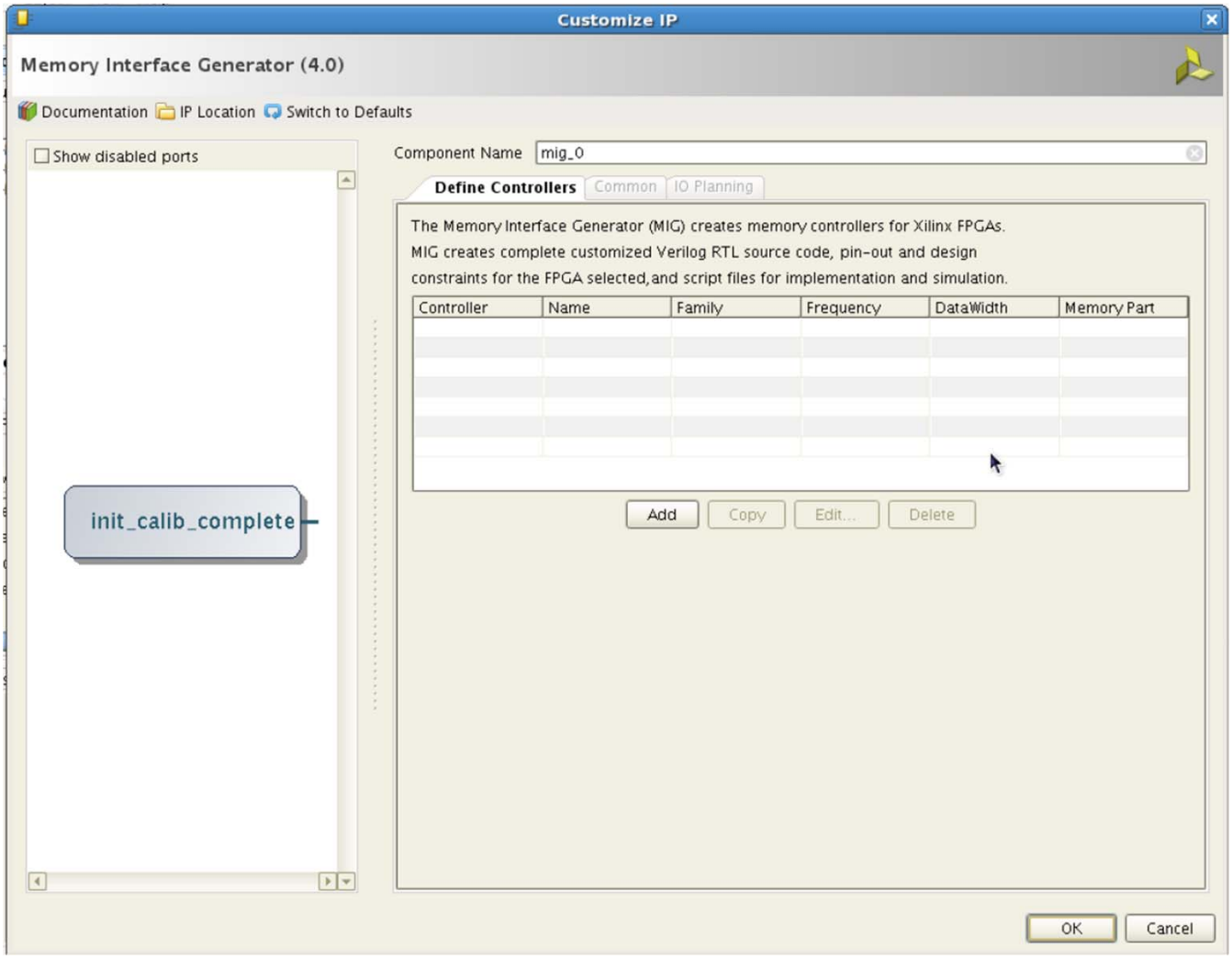

Figure 5-1: **Vivado Customize IP Dialog Box – Define Controllers**

Only one controller instance can be created and only two kinds of controllers are available for instantiation:

- DDR3
- DDR4

1. After a controller is added with the **Add** button, the **Common** and **I/O Planning** tabs are enabled.
2. **Copy** and **Delete** are not available in the release.

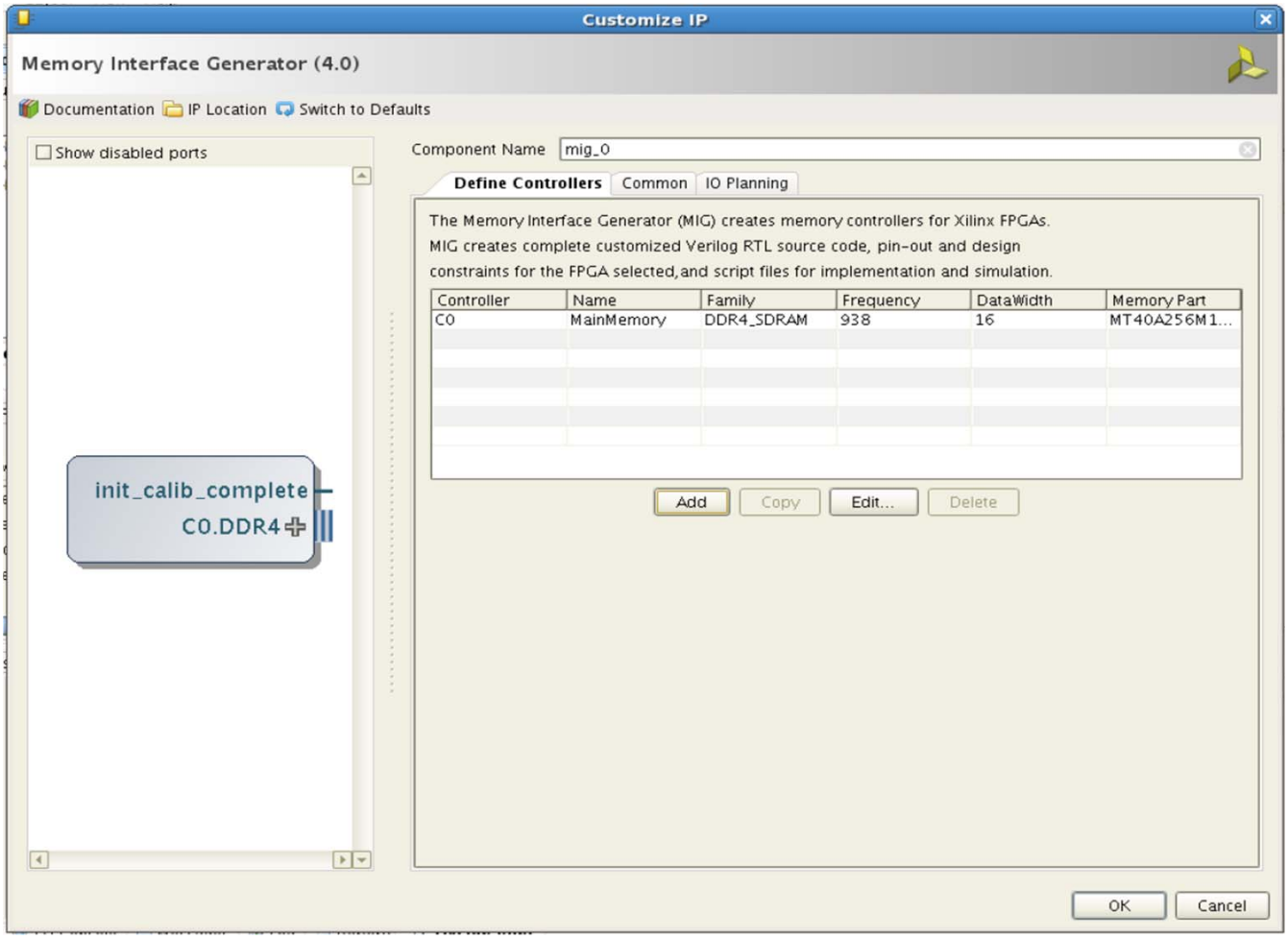

Figure 5-2: **Vivado Customize IP Dialog Box – Common and I/O Planning**

3. After adding a controller, select the controller row in the table and click the **Edit** button to edit controller options. Controller options are divided into two tabs as shown in Figure 5-3 and Figure 5-4.

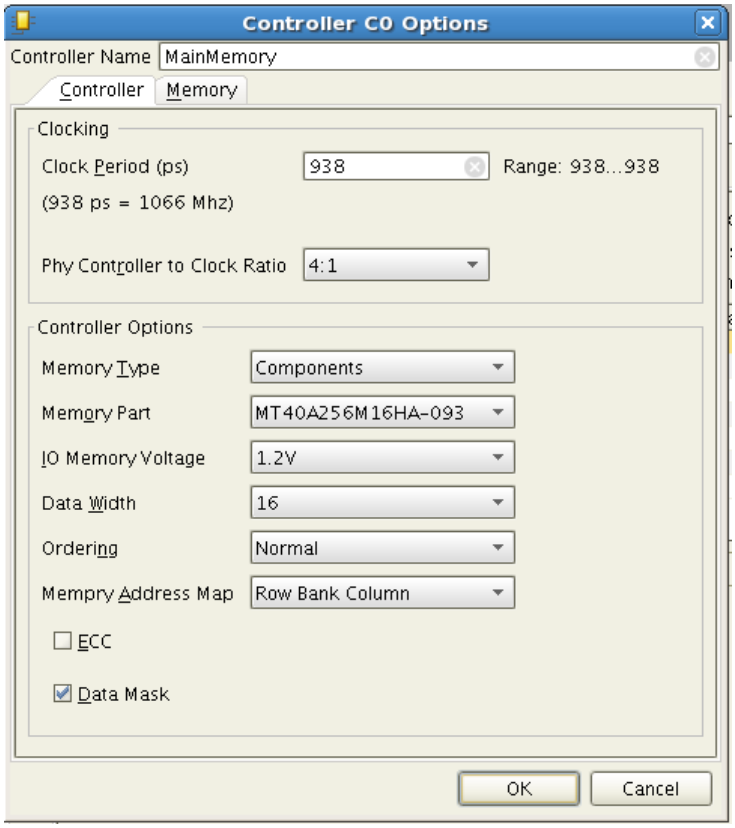

Figure 5-3: **Controller CO Options Dialog Box – Controller**

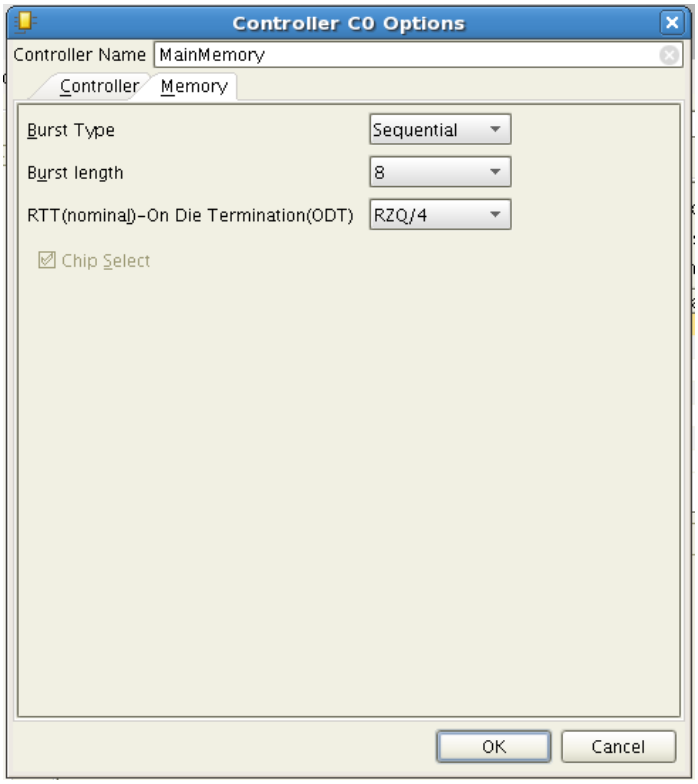

Figure 5-4: **Controller CO Options Dialog Box – Memory**

IMPORTANT: *All parameters shown in the controller options dialog are limited selection options in this release.*

MIG I/O Planning

I/O Planning tab provides options to modify pin allocation for controllers. It also shows the resource summary of the available I/O ports, allocated I/O ports, and others.

Two methods of performing pin assignment are available:

- Bank Assignment
- Individual Pin Assignment

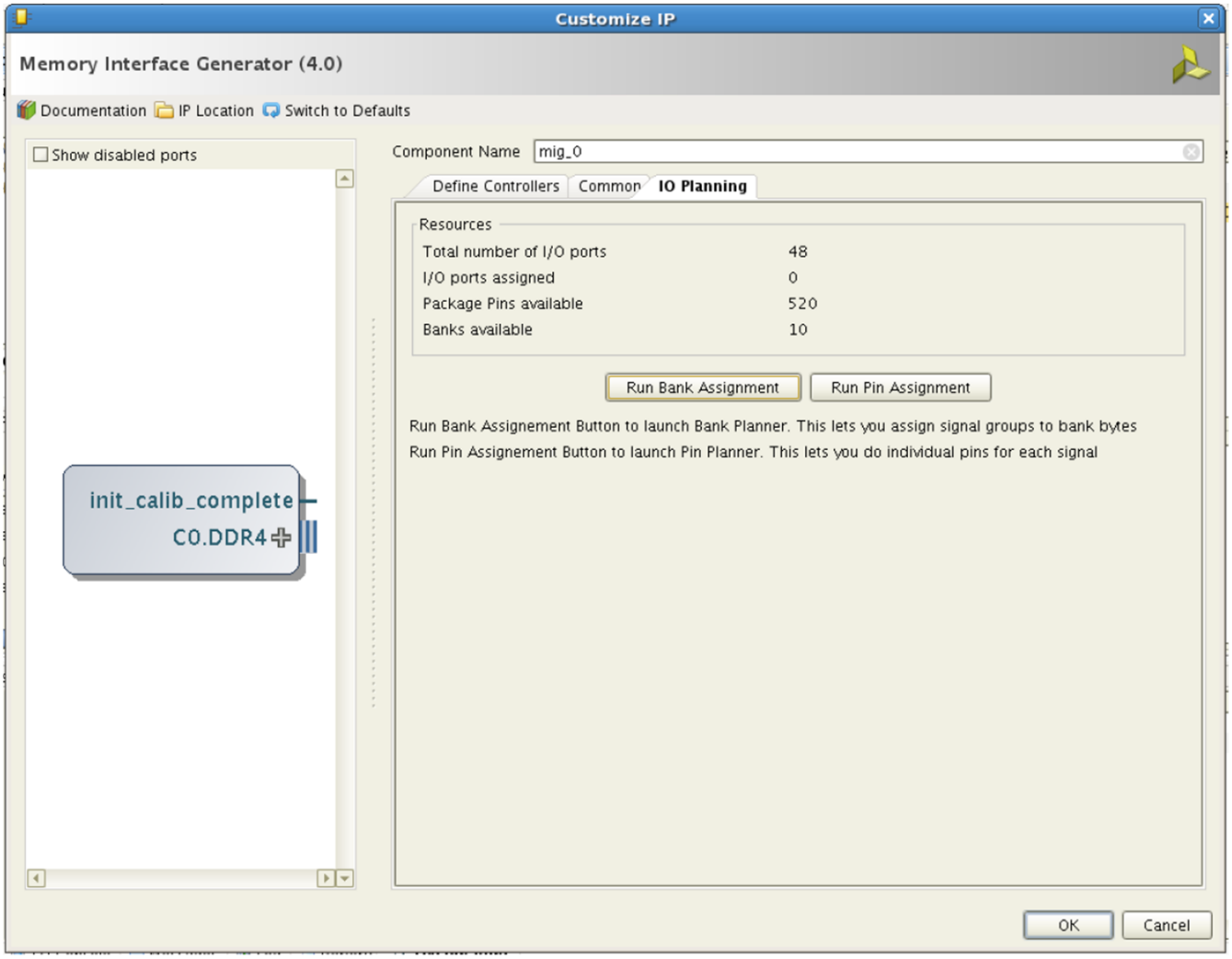

Figure 5-5: **Vivado Customize IP Dialog Box – I/O Planning**

Bank Assignment dialog provides a method to assign bank bytes to signal groups. At the first level, it shows the list of I/O banks available for MIG. All available byte groups are shown inside each bank. Warnings are issued regarding rule violations by a change in color of the selected options and the log window at the bottom of the dialog.

IMPORTANT: *More alternate views to modify bank/byte allocation in subsequent releases.*

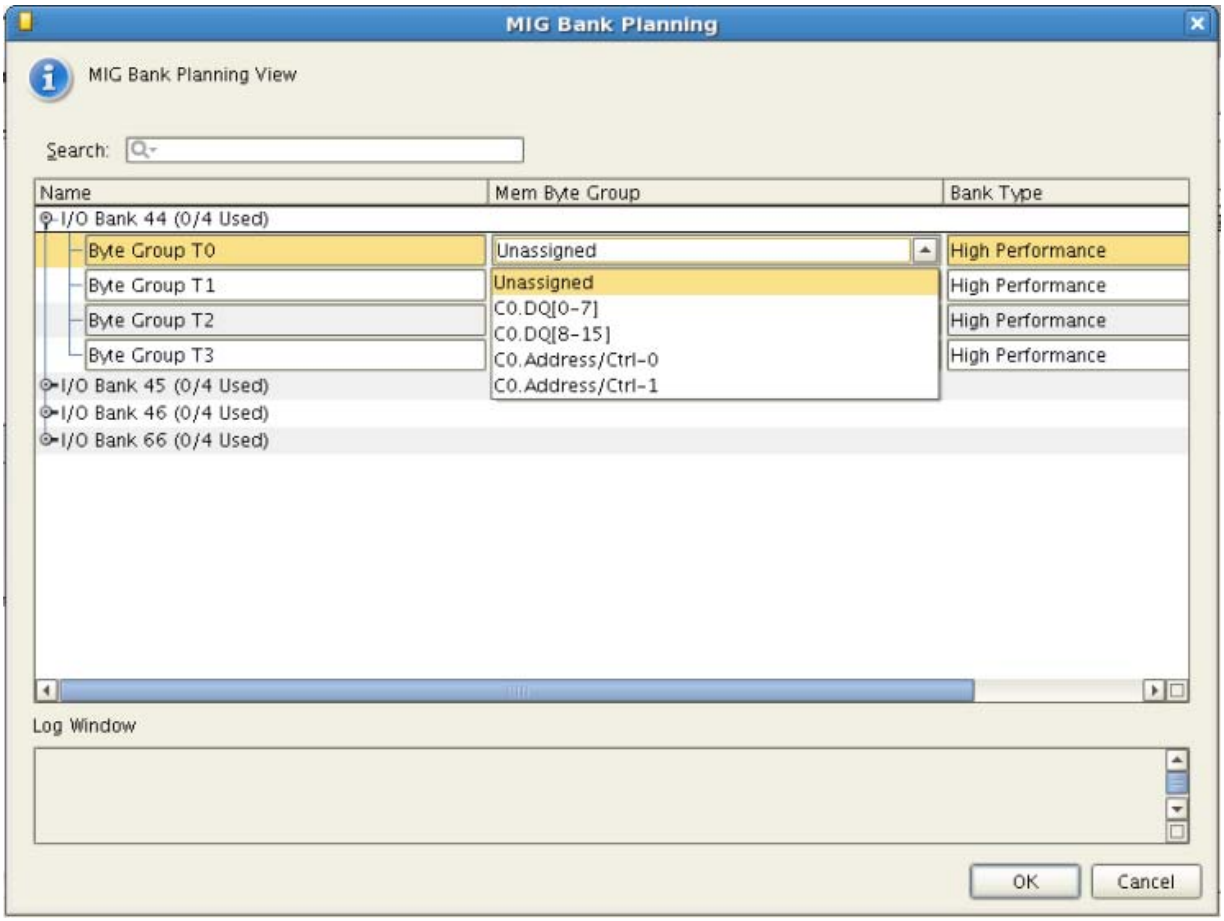

Figure 5-6: **MIG Bank Planning Dialog Box**

Pin swapping or completely new pin assignment is possible through the various methods provided by the pin planner.

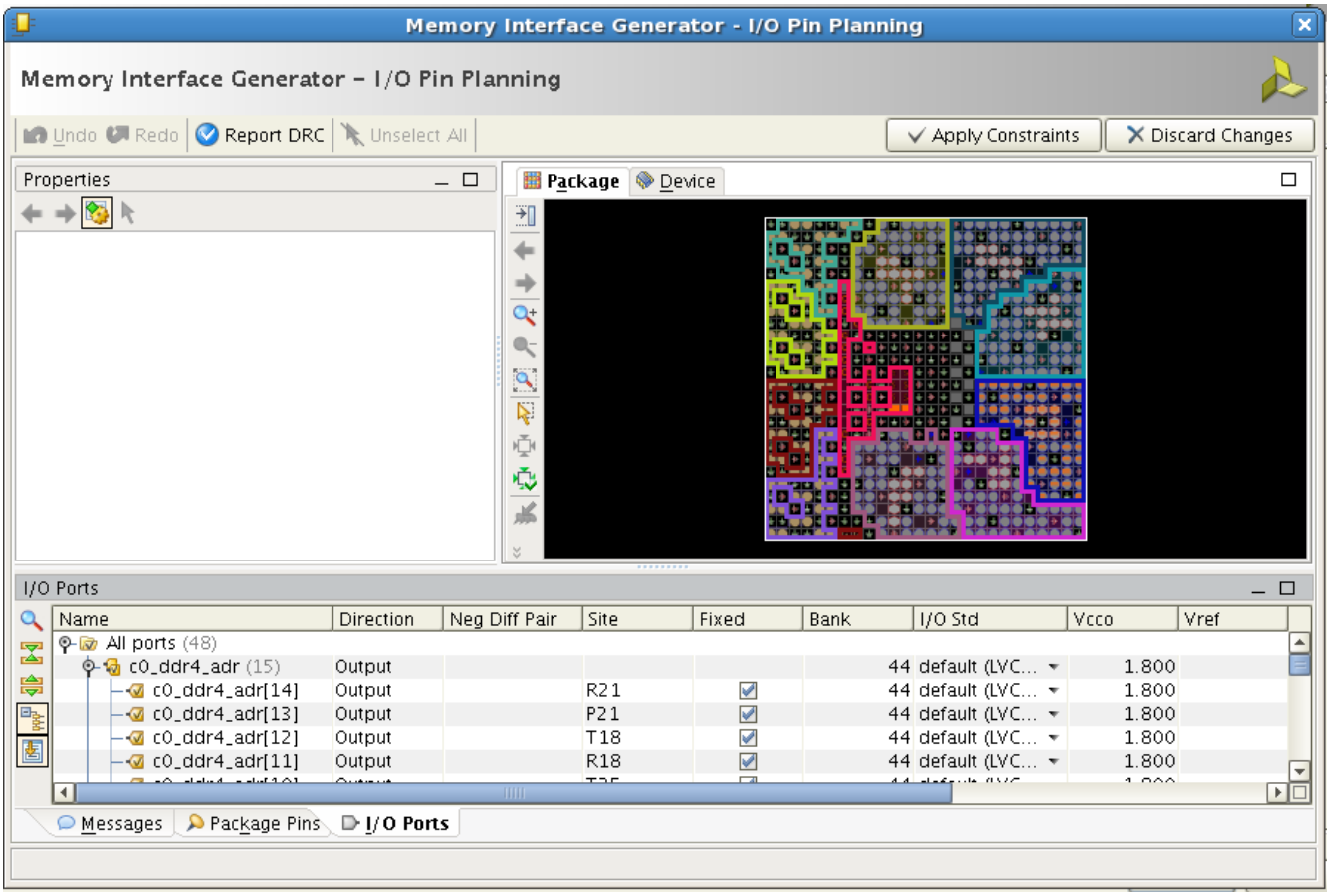

Figure 5-7: **MIG I/O Pin Planning Dialog Box**

Limitations

There are certain limitations for this release. These are relaxed through the production release in a staged fashion.

1. System clock selection is available through Pin Planner Lite with additional DRC for 2013.4.
2. Only one controller instance can be created. Multi-controller is not supported.
3. Only DDR3, DDR4, and RLDRAM 3 controllers are available.
4. AXI interface is not available.
5. Simulation model is not generated.
6. User selection of data width is supported up to 80-bit for DDR3 or DDR4. For RLDRAM 3, 36-bit data width is supported.
7. Memory interfaces can be assigned to High Performance (HP) or High Range (HR) banks depending on design requirements.
8. Memory port allocation can spread across five banks for DDR3 or DDR4 interfaces. Whereas for RLDRAM 3, span is limited to two banks.
9. Example design with Logic Debug cores (ILA, VIO) is not available.
10. Memory clock to FPGA logic clock ratio is restricted to 4:1.
11. DCI cascade is always enabled for DDR4 interfaces.
12. Internal V_{REF} is always required for DDR4 interfaces. User selection in the Vivado IDE will not have any effect.
13. Calibration block is included but calibration routines are disabled. Fixed delays are provided for proper write and read operations.
14. The dialog box for I/O planning at a pin-level takes about eight seconds to open up.
15. While the I/O planning view allows for changes to I/O Standard for pin locations, these are enabled for selection with DRCs.
16. There is no provision for reading an existing constraints file for importing I/O locations.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Constraining the Core

This chapter contains information about constraining the core in the Vivado® Design Suite environment.

Required Constraints

The MIG Vivado IDE generates the required constraints. A location constraint and an I/O standard constraint are added for each external pin in the design. The location is chosen by the Vivado IDE according to the banks and byte lanes chosen for the design.

The I/O standard is chosen by the memory type selection and options in the Vivado IDE and by the pin type. A sample for `dq`[0] is shown here.

```
set_property LOC IOB_X0Y2 [get_ports {DDR4_DQ[0]}]
set_property IOSTANDARD POD12_DCI [get_ports {DDR4_DQ[0]}]
```

Internal V_{REF} is always used for DDR4. Internal V_{REF} is optional for DDR3. A sample for DDR4 is shown here.

```
set_property INTERNAL_VREF 0.600 [get_iobanks 45]
```

The system clock must have the period set properly:

```
create_clock -name c0_sys_clk -period.938 [get_ports c0_sys_clk_p]
```

IMPORTANT: *Do not alter these constraints. If the pin locations need to be altered, rerun the MIG Vivado IDE to generate a new XDC file.*

I/O Standard and Placement

The MIG tool generates the appropriate I/O standards and placement based on the selections made in the Vivado IDE for the interface type and options.

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado® Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Example Design

This chapter contains information about the provided example design in the Vivado® Design Suite environment.

Vivado supports Open IP Example Design flow. To create the example design using this flow, right-click the IP in the **Source Window**, as shown in Figure 9-1 and select **Open IP Example Design**.

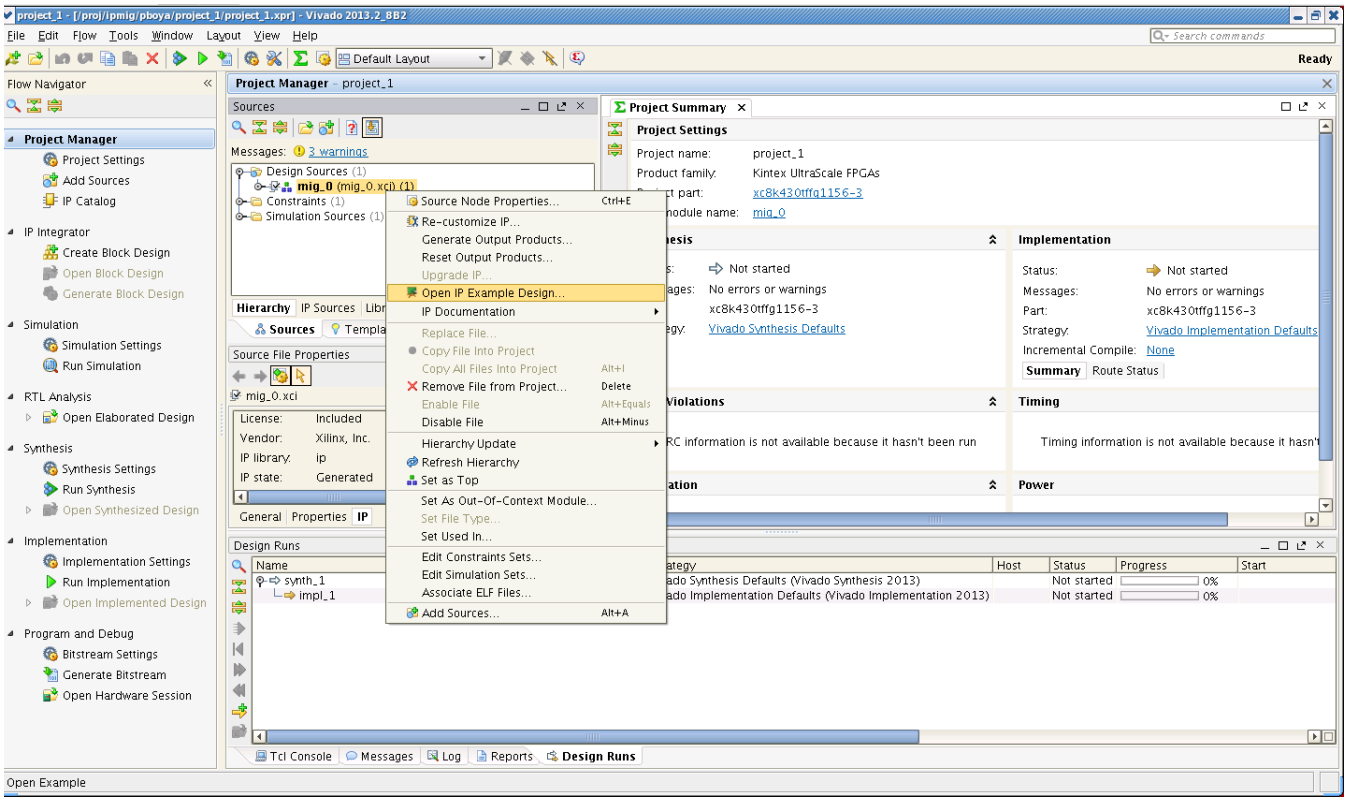

Figure 9-1: **Open IP Example Design**

This option creates a new Vivado project. Upon selecting the menu, a dialog box to enter the directory information for the new design project opens.

Select a directory, or use the defaults, and click **OK**. This launches a new Vivado with all of the example design files and a copy of the IP. This project has `example_top` as the Implementation top directory and `sim_tb_top` as the Simulation top directory, as shown in Figure 9-2.

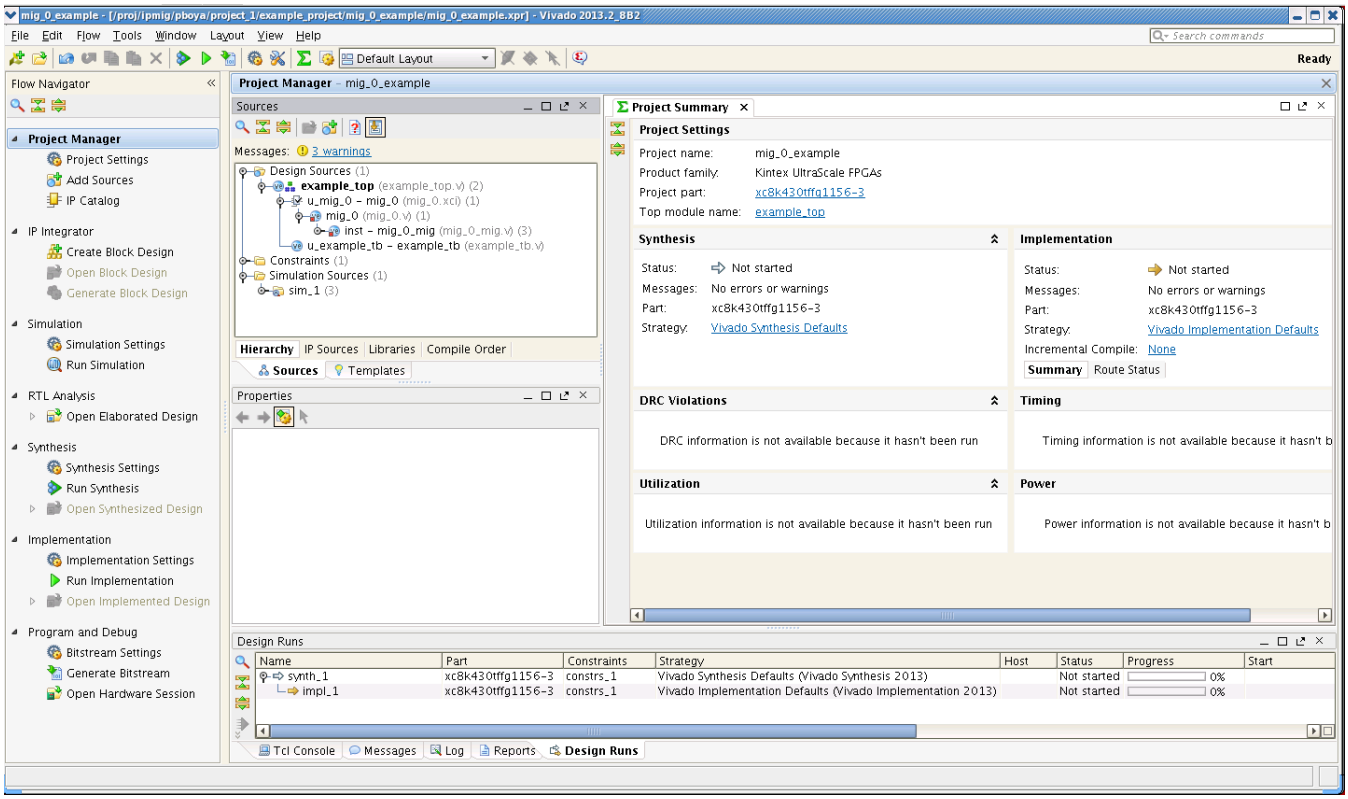

Figure 9-2: **Example Design Project**

Simulating the Example Design (Designs with Standard User Interface)

The example design provides a synthesizable test bench to generate a fixed simple data pattern to the Memory Controller. This test bench consists of an IP wrapper and an `example_tb` that generates 10 writes and 10 reads. Memory model files are not generated when designs are generated from the IP. You need to download memory models from the Micron® website.

IMPORTANT: *Xilinx `UNISIMS_VER` and `SECUREIP` library must be mapped into the simulator.*

To run the simulation, go to this directory:

```
<project_dir>/example_project/<Component_Name>example/
<Component_Name>_example.srcs/sim_1/imports/<Component_Name>/tb
```

If the MIG design is generated with the Component Name entered in the Vivado IDE as `mig_0`, the simulation directory path is the following:

```
<project_dir>/example_project/mig_0_example/mig_0_example.srcs/
sim_1/imports/mig_0/tb
```

Copy the memory models in the above directory and see the `readme.txt` file located in the folder for running simulations.

The Questa® SIM simulation tool is used for verification of MIG IP at each software release. Script file to run simulations with Questa SIM is generated in MIG generated output. MIG designs are not verified with Vivado Simulator. Other simulation tools can be used for MIG IP simulation but are not specifically verified by Xilinx.

Test Bench

This chapter contains information about the provided test bench in the Vivado® Design Suite environment.

The intent of the performance test bench is for you to obtain an estimate on the efficiency for a given traffic pattern with the MIG controller. The test bench passes your supplied commands and address to the Memory Controller and measures the efficiency for the given pattern. The efficiency is measured by the occupancy of the `dq` bus. The primary use of the test bench is for efficiency measurements so no data integrity checks are performed. Static data is written into the memory during write transactions and the same data is always read back.

The stimulus to the traffic generator is provided through a `mig_v4_2_ddr4_stimulus.txt` file. The stimulus consists of command, address, and command repetition count. Each line in the stimulus file represents one stimulus (command repetition, address, and command). Multiple stimuli can be provided in a stimulus file and each stimulus is separated by the new line.

Table 10-1: **Modules for Performance Traffic Generator**

File Name	Description
mig_v4_2_ddr4_traffic_generator.sv	This file has the traffic generator code for sending out the traffic for DDR4 and also for the calculation of bus utilized.
mig_v4_2_ddr4_stimulus.txt	These files have the stimulus with Writes, Reads, and NOPs for DDR4 for the calculation of bus utilization.
mig_v4_2_ddr3_traffic_generator.sv	This file has the traffic generator code for sending out the traffic for DDR3 and also for the calculation of bus utilized.
mig_v4_2_ddr3_stimulus.txt	These files have the stimulus with Writes, Reads, and NOPs for DDR3 for the calculation of bus utilization.

Stimulus Pattern

Each stimulus pattern is 48 bits and the format is described in Table 10-2 and Table 10-3.

Table 10-2: **Stimulus Command Pattern**

Command Repeat[47:40]	Address [39:4]	Command[3:0]

Table 10-3: **Stimulus Pattern Description**

Signal	Description
Command[3:0]	This corresponds to the WRITE/READ/NOP command that is sent to the user interface.
Address[35:0]	This corresponds to the address to the user interface.
Command Repeat[7:0]	This corresponds to the repetition count of the command. Up to 128 repetitions can be made for a command. In the burst length of eight mode, 128 transactions fill up the page in the memory.

Command Encoding (Command[3:0])

Table 10-4: **Command Description**

Command	Code	Description
WRITE	0	This corresponds to the Write operation that needs to be performed.
READ	1	This corresponds to the Read operation that needs to be performed.
NOP	7	This corresponds to the idle situation for the bus.

Address Encoding (Address[35:0])

Address is encoded in the stimulus as per Figure 10-1 to Figure 10-6. All the address fields need to be entered in the hexadecimal format. All the address fields are the width that is divisible by four to enter in the hexadecimal format. The test bench only sends the required bits of an address field to the Memory Controller.

For example, an eight bank configuration only bank Bits[2:0] is sent to the Memory Controller and the remaining bits are ignored. The extra bits for an address field are provided for you to enter the address in a hexadecimal format. You must confirm the value entered corresponds to the width of a given configuration.

Table 10-5: **Address Encoded**

Rank[3:0]	Bank[3:0]	Row[15:0]	Column[11:0]

- **Column Address (Column[11:0])** – Column Address in the stimulus is provided maximum of 12 bits, but you need to address this based on the column width parameter set in your design.

- **Row Address (Row[15:0])** – Row address in the stimulus is provided maximum of 16 bits, but you need to address this based on the row width parameter set in your design.
- **Bank Address (Bank[3:0])** – Bank address in the stimulus is provided maximum of four bits, but you need to address this based on the bank width parameter set in your design.

  Note: In case of DDR4, use the 2-bit LSB for Bank Address and two bits of MSB for Bank Groups.
- **Rank Address (Rank[3:0])** – Rank address in the stimulus is provided maximum of four bits, but you need to address this based on the rank width parameter set in your design.

The address is assembled based on the top-level MEM_ADDR_ORDER parameter and sent to the user interface.

Command Repeat (Command Repeat[7:0])

The command repetition count is the number of time the respective command is repeated at the user interface. The address for each repetition is incremented by 8. The maximum repetition count is 128. The test bench does not check for the column boundary and it wraps around if the maximum column limit is reached during the increments. The 128 commands fill up the page. For any column address other than zero, the repetition count of 128 ends up crossing the column boundary and wrapping around to the start of the column address.

Bus Utilization

The bus utilization is calculated at the user interface taking total number of Reads and Writes into consideration and the following equation is used:

$$bw\_cumulative = \frac{((rd\_command\_cnt + wr\_command\_cnt) \times (BURST\_LEN / 2) \times 100)}{((end\_of\_stimulus - calib\_done) / tCK);} \quad \textit{Equation 10-1}$$

- BL8 takes four memory clock cycles.
- end_of_stimulus is the time when all the commands are done.
- calib_done is the time when the calibration is done.

Example Patterns

These examples are based on the MEM_ADDR_ORDER set to BANK_ROW_COLUMN.

Single Read Pattern

00_0_2_000F_00A_1 – This pattern is a single read from 10th column, 15th row, and second bank.

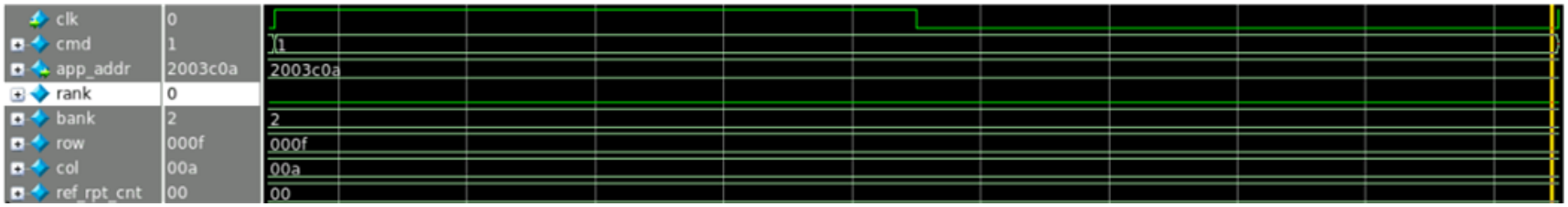

Figure 10-1: **Single Read Pattern**

Single Write Pattern

00_0_1_0040_010_0 – This pattern is a single write to the 32nd column, 128th row, and first bank.

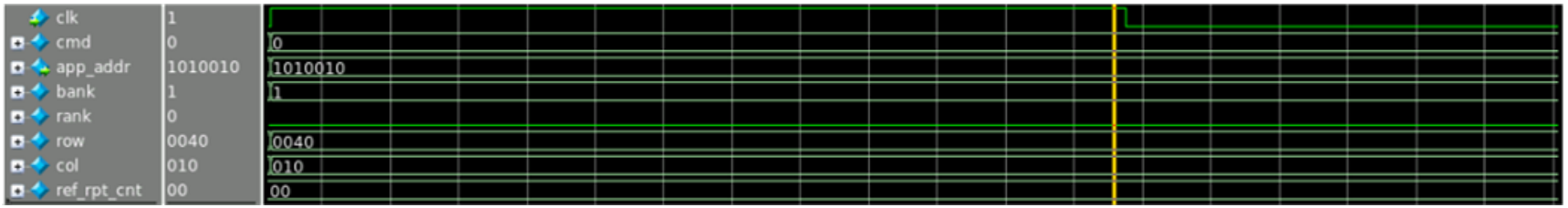

Figure 10-2: **Single Write Pattern**

Single Write and Read to Same Address

00_0_2_000F_00A_0 – This pattern is a single write to 10th column, 15th row, and second bank.

00_0_2_000F_00A_1 – This pattern is a single read from 10th column, 15th row, and second bank.

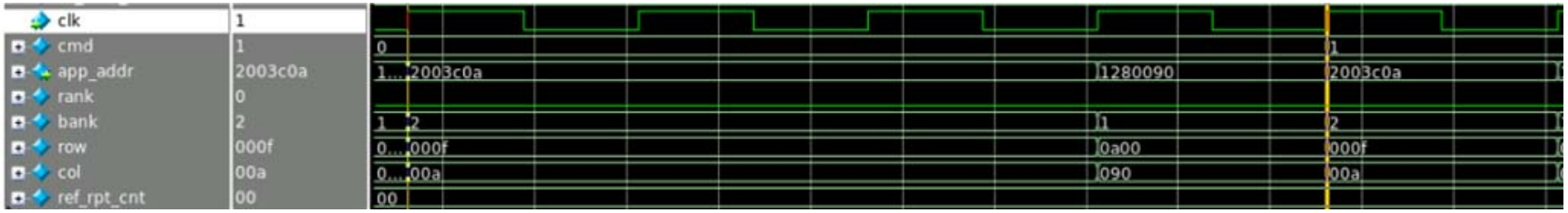

Figure 10-3: **Single Write and Read to Same Address**

Multiple Writes and Reads with Same Address

0A_0_0_0010_000_0 – This corresponds to 10 writes with address starting from 0 to 80 which can be seen in the column.

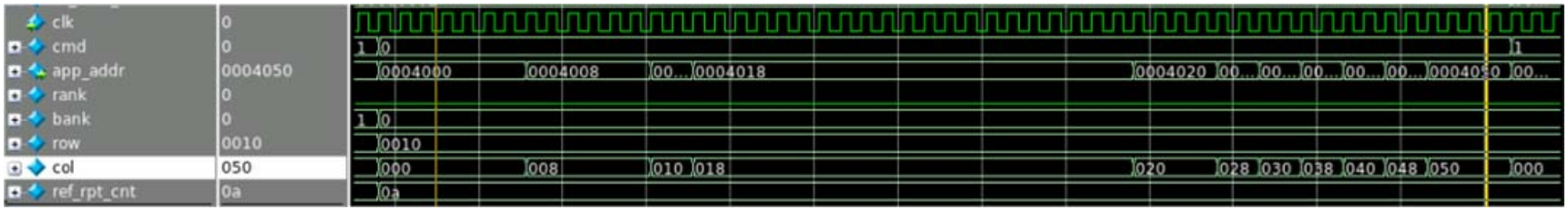

Figure 10-4: **Multiple Writes with Same Address**

0A_0_0_0010_000_1 – This corresponds to 10 reads with address starting from 0 to 80 which can be seen in the column.

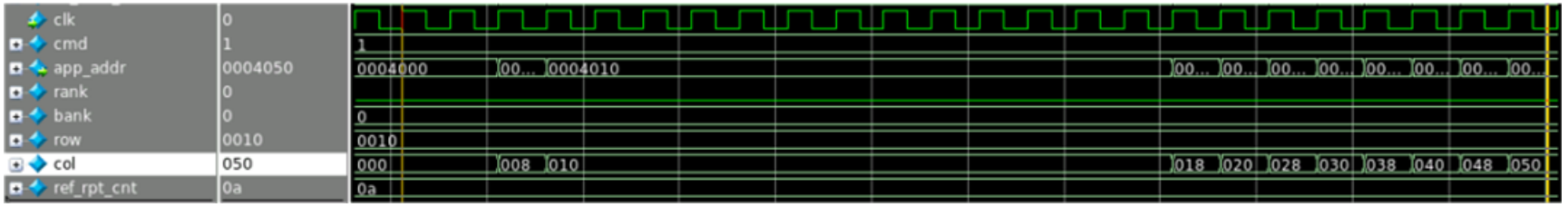

Figure 10-5: **Multiple Reads with Same Address**

Page Wrap During Writes

0A_0_2_000F_3F8_0 – This corresponds to 10 writes with column address wrapped to the starting of the page after one write.

Figure 10-6: **Page Wrap During Writes**

Simulating the Performance Traffic Generator

1. Map Xilinx `UNISIMS_VER` and `SECUREIP` library into the simulator.
2. Copy the memory models in the following directory:

   `<project_dir>/example_project/<Component_Name>example/<Component_Name>_example.srcs/sim_1/imports/<Component_Name>/tb`

3. Modify the `mig_v4_2_ddr4_stimulus.txt` for DDR4 and `mig_v4_2_ddr3_stimulus.txt` for DDR3 present in the following directory with the stimulus that wanted the percentage of bus utilization:

   `<project_dir>/example_project/<Component_Name>example/<Component_Name>_example.srcs/sim_1/imports/<Component_Name>/tb`

4. Run the `performance_sim.do` file.
5. For Questa® SIM, run the following command `vsim -do performance_sim.do`.
6. After the run in the `tb` directory, `mig_v4_2_ddr4_band_width_cal.txt` for DDR4 and `mig_v4_2_ddr3_band_width_cal.txt` for DDR3 are found with all of the output about bus percentage utilization metrics.

SECTION III: RLDRAM 3

Product Specification

Core Architecture

Designing with the Core

Customizing and Generating the Core

Constraining the Core

Simulation

Synthesis and Implementation

Example Design

Test Bench

Product Specification

Standards

For more information on UltraScale™ architecture documents, see References, page 129.

Performance

Maximum Frequencies

For more information on the maximum frequencies, see *Kintex UltraScale Architecture Data Sheet, DC and AC Switching Characteristics* (DS892) [Ref 2].

Resource Utilization

Kintex UltraScale Devices

Table 11-1 provides approximate resource counts on Kintex® UltraScale™ devices.

Table 11-1: **Device Utilization – Kintex UltraScale FPGAs**

Parameter Values	Device Resources						
Interface Width	Slices	LUTs	FFs	CLB LUTs	CLB Registers	RAMB36E2	PLLE3_ADV
72	413	732	722	10,489	12,534	16	3

Resources required for the UltraScale architecture-based FPGAs MIS core have been estimated for the Kintex UltraScale devices (Table 11-1). These values were generated using Vivado® IP catalog. They are derived from post-synthesis reports, and might change during implementation.

Port Descriptions

There are three port categories at the top-level of the memory interface core called the "user design."

- The first category are the memory interface signals that directly interfaces with the RLDRAM. These are defined by the RLDRAM 3 specification.
- The second category are the application interface signals which is the "user interface." These are described in the Protocol Description in Chapter 13.
- The third category includes other signals necessary for proper operation of the core.

These include the clocks, reset, and status signals from the core. The clocking and reset signals are described in their respective sections.

The active-High `init_calib_complete` signal indicates that the initialization and calibration are complete and that the interface is now ready to accept commands for the interface.

Core Architecture

This chapter describes the UltraScale™ architecture-based FPGAs Memory Interface Solutions core with an overview of the modules and interfaces.

Overview

Figure 12-1 shows a high-level block diagram of the RLDRAM 3 memory interface solution. This figure shows both the internal FPGA connections to the user interface for initiating read and write commands, and the external interface to the memory device.

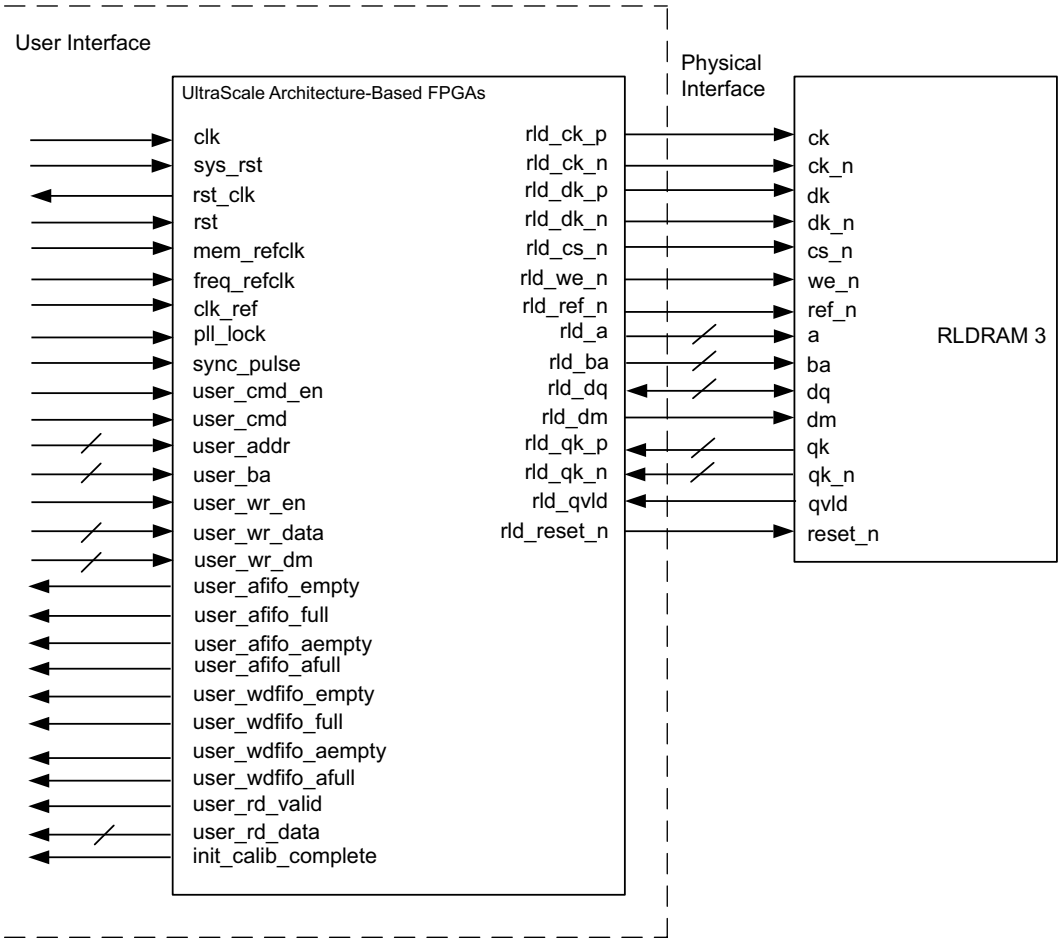

Figure 12-1: **High-Level Block Diagram of RLDRAM 3 Interface Solution**

Figure 12-2 shows the UltraScale architecture-based FPGAs Memory Interface Solutions diagram.

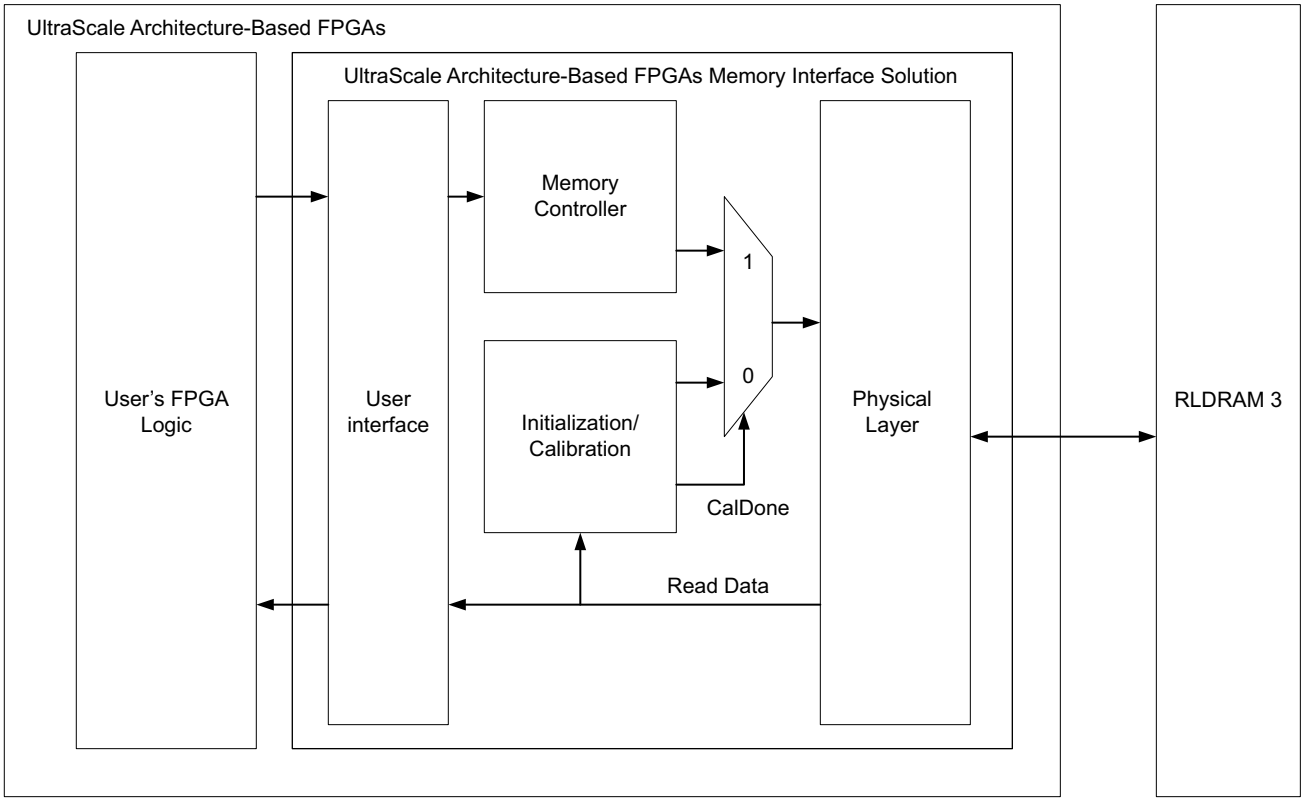

Figure 12-2: **UltraScale Architecture-Based FPGAs Memory Interface Solution Core**

The user interface uses a simple protocol based entirely on SDR signals to make read and write requests. See User Interface in Chapter 13 for more details describing this protocol.

The Memory Controller takes commands from the user interface and adheres to the protocol requirements of the RLDRAM 3 device. See Memory Controller for more details.

The physical interface generates the proper timing relationships and DDR signaling to communicate with the external memory device, while conforming to the RLDRAM 3 protocol and timing requirements. See Physical Interface in Chapter 13 for more details.

Memory Controller

The Memory Controller (MC) enforces the RLDRAM 3 access requirements and interfaces with the PHY. The controller processes commands in order, so the order of commands presented to the controller is the order in which they are presented to the memory device.

The MC first receives commands from the user interface and determines if the command can be processed immediately or needs to wait. When all requirements are met, the command is placed on the PHY interface. For a write command, the controller generates a signal for the user interface to provide the write data to the PHY. This signal is generated based on the memory configuration to ensure the proper command-to-data relationship. Auto-refresh commands are inserted into the command flow by the controller to meet the memory device refresh requirements.

For CIO devices, the data bus is shared for read and write data. Switching from read commands to write commands and vice versa introduces gaps in the command stream due to switching the bus. For better throughput, changes in the command bus should be minimized when possible.

CMD_PER_CLK is a top-level parameter used to determine how many memory commands are provided to the controller per FPGA logic clock cycle. It depends on nCK_PER_CLK and the burst length. For example if nCK_PER_CLK = 4, the CMD_PER_CLK is set to 1 for burst length = 8 and CMD_PER_CLK is set to 2 for burst length = 4.

The controller remains in the CTL_IDLE state until calibration completes. When the calibration `done` signal is asserted and a command request is received, the state machine transitions to CTL_LOAD_CMD2 through the CTL_LOAD_CMD1 state (essentially a pipeline state).

For a single command request the controller state machine transitions from CTL_LOAD_CMD2 to CTL_PROC_LAST_CMD and back to CTL_IDLE.

For multiple commands to the same RLDRAM 3 bank and CMD_PER_CLK = 1, the state machine transitions from CTL_LOAD_CMD2 → CTL_PROC_CMD → CTL_PROC_CMD1 → CTL_PROC_LAST_CMD → CTL_IDLE.

For multiple commands to the same RLDRAM 3 bank and CMD_PER_CLK > 1, the state machine transitions from CTL_LOAD_CMD2 → CTL_PROC_CMD → CTL_PROC_CMD1 → CTL_PROC_LAST_CMD → CTL_PROC_LAST_CMD1 → CTL_IDLE.

For multiple commands to different RLDRAM 3 banks and CMD_PER_CLK = 1, the state machine transitions from CTL_LOAD_CMD2 → CTL_PROC_CMD → CTL_PROC_LAST_CMD → CTL_IDLE.

For multiple commands to different RLDRAM 3 banks and CMD_PER_CLK > 1, the state machine transitions from CTL_LOAD_CMD2 → CTL_PROC_CMD → CTL_PROC_LAST_CMD → CTL_PROC_LAST_CMD1 → CTL_IDLE.

Figure 12-3 shows the state machine logic for the controller.

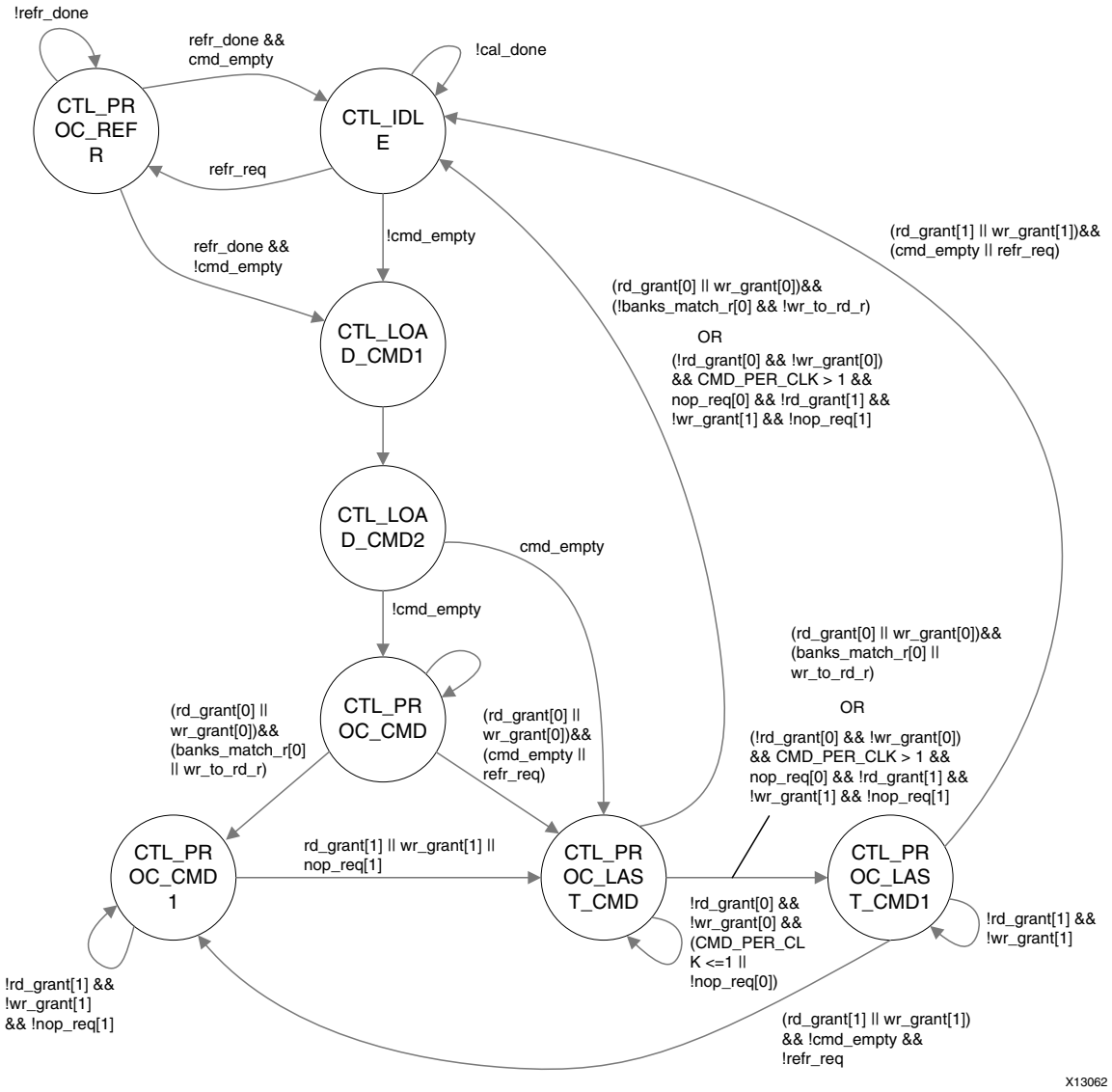

Figure 12-3: **Controller State Machine Logic (CMD_PER_CLK == 1 or 2)**

PHY

PHY is considered the low-level physical interface to an external RLDRAM 3 device as well as all calibration logic for ensuring reliable operation of the physical interface itself. PHY generates the signal timing and sequencing required to interface to the memory device.

PHY contains the following features:

- Clock/address/control-generation logics
- Write and read datapaths
- Logic for initializing the SDRAM after power-up

In addition, PHY contains calibration logic to perform timing training of the read and write datapaths to account for system static and dynamic delays.

Overall PHY Architecture

The UltraScale architecture PHY is composed of dedicated blocks and soft calibration logic. The dedicated blocks are structured adjacent to one another with back-to-back interconnects to minimize the clock and datapath routing necessary to build high performance physical layers.

The Memory Controller and calibration logic communicate with this dedicated PHY in the slow frequency clock domain, which is either divided by 4 or divided by 2. This depends on the RLDRAM 3 memory clock. A more detailed block diagram of the PHY design is shown in Figure 12-4.

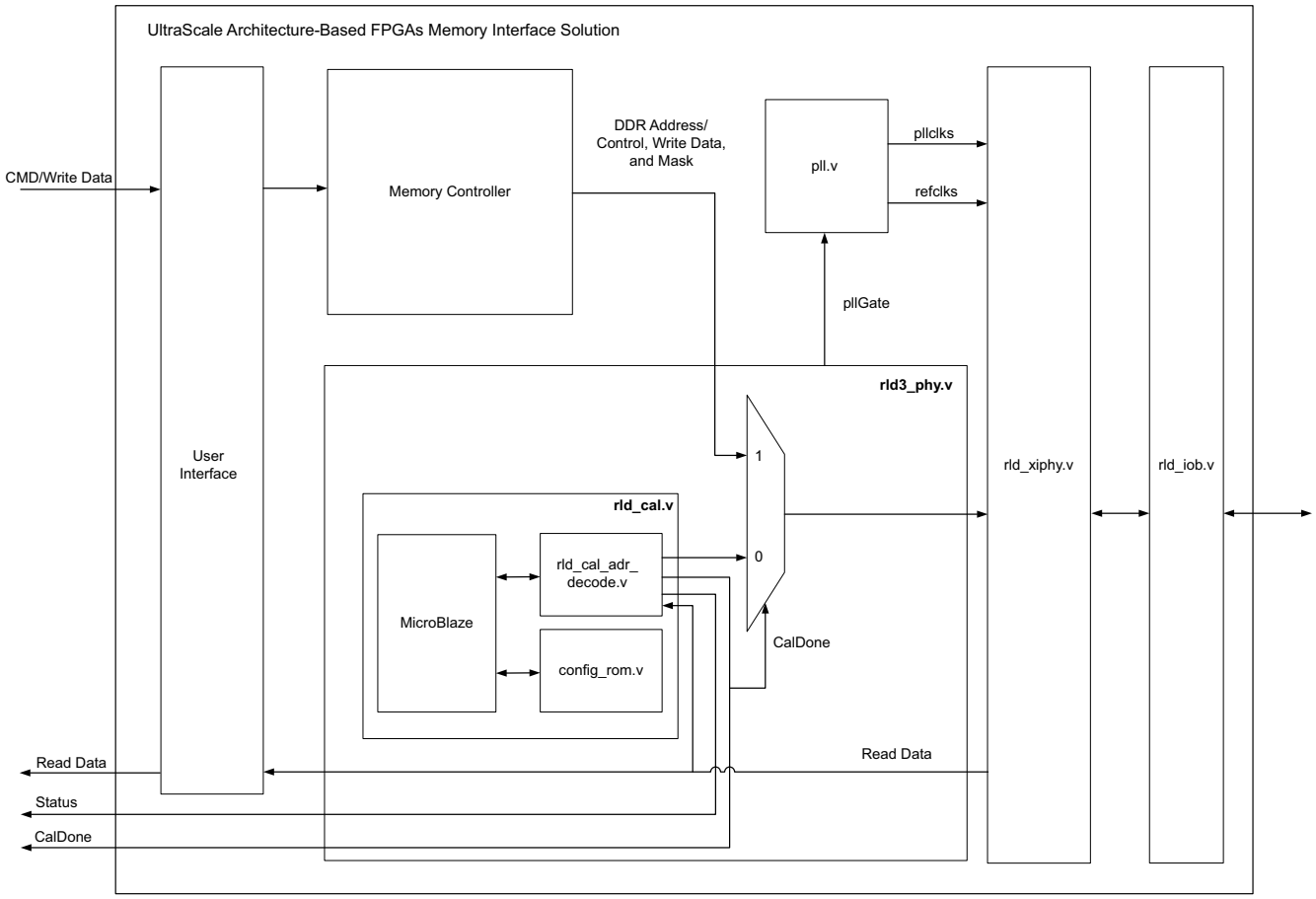

Figure 12-4: **PHY Block Diagram**

The MC is designed to separate out the command processing from the low-level PHY requirements to ensure a clean separation between the controller and physical layer. The command processing can be replaced with custom logic if desired, while the logic for interacting with the PHY stays the same and can still be used by the calibration logic.

Table 12-1: **PHY Modules**

Module Name	Description
rld3_phy.v	Contains infrastructure (pll.v), rld_cal.v, rld_xiphy.v, and MUXes between the calibration and the Memory Controller.
rld_cal.v	Contains the MicroBlaze processing system and associated logic.
rld_cal_adr_decode.v	FPGA logic interface for the MicroBlaze processor.
config_rom.v	Configuration storage for calibration options.
microblaze	MicroBlaze processor
rld_iob.v	Instantiates all byte IOB modules
rld_iob_byte.v	Generates the I/O buffers for all the signals in a given byte lane.

The PHY architecture encompasses all of the logic contained in `rld_xiphy.v`. The PHY contains wrappers around dedicated hard blocks to build up the memory interface from smaller components. A byte lane contains all of the clocks, resets, and datapaths for a given subset of I/O. Multiple byte lanes are grouped together, along with dedicated clocking resources, to make up a single bank memory interface. For more information on the hard silicon physical layer architecture, see the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3].

The memory initialization and calibration are implemented in C programming on a small soft core processor. The MicroBlaze™ Controller System (MCS) is configured with an I/O Module, MicroBlaze Debug Module (MDM), and block RAM. The `rld_cal_adr_decode.v` module provides the interface for the processor to the rest of the system and implements helper logic. The `config_rom.v` module stores settings that control the operation of initialization and calibration, providing run time options that can be adjusted without having to recompile the source code.

The address unit connects the MCS to the local register set and the PHY by performing address decode and control translation on the I/O module bus from spaces in the memory map and MUXing return data (`rld_cal_adr_decode.v`). In addition, it provides address translation (also known as "mapping") from a logical conceptualization of the DRAM interface to the appropriate pinout-dependent location of the delay control in the PHY address space.

Although the calibration architecture presents a simple and organized address map for manipulating the delay elements for individual data, control and command bits, there is flexibility in how those I/O pins are placed. For a given I/O placement, the path to the FPGA logic is locked to a given pin. To enable a single binary software file to work with any memory interface pinout, a translation block converts the simplified RIU addressing into the pinout-specific RIU address for the target design. The specific address translation is written by MIG after a pinout is selected. The code shows an example of the RTL structure that supports this.

```
Casez(io_address)// MicroBlaze I/O module address
  // … static address decoding skipped
  //=======================================//
  //===========DQ ODELAYS==================//
  //=======================================//
  //Byte0
  28'h0004100: begin //dq2
    riu_addr_cal =  /* MIG Generated */ 6'hd;
    riu_nibble =    /* MIG Generated */ 'h0;
  end
  // … additional dynamic addressing follows
```

In this example, `DQ0` is pinned out on Bit[0] of nibble 0 (nibble 0 according to instantiation order). The RIU address for the ODELAY for Bit[0] is 0x0D (for more details on the RIU address map, see the RIU specification). When `DQ0` is addressed — indicated by address 0x000_4100), this snippet of code is active. It enables nibble 0 (decoded to one-hot downstream) and forwards the address 0x0D to the RIU address bus.

The MicroBlaze I/O module interface updates at a maximum rate of once every three clock cycles, which is not always fast enough for implementing all of the functions required in calibration. A helper circuit implemented in `rld_cal_adr_decode.v` is required to obtain commands from the registers and translate at least a portion into single-cycle accuracy for submission to the PHY. In addition, it supports command repetition to enable back-to-back read transactions and read data comparison.

Memory Initialization and Calibration Sequence

After deassertion of the system reset, PHY performs some required internal calibration steps first.

1. The built-in self-check of the PHY (BISC) is run.
2. BISC is used in the PHY to compute internal skews for use in voltage and temperature tracking after calibration is completed.
3. After BISC is completed, calibration logic performs the required power-on initialization sequence for the memory. This is followed by several stages of timing calibration for the write and read datapaths.
4. After calibration is completed, PHY calculates internal offsets to be used in voltage and temperature tracking.
5. PHY indicates calibration is finished and the controller begins issuing commands to the memory.

Figure 12-5 shows the overall flow of memory initialization and the different stages of calibration.

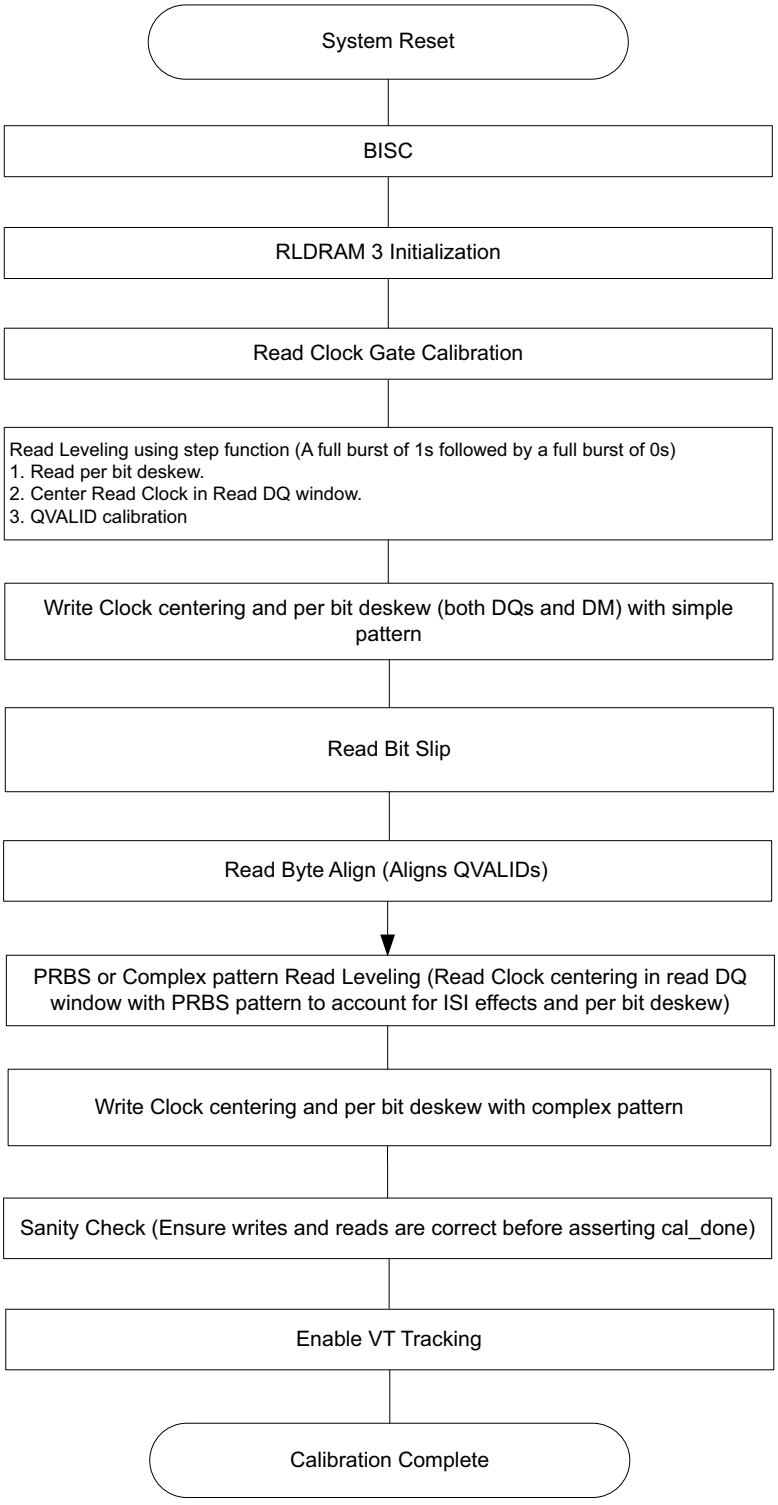

Figure 12-5: **PHY Overall Initialization and Calibration Sequence**

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Clocking

The memory interface requires one PLL in each bank that is occupied by the interface. There are two PLLs per bank. If a bank is shared by two interfaces, both PLLs in that bank are used.

The memory interface requires a high quality reference clock. This clock should come from a differential pair on the same column of the FPGA that the memory interface occupies. The memory interface PLLs generate the appropriate frequencies and phases necessary for proper operation. An output clock is provided for the FPGA logic which includes the controller and the interface logic. This output clock is 1/4 of the memory interface clock in the 4:1 memory clock to FPGA logic clock mode.

Resets

An asynchronous reset input is provided. This active-High reset must assert for a minimum of 20 cycles of the controller clock.

PCB Guidelines for RLDRAM 3

Overview

Contact Xilinx for more information about PCB guidelines,

Pin and Bank Rules

RLDRAM 3 Pin Rules

The rules are for single rank memory interfaces.

- Address/control means `cs_n`, `ref_n`, `we_n`, `ba`, `ck`, `reset_n`, and `a`.
- All groups such as, Data, Address/Control, and System clock interfaces must be selected in a single column.
- Pins in a byte lane are numbered N0 to N12.
- Byte lanes in a bank are designed by T0, T1, T2, or T3. Nibbles within a byte lane are distinguished by a “U” or “L” designator added to the byte lane designator (T0, T1, T2, or T3). Thus they are T0L, T0U, T1L, T1U, T2L, T2U, T3L, and T3U.

Note: There are two PLLs per bank and a controller uses one PLL in every bank that is being used by the interface.

1. Read Clock (`qk/qk_n`), Write Clock (`dk/dk_n`), `dq`, `qvld`, and `dm`.
   a. Read Clock pairs (`qkx_p/n`) must be placed on N0 and N1 pins in the lower nibble of a byte lane designated with "L." `dq` associated with a `qk/qk_n` pair must be in same byte lane on any of the other pins except pin N12. `dq` and `dm` bits must be placed on pins N2 to N11 in a byte lane.
   b. One `dm` pin is associated with nine bits in x18 devices or with 18 bits in x36 devices. It must be placed in its associated `dq` byte lanes as listed:
      - For x18 part, `dm[0]` must be allocated in `dq[8:0]` allocated byte group and `dm[1]` must be allocated in `dq[17:9]`.
      - For x36 part, `dm[0]` must be allocated in `dq[8:0]` or `dq[26:18]` allocated byte lane. Similarly `dm[1]` must be allocated in `dq[17:9]` or `dq[35:27]` allocated byte group.
   c. `dk/dk_n` must be allocated to any P-N pair. In case of overflow where only write clock pairs need to be placed in a byte lane, place it on pin pairs other than 0/1 and 6/7.

Note: Note that pin 12 is not part of a pin pair and must not be used for differential clocks.

d. `qvld` signal must be placed on pins N2 to N12 but first priority must be given for `dq` and `dm` allocation. `qvld0` signal must be placed on pin N11 of byte lane (if available, `dm` is disabled), pin N12 of byte lane of the `qk0/qk0_n` data byte lane and `qvld1` signal must be placed on pin N11 of byte lane (if available), or pin N12 of byte lane of the `qk2/qk2_n` data byte lane

2. Byte lanes are configured as either data or address/control.
   a. Pin N12 can be used for address/control in a data byte lane.
   b. No data signals (`qvalid, dq, dm`) can be placed in an address/control byte lane.
3. Address/control can be on any of the 13 pins in the address/control byte lanes. Address/control must be contained within the same bank. Address/control must be in the centermost bank.
4. There is one `vr` pin per bank and DCI is required. DCI cascade is not permitted. All rules for the DCI in the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3] must be followed.
5. `ck` must be on the PN pair in the Address/Control byte lane.
6. `reset_n` can be on any pin as long as FPGA logic timing is met and I/O standard can be accommodated for the chosen bank (LVCMOS12).
7. Banks can be shared between two controllers.
   a. Each byte lane is dedicated to a specific controller (except for `reset_n`).
   b. Byte lanes from one controller cannot be placed inside the other. For example, with controllers A and B, “AABB” is allowed, while “ABAB” is not.
8. All I/O banks used by the memory interface must be in the same column.
9. Maximum height of interface is three contiguous banks for 72-bit wide interface.
10. Bank skipping is not allowed.
11. The input clock for the master PLL in the interface must come from the a clock capable pair in the I/O column used for the memory interface.
12. There are dedicated V_{REF} pins (not included in the rules above). If an external V_{REF} is not used, the V_{REF} pins should be pulled to ground by a 500Ω resistor. For more information, see the *UltraScale™ Architecture-Based FPGAs SelectIO™ Resources User Guide* (UG571) [Ref 3]. These pins must be connected appropriately for the standard in use.
13. The interface must be contained within the same I/O bank type (High Range or High Performance). Mixing bank types is not permitted with the exceptions of the `reset_n` in step 6 above and the input clock mentioned in step 11 above.

RLDRAM 3 Pinout Examples

Table 13-1 shows an example of a 18-bit RLDRAM 3 interface contained within one bank. This example is for a component interface using one x18 RLDRAM3 component with Address Multiplexing.

Table 13-1: **18-Bit RLDRAM 3 Interface Contained in One Bank**

Bank	Signal Name	Byte Group	I/O Type	Special Designation
1	qvld0	T3U_12	–	–
1	dq8	T3U_11	N	–
1	dq7	T3U_10	P	–
1	dq6	T3U_9	N	–
1	dq5	T3U_8	P	–
1	dq4	T3U_7	N	DBC-N
1	dq3	T3U_6	P	DBC-P
1	dq2	T3L_5	N	–
1	dq1	T3L_4	P	–
1	dq0	T3L_3	N	–
1	dm0	T3L_2	P	–
1	qk0_n	T3L_1	N	DBC-N
1	qk0_p	T3L_0	P	DBC-P
1	reset_n	T2U_12	–	–
1	we#	T2U_11	N	–
1	a18	T2U_10	P	–
1	a17	T2U_9	N	–
1	a14	T2U_8	P	–
1	a13	T2U_7	N	QBC-N
1	a10	T2U_6	P	QBC-P
1	a9	T2L_5	N	–
1	a8	T2L_4	P	–
1	a5	T2L_3	N	–
1	a4	T2L_2	P	–
1	a3	T2L_1	N	QBC-N
1	a0	T2L_0	P	QBC-P
1	–	T1U_12	–	–
1	ba3	T1U_11	N	–

Table 13-1: **18-Bit RLDRAM 3 Interface Contained in One Bank *(Cont'd)***

Bank	Signal Name	Byte Group	I/O Type	Special Designation
1	ba2	T1U_10	P	–
1	ba1	T1U_9	N	–
1	ba0	T1U_8	P	–
1	dk1_n	T1U_7	N	QBC-N
1	dk1_p	T1U_6	P	QBC-P
1	dk0_n	T1L_5	N	–
1	dk0_p	T1L_4	P	–
1	ck_n	T1L_3	N	–
1	ck_p	T1L_2	P	–
1	ref_n	T1L_1	N	QBC-N
1	cs_n	T1L_0	P	QBC-P
1	vr	T0U_12	–	–
1	dq17	T0U_11	N	–
1	dq16	T0U_10	P	–
1	dq15	T0U_9	N	–
1	dq14	T0U_8	P	–
1	dq13	T0U_7	N	DBC-N
1	dq12	T0U_6	P	DBC-P
1	dq11	T0L_5	N	–
1	dq10	T0L_4	P	–
1	dq9	T0L_3	N	–
1	dm1	T0L_2	P	–
1	qk1_n	T0L_1	N	DBC-N
1	qk1_p	T0L_0	P	DBC-P

Protocol Description

This core has the following interfaces:

- Memory Interface
- User Interface
- Physical Interface

Memory Interface

The RLDRAM 3 memory interface solution is customizable to support several configurations. The specific configuration is defined by Verilog parameters in the top-level of the core.

User Interface

The user interface connects to an FPGA user design to the RLDRAM 3 memory solutions core to simplify interactions between you and the external memory device.

Command Request Signals

The user interface provides a set of signals used to issue a read or write command to the memory device. These signals are summarized in Table 13-2.

Table 13-2: **User Interface Request Signals**

Signal	Direction	Description
user_cmd_en	Input	Command Enable. This signal issues a read or write request and indicates that the corresponding command signals are valid.
user_cmd[2 × CMD_PER_CLK – 1:0]	Input	Command. This signal issues a read, write, or NOP request. When `user_cmd_en` is asserted: 2'b00 = Write Command 2'b01 = Read Command 2'b10 = NOP 2'b11 = NOP The NOP command is useful when more than one command per clock cycle must be provided to the Memory Controller yet not all command slots are required in a given clock cycle. The Memory Controller acts on the other commands provided and ignore the NOP command. NOP is not supported when CMD_PER_CLK == 1. CMD_PER_CLK is a top-level parameter used to determine how many memory commands are provided to the controller per FPGA logic clock cycle, it depends on `nCK_PER_CLK` and the burst length (see Figure 13-1)
user_addr[CMD_PER_CLK × ADDR_WIDTH – 1:0]	Input	Command Address. This is the address to use for a command request. It is valid when `user_cmd_en` is asserted.
user_ba[CMD_PER_CLK × BANK_WIDTH – 1:0]	Input	Command Bank Address. This is the address to use for a write request. It is valid when `user_cmd_en` is asserted.
user_wr_en	Input	Write Data Enable. This signal issues the write data and data mask. It indicates that the corresponding `user_wr_*` signals are valid.
user_wr_data[2 × nCK_PER_CLK × DATA_WIDTH – 1:0]	Input	Write Data. This is the data to use for a write request and is composed of the rise and fall data concatenated together. It is valid when `user_wr_en` is asserted.
user_wr_dm[2 × nCK_PER_CLK × DM_WIDTH – 1:0]	Input	Write Data Mask. When active-High, the write data for a given selected device is masked and not written to the memory. It is valid when `user_wr_en` is asserted.
user_afifo_empty	Output	Address FIFO empty. If asserted, the command buffer is empty.
user_wdfifo_empty	Output	Write Data FIFO empty. If asserted, the write data buffer is empty.
user_afifo_full	Output	Address FIFO full. If asserted, the command buffer is full, and any writes to the FIFO are ignored until deasserted.

Table 13-2: **User Interface Request Signals *(Cont'd)***

Signal	Direction	Description
user_wdfifo_full	Output	Write Data FIFO full. If asserted, the write data buffer is full, and any writes to the FIFO are ignored until deasserted.
user_afifo_aempty	Output	Address FIFO almost empty. If asserted, the command buffer is almost empty.
user_afifo_afull	Output	Address FIFO almost full. If asserted, the command buffer is almost full.
user_wdfifo_aempty	Output	Write Data FIFO almost empty. If asserted, the write data buffer is almost empty.
user_wdfifo_afull	Output	Write Data FIFO almost full. If asserted, the Write Data buffer is almost full.
user_rd_valid[nCK_PER_CLK – 1:0]	Output	Read Valid. This signal indicates that data read back from memory is available on user_rd_data and should be sampled.
user_rd_data[2 × nCK_PER_CLK × DATA_WIDTH – 1:0]	Output	Read Data. This is the data read back from the read command.
init_calib_complete	Output	Calibration Done. This signal indicates back to the user design that read calibration is complete and requests can now take place.
mem_ck_lock_complete	Output	Memory CK Lock Done. The system should be kept in a quiet state until assertion of `mem_ck_lock_complete` to ensure minimal noise on the CK being driven to the memory.

Interfacing with the Core through the User Interface

The width of certain user interface signals is dependent on the system clock frequency and the burst length. This allows the client to send multiple commands per FPGA logic clock cycle as might be required for certain configurations.

Figure 13-1 shows the `user_cmd` signal and how it is made up of multiple commands depending on the configuration.

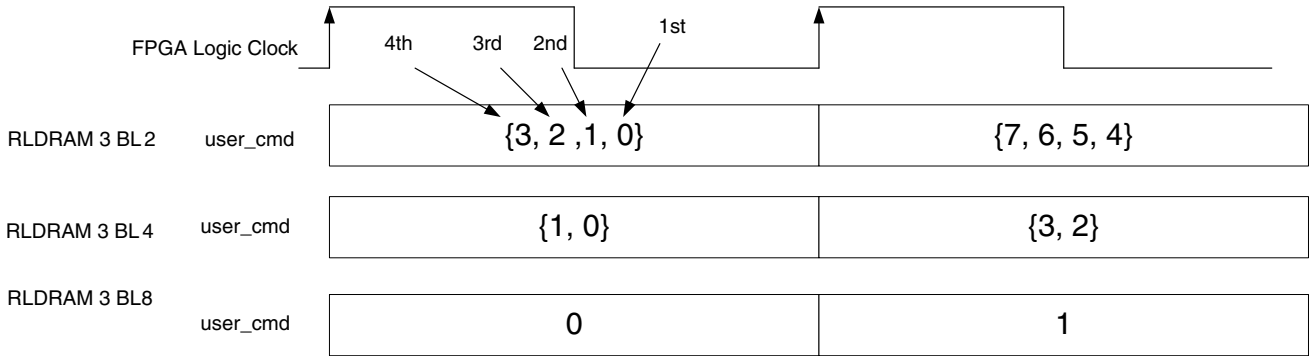

Figure 13-1: **Multiple Commands for user_cmd Signal**

The user interface protocol for the RLDRAM 3 four-word burst architecture is shown in Figure 13-2.

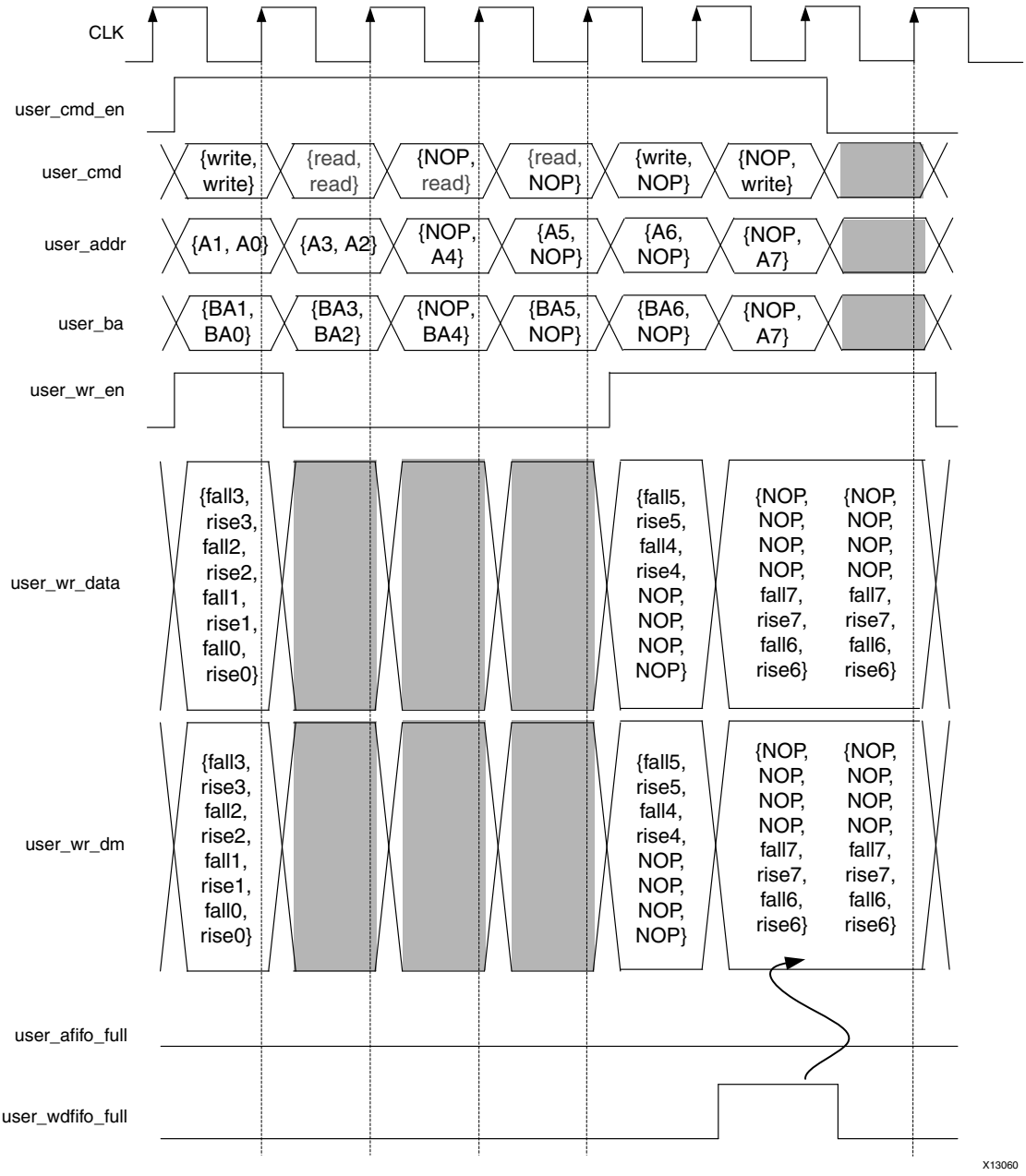

Figure 13-2: **RLDRAM 3 User Interface Protocol (Four-Word Burst Architecture)**

Before any requests can be accepted, the `ui_clk_sync_rst` signal must be deasserted Low. After the `ui_clk_sync_rst` signal is deasserted, the user interface FIFOs can accept commands and data for storage. The `init_calib_complete` signal is asserted after the memory initialization procedure and PHY calibration are complete, and the core can begin to service client requests.

A command request is issued by asserting `user_cmd_en` as a single cycle pulse. At this time, the `user_cmd`, `user_addr`, and `user_ba` signals must be valid. To issue a read request, `user_cmd` is set to 2'b01, while for a write request, `user_cmd` is set to 2'b00. For a write request, the data is to be issued in the same cycle as the command by asserting the `user_wr_en` signal High and presenting valid data on `user_wr_data` and `user_wr_dm`. The user interface protocol for the RLDRAM 3 eight-word burst architecture is shown in Figure 13-3.

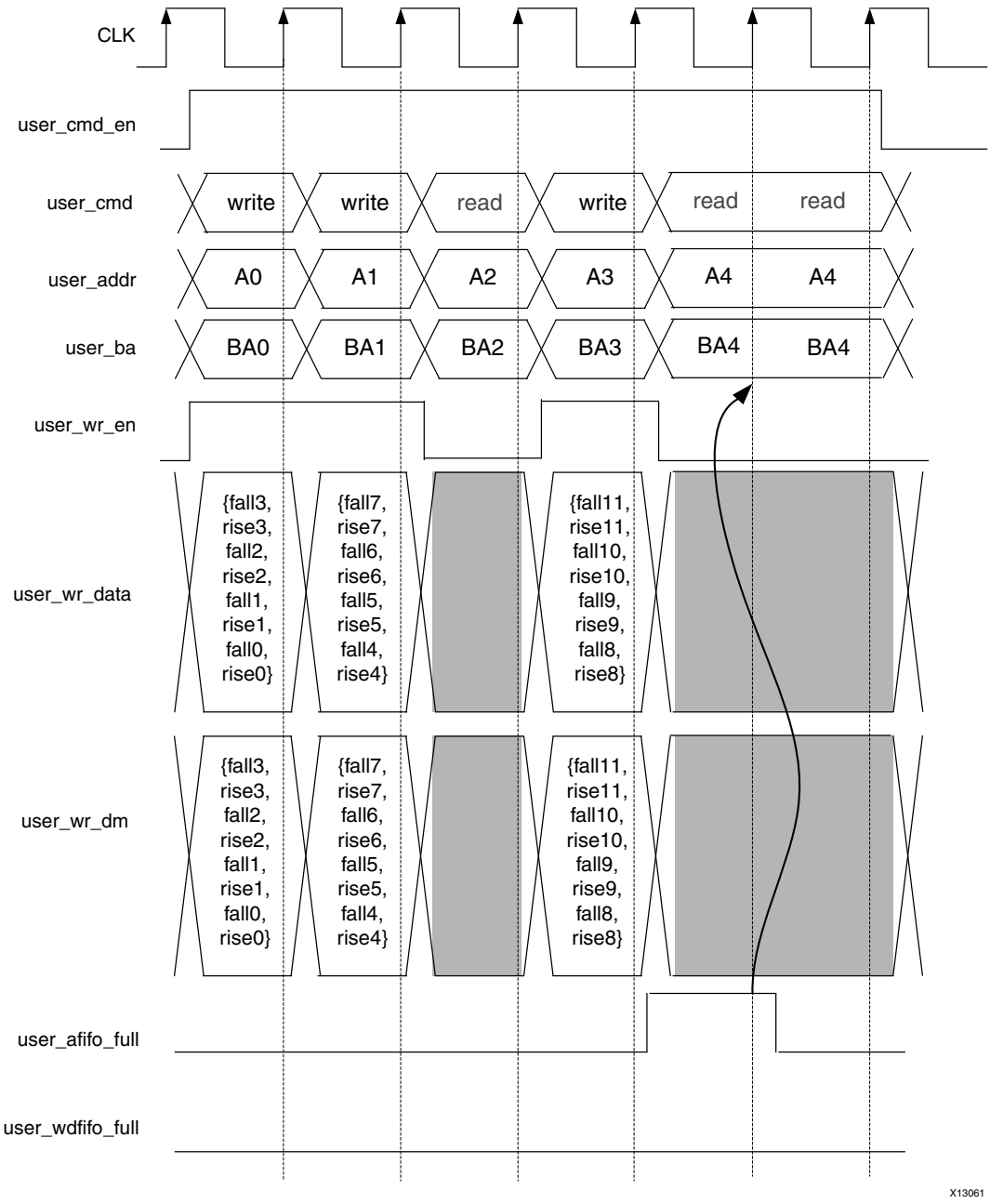

Figure 13-3: **RLDRAM 3 User Interface Protocol (Eight-Word Burst Architecture)**

When a read command is issued some time later (based on the configuration and latency of the system), the `user_rd_valid[0]` signal is asserted, indicating that `user_rd_data` is now valid, while `user_rd_valid[1]` is asserted indicating that `user_rd_data` is valid, as shown in Figure 13-4. The read data should be sampled on the same cycle that `user_rd_valid[0]` and `user_rd_valid[1]` are asserted because the core does not buffer returning data. This functionality can be added in by you, if desired.

The Memory Controller only puts commands on certain slots to the PHY such that the `user_rd_valid` signals are all asserted together and return the full width of data, but the extra `user_rd_valid` signals are provided in case of controller modifications.

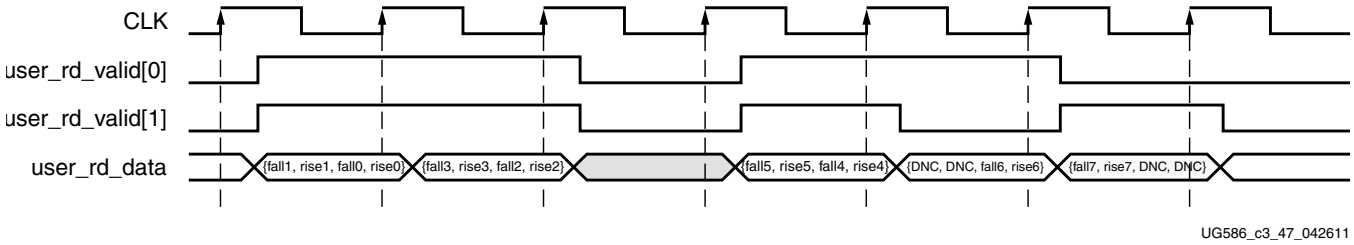

Figure 13-4: **User Interface Protocol Read Data**

Physical Interface

The physical interface is the connection from the FPGA memory interface solution to an external RLDRAM 3 device. The I/O signals for this interface are defined in Table 13-3. These signals can be directly connected to the corresponding signals on the RLDRAM 3 device.

Table 13-3: **Physical Interface Signals**

Signal	Direction	Description
rld_ck_p	Output	System Clock CK. This is the address/command clock to the memory device.
rld_ck_n	Output	System Clock CK#. This is the inverted system clock to the memory device.
rld_dk_p	Output	Write Clock DK. This is the write clock to the memory device.
rld_dk_n	Output	Write Clock DK#. This is the inverted write clock to the memory device.
rld_a	Output	Address. This is the address supplied for memory operations.
rld_ba	Output	Bank Address. This is the bank address supplied for memory operations.
rld_cs_n	Output	Chip Select CS#. This is the active-Low chip select control signal for the memory.
rld_we_n	Output	Write Enable WE#. This is the active-Low write enable control signal for the memory.
rld_ref_n	Output	Refresh REF#. This is the active-Low refresh control signal for the memory.
rld_dm	Output	Data Mask DM. This is the active-High mask signal, driven by the FPGA to mask data that a user does not want written to the memory during a write command.
rld_dq	Input/Output	Data DQ. This is a bidirectional data port, driven by the FPGA for writes and by the memory for reads.

Table 13-3: **Physical Interface Signals *(Cont'd)***

Signal	Direction	Description
rld_qk_p	Input	Read Clock QK. This is the read clock returned from the memory edge aligned with read data on rld_dq. This clock (in conjunction with QK#) is used by the PHY to sample the read data on `rld_dq`.
rld_qk_n	Input	Read Clock QK#. This is the inverted read clock returned from the memory. This clock (in conjunction with QK) is used by the PHY to sample the read data on `rld_dq`.
rld_reset_n	Output	RLDRAM 3 reset pin. This is the active-Low reset to the RLDRAM 3 device.

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl Console.

Vivado Integrated Design Environment

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or popup menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6].

Limitations

For details on limitations, see Limitations, page 79.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Constraining the Core

There is no constraints for this IP core release.

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado® Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Example Design

There is no example design for this IP core release.

Test Bench

There is no test bench for this IP core release.

SECTION IV: APPENDICES

Debugging

Additional Resources

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the MIS, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

Documentation

This product guide is the main document associated with the MIS. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

The Solution Center specific to the MIS core is located at Xilinx MIG Solution Center.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the MIS

AR: 58435

Contacting Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Additional Resources.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Note: Access to WebCase is not available in all cases. Login to the WebCase tool to see your specific support options.

Debug Tools

There are many tools available to address MIS design issues. It is important to know which tools are useful for debugging various situations.

Vivado Lab Tools

Vivado® lab tools insert logic analyzer and virtual I/O cores directly into your design. Vivado lab tools allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx devices in hardware.

The Vivado lab tools logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 9].

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

References

These documents provide supplemental material useful with this product guide:

1. JESD79-3F, *DDR3 SDRAM Standard* and JESD79-4, *DDR4 SDRAM Standard*, JEDEC® Solid State Technology Association
2. *Kintex® UltraScale™ Architecture Data Sheet: DC and AC Switching Characteristics* (DS892)
3. *UltraScale Architecture SelectIO™ Resources User Guide* (UG571)
4. *Vivado® Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)
6. *Vivado Design Suite User Guide: Getting Started* (UG910)
7. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
8. *Vivado Design Suite User Guide: Implementation* (UG904)
9. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
10. *ISE® to Vivado Design Suite Migration Guide* (UG911)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/18/2013	4.2	Initial Xilinx release.

Notice of Disclaimer