



LPDDR4 Memory Interface Module - Lattice Radiant Software

IP Version: v1.5.0

User Guide

FPGA-IPUG-02154-1.2

December 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language [FAQ 6878](#) for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Abbreviations in This Document.....	5
1. Introduction.....	6
1.1. Quick Facts	6
1.2. Features	6
1.3. Conventions	7
1.3.1. Nomenclature.....	7
1.3.2. Signal Names	7
2. Functional Description.....	8
2.1. Overview	8
2.2. Signal Description.....	10
2.3. Attribute Summary.....	13
2.4. DDR Memory Primitives.....	14
2.4.1. Input/Output DELAY	15
2.4.2. IDDR/ODDR	16
2.4.3. ODDRX4DQS	17
2.4.4. Memory Output DDR Primitives for Tristate Output Control	17
2.4.5. OSHX4.....	18
2.4.6. DDRDLL.....	18
2.4.7. ECLKDIV	19
2.4.8. ECLKSYNC	19
2.4.9. DQSBUF_IVREF	19
2.5. Clock Synchronization Logic	20
2.6. Data Input/Output Path	22
2.6.1. Write Data access.....	23
2.6.2. Read Data access	23
2.6.3. DQS Read Training.....	24
2.7. Command/Address Path	25
2.8. Training Support.....	26
2.8.1. Command Bus Training	26
3. Core Generation, Simulation, and Validation	27
3.1. Licensing the IP.....	27
3.2. Generation and Synthesis	27
3.3. Running Functional Simulation	29
Appendix A. Resource Utilization	31
References.....	32
Technical Support Assistance	33
Revision History	34

Figures

Figure 2.1. LPDDR4 Memory Interface Application	8
Figure 2.2. DDR Memory Interface Module Block Diagram (No PLL Instance)	9
Figure 2.3. Reference Guide for the Primitive Library	14
Figure 2.4. DELAYA Block Diagram	15
Figure 2.5. DELAYB Block Diagram	15
Figure 2.6. OUTDELAYA Block Diagram	16
Figure 2.7. IDDRX2DQ Block Diagram	16
Figure 2.8. ODDRX4DQ Block Diagram	17
Figure 2.9. ODDRX4DQS Block Diagram	17
Figure 2.10. TSHX4DQ Block Diagram	17
Figure 2.11. TSHX4DQS Block Diagram	18
Figure 2.12. OSHX4 Block Diagram	18
Figure 2.13. DDRDLL Block Diagram	18
Figure 2.14. ECLKDIV Block Diagram	19
Figure 2.15. ECLKSYNC Block Diagram	19
Figure 2.16. DQSBUF_IVREF Block Diagram	19
Figure 2.17. Clock Synchronization Logic Block Diagram	20
Figure 2.18. Clock Synchronization Logic Block Diagram	21
Figure 2.19. Data Input/Output Path Block Diagram	22
Figure 2.20. Write BL16 timing diagram	23
Figure 2.21. Write BL16 timing diagram with added 2nCK latency	23
Figure 2.22. Read BL16 timing diagram (rd_clkssel_i = 4'h0)	23
Figure 2.23. Read BL16 timing diagram (rd_clkssel_i = 4'hF)	24
Figure 2.24. Command/Address Path Block Diagram	25
Figure 2.25. Command/Address Path Timing diagram	26
Figure 2.26. LPDDR4 Memory V _{REFCA} Programming Timing Diagram	26
Figure 2.27. Sending/Capture a CA Pattern Timing Diagram	26
Figure 3.1. Module/IP Block Wizard	27
Figure 3.2. Configure Block of DDR Memory Module	28
Figure 3.3. Check Generating Result	28
Figure 3.4. Simulation Wizard	29
Figure 3.5. Adding and Reordering Source	30
Figure 3.6. Simulation Waveform	30

Tables

Table 1.1. Quick Facts	6
Table 2.1. LPDDR4 Memory Interface Module Signal Description	10
Table 2.2. Attribute Table	13
Table 3.1. Generated File List	29
Table A.1. Resource Utilization	31

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
CS	Chip Select
DDR	Double Data Rate
DDRDL	DDR Delay-Locked Loop
DLL	Delay-Locked Loop
DMI	Data Mask Inversion
DQ	Data (DQ)
DQS	Data Strobe
EBR	Embedded Block RAM
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IDDR	Input Double Data Rate
LPDDR	Low-Power Double Data Rate
LPDDR4	Low-Power Double Data Rate 4
LSE	Lattice Synthesis Engine
LUT	Look-Up Table
ODDR	Output Double Data Rate
ODT	On-Die Termination
PLL	Phase-Locked Loop
RTL	Register Transfer Level
SDR	Single Data Rate

1. Introduction

The Lattice Semiconductor Low Power Double Data Rate 4 (LPDDR4) Memory Interface Module generates a module that can be used to interface to an LPDDR Memory and includes a bidirectional port and the associated clocking scheme.

1.1. Quick Facts

Table 1.1 presents a summary of LPDDR4 Memory Module.

Table 1.1. Quick Facts

IP Requirements	Supported Devices	CertusPro™-NX (LFCPNX-100, LFCPNX-50, LFCPNX-100AUTODIE), and MachXO5™-NX (LFMXO5-55T, LFMXO5-100T).
	IP Changes ¹	Refer to the LPDDR4 Memory Interface Module IP Release Notes (FPGA-RN-02103) .
Resource Utilization	Supported User Interface	LPDDR4, Native interface – see Signal Description section.
	Resources	See Table A.1 .
Design Tool Support	Lattice Implementation ²	IP Core v1.5.0 – Lattice Radiant™ software 2025.2
	Synthesis	Lattice Synthesis Engine (LSE). Synopsys® Synplify Pro® for Lattice .
	Simulation	Refer to the Lattice Radiant Software User Guide for the list of supported simulators.

Notes:

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.
2. Lattice Implementation indicates the IP version release coinciding with the software version release. Check the software for IP version compatibility with earlier or later software versions.

1.2. Features

Key features of Low Power Double Data Rate 4 (LPDDR4) Memory Module include:

- Supports LPDDR4 memory interface
- Frequency Supported: 200, 250, 300, 350, 400, 533 MHz
- Supports 8:1 (X4) gearing ratio
- Write Leveling support for LPDDR4
- DQ-DQS skew optimization for Write Training
- Dynamic valid window optimization (Read and Write Path)
- Configurable number of chip selects
- Configurable number of clocks
- Internal programmable Vref
- Includes PLL for clock generation

1.3. Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.3.2. Signal Names

Signal names that end with:

- `_n` are active low
- `_i` are input signals
- `_o` are output signals
- `_io` are bi-directional input/output signals

2. Functional Description

2.1. Overview

LPDDR4 Memory Interface Module instantiates the DDR primitives to implement the necessary features for interfacing with LPDDR4 memory devices with data rates up to 1,066 Mbps. This module is used to pass commands and data signals from the FPGA fabric operating in system clock to the LPDDR4 memory device operating in the high-speed DDR clock. The LPDDR4 Memory Interface is operating on 8:1 gearing ratio or X4 gearing mode which means:

- The DDR clock (ddr_ck_o) frequency is 4x the system clock (sclk_o) frequency
- The bit width of data signal in system clock domain is equivalent to DDR DQ/DQS/DMI bit width x 8. Thus, it takes 4 ddr_ck_o cycles to transfer the data to/from system clock domain. Note that data is transferred on both ddr_ck_o edges.
- On the other hand, bit width of command/address signal in system clock domain is equivalent to DDR CKE/CS/CA/ODT bit width x 4. These signals are shifted out on ddr_ck_o falling edge and the external memory sample these on the rising edge.

This module is designed to be used with a memory controller as show in [Figure 2.1](#). The memory controller organizes the command and data signals in system clock domain, and this module converts them to high-speed LPDDR4 signaling.

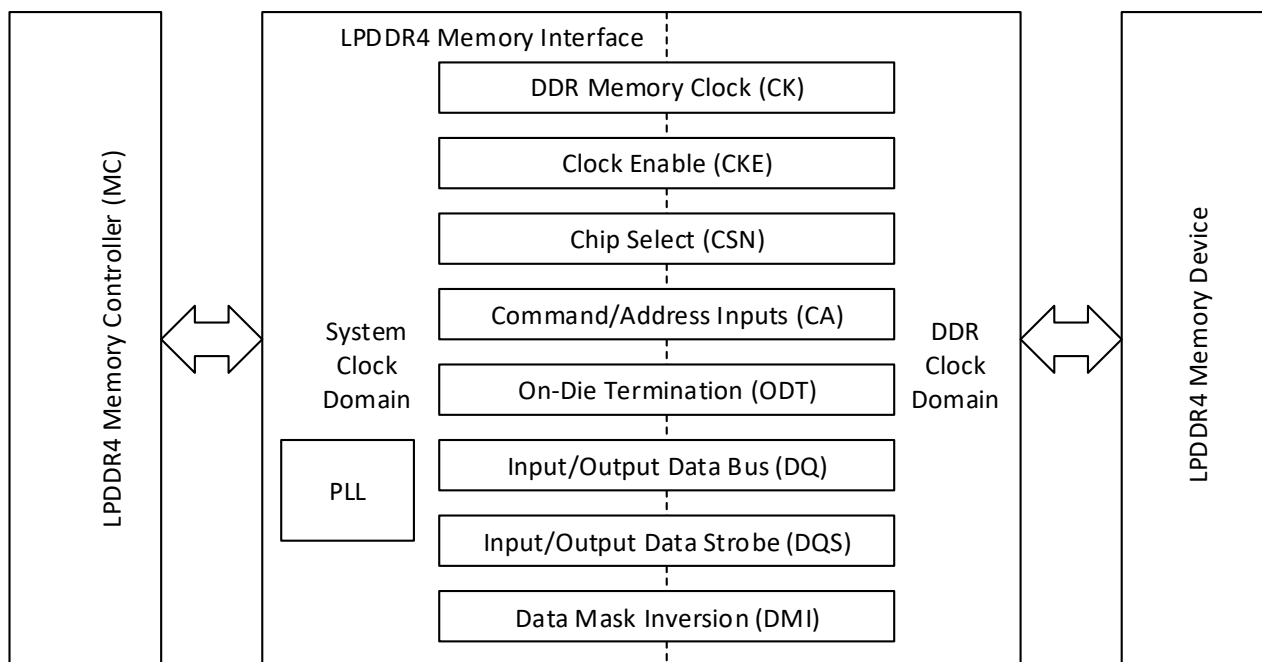


Figure 2.1. LPDDR4 Memory Interface Application

The top-level block diagram of the LPDDR4 Memory Interface Module is shown in Figure 2.2. This shows that the bit width of command/address path signals in the memory controller side is 4x of the bit width of the corresponding memory the device-side signals. While bit width of data I/O path signals in the memory controller side is 8x of the bit width of the memory the device-side signals.

The command bus signals have configurable output delay signals per bit (Per Bit Delay Adj.) to support command bus training. The data I/O path on the other hand have two levels of delay adjustment: the Per DQS Grp. Delay Adj. and the Per Bit Delay Adj.

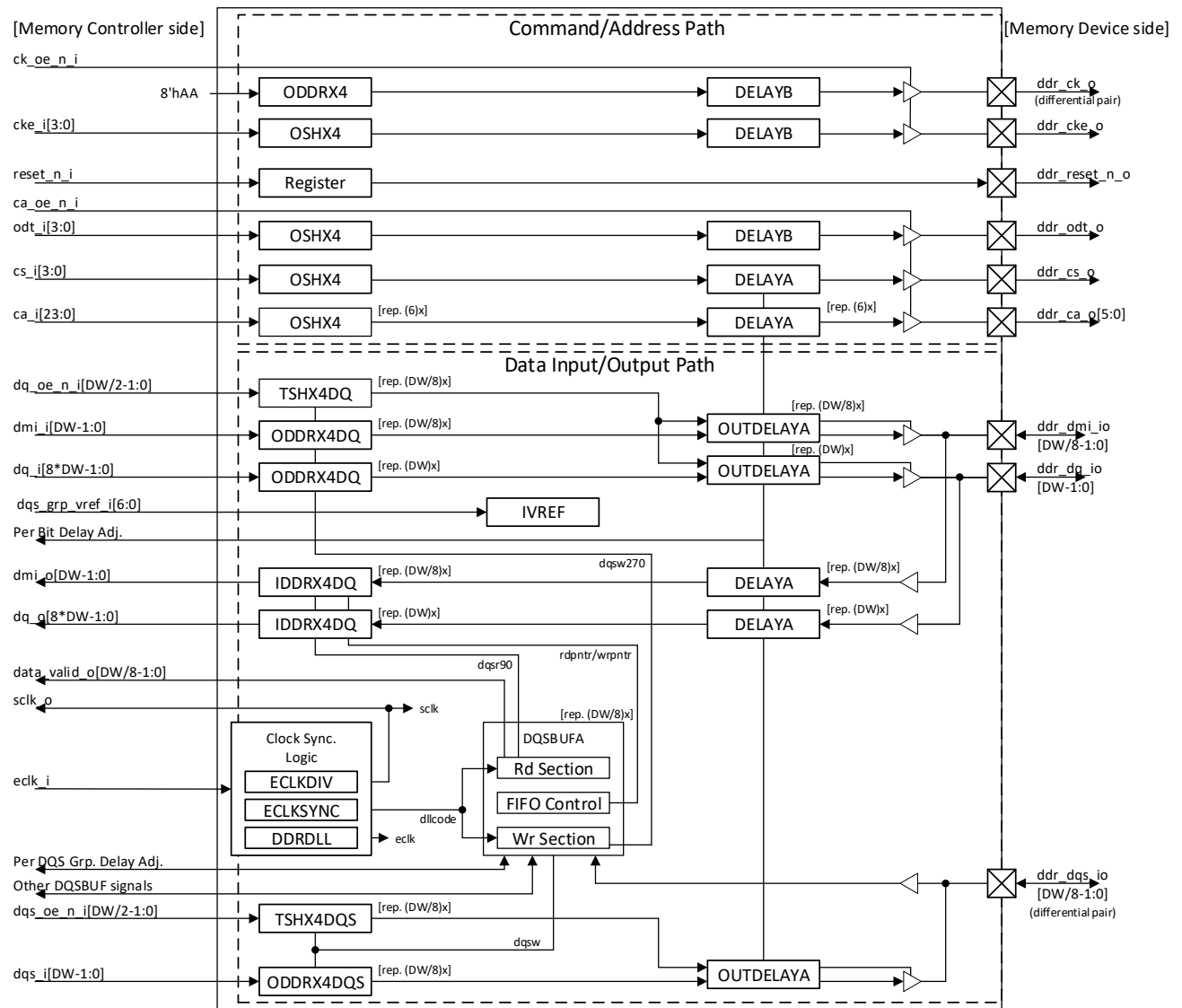


Figure 2.2. DDR Memory Interface Module Block Diagram (No PLL Instance)

2.2. Signal Description

Table 2.1 describes the LPDDR4 Memory Interface Module signals. The signals are grouped as follows:

- Clock and Reset – As the group name implies, these are clock and reset signals. This also includes the signal that controls the clock and indicates the clock ready status. Please refer to [Clock Synchronization Logic](#) section for more information on this group.
- User Interface – The LPDDR4 Controller use this group to control the LPDDR4 Interface group in implementing the LPDDR4 protocol. Please refer to [Data Input/Output Path](#) and [Command/Address Path](#) sections for more information on this group.
- Per DQS Group Delay Adjustment – Implements delay adjustment for DQS-DQ read data path, DQS-DQ write data path and CK-DQS write data path (write leveling). A DQS group is composed of 8-bit of DQ and the corresponding DQS and DMI bit. Please refer to [Training Support](#) section for more information on this group.
- Per Bit Delay Adjustment – Implements delay per bit adjustment for the command bus (CS/CA). Please refer to [Command Bus Training](#) section for more information on this group.

Note: We are planning to remove the per bit delay adjustment on DQ/DMI signal

Table 2.1. LPDDR4 Memory Interface Module Signal Description

Pin Name	Direction	Width (Bits) ¹	Description
Clock and Reset			
pll_refclk_i	In	1	Reference clock input for PLL
pll_lock_o	Out	1	Signal from PLL indicating stable clock condition
rst_i	In	1	Active HIGH reset signal
sync_clk_i	In	1	Input clock that drives the soft logic that controls the PLL instance. You should not connect to sclk_o because the sclk_o is stopped during clock frequency change. If the <i>PLL Reference Clock from Pin</i> attribute is checked, you should not connect this to pll_refclk_i.
sclk_o	Out	1	Output clock with frequency equal to DDR clock divided by 4
ddr_clk_sel_i	In	3	Selects the DDR clock frequency when performing clock frequency change. This signal must remain stable when clock_update_i asserts until ready_o negates. 0: Operating frequency – <i>DDR Memory Frequency</i> attribute 1: Command Bus Training frequency – 50 MHz
dll_update_en_i	In	1	Enables DLL update during clock frequency change. DLL update has long delay. This is recommended to only perform DLL update on the final clock frequency change of command bus training.
clk_update_i	In	1	A transition from LOW to HIGH triggers the clock frequency change according to ddr_clk_sel_i signal. This signal must remain stable until ready_o goes HIGH again.
ddr_ck_en_i	In	1	Enables the clock toggling of ddr_ck_o
ready_o	Out	1	Indicates internal DDR clock and sclk_o are stable and ready to operate. This signal negates during clock frequency change and asserts again when both DDR clock and sclk_o are valid for the new frequency setting.
LPDDR4 User Interface			
ck_oe_n_i	In	1	Tristate control port for CK/CKE
cke_i	In	4	Clock Enable input
ca_oe_n_i	In	1	Tristate control port for CS/CA/ODT
cs_i	In	4	Chip Select input
ca_i	In	24	Command/Address input
odt_i	In	4	On-die Termination input
reset_n_i	In	1	DDR Reset input
write_dqs_oe_n_i	In	DW/2	Tristate control port for DQS
write_dqs_i	In	DW	Parallel DQS input
write_dq_oe_n_i	In	DW/2	Tristate control port for DQ/DMI.

Pin Name	Direction	Width (Bits) ¹	Description
write_dmi_i	In	DW	Data Mask inversion input.
write_dq_i	In	8*DW	Parallel data bus input.
read_dqs_ie_i	In	4	Read enable signal for capturing the incoming read BL16. Each bit captures the DQ/DMI for the corresponding ddr_ck_o cycle. Please refer to DQS Read Training for more information.
read_dmi_o	Out	DW	Data Mask Inversion output.
read_dq_o	Out	8*DW	Parallel data bus output.
read_data_valid_o	Out	DW/8	Data valid flag for READ mode.
DQS Group Training			
pause_i	In	1	Set this to 1 to stop the DQSBUF-generated internal clocks when updating rd_clkssel_i and the delay codes. This is to avoid metastability.
rd_clkssel_i	In	4	Used to select read clock source and polarity control. [1:0]: selects the 0, 45, 90, and 135 (2b00 to 2b11 respectively) degree paths from the DQS write section delay cell. [2] = 1b0: use inverted clock [2] = 1b1: use non-inverted clock – adds 180-degree shift [3] = 1b0: bypasses the register in the read enable path [3] = 1b1: selects the register in the read enable path Please refer to DQS Read Training for more information.
burst_det_sclk_o	Out	DW/8	Clock generated using burst_det_o.
dqs_grp_vref_i	In	7	Controls I/O drive strength of data I/O path signals by changing the internal voltage reference.
rd_load_n_i	In	DW/8	Asynchronously resets the delay code according to the <i>Dynamic Margin Control</i> attribute for DQS-DQ skew compensation in read data path.
rd_move_i	In	DW/8	At rising edge, it changes (+/- 1) the delay code according to rd_dir_i signal for DQS-DQ in read data path.
rd_dir_i	In	DW/8	Controls the direction of delay code change for DQS-DQ skew compensation in read data path. '0' to increase and '1' to decrease the delay code.
rd_cout_o ⁴	Out	DW/8	Margin test output flag to indicate the under-flow or over-flow in delay code for DQS-DQ skew compensation in read data path.
wr_load_n_i	In	DW/8	Reset the delay code according to the <i>Dynamic Margin Control</i> attribute for DQS-DQ skew compensation in write data path. A pulse (Low-High-Low) in wr_move_i is needed to perform reset.
wr_move_i	In	DW/8	At rising edge, it changes (+/- 1) the delay code according to wr_dir_i signal for DQS-DQ skew compensation in write data path.
wr_dir_i	In	DW/8	Controls the direction of delay code change for DQS-DQ skew compensation in write data path. '0' to increase and '1' to decrease the delay code.
wr_cout_o ⁴	Out	DW/8	Margin test output flag to indicate the under-flow or over-flow in delay code for DQS-DQ skew compensation in write data path.
wrlvl_load_n_i	In	DW/8	Asynchronously resets the delay code to factory default value for CK-DQS skew compensation – Write Leveling. Note: The reset value is 0 (subject for revision).
wrlvl_move_i	In	DW/8	At rising edge, it changes (+/-1) the delay code according to the direction set by wrlvl_dir_i for CK-DQS skew compensation – Write Leveling.
wrlvl_dir_i	In	DW/8	Controls the direction of delay code change for CK-DQS skew compensation (Write Leveling) in write data path. '0' to increase and '1' to decrease the delay code.
wrlvl_cout_o ⁴	Out	DW/8	Margin test output flag to indicate the under-flow or over-flow for CK-DQS skew compensation – Write Leveling.

Pin Name	Direction	Width (Bits) ¹	Description
dqwl_o ⁴	Out	DW	Data output of write leveling.
Per Bit DQ/DMI Training			
dq_out_adj_load_n_i ²	In	DW	DQ output delay per bit adjustment load signal.
dq_out_adj_move_i ²	In	DW	DQ output delay per bit adjustment signal.
dq_out_adj_dir_i ²	In	DW	DQ output delay per bit adjustment direction signal.
dq_out_adj_cout_o ^{2,4}	Out	DW	DQ output delay per bit adjustment cout signal.
dmi_out_adj_load_n_i ²	In	DW/8	DMI output delay per bit adjustment load signal.
dmi_out_adj_move_i ²	In	DW/8	DMI output delay per bit adjustment move signal.
dmi_out_adj_dir_i ²	In	DW/8	DMI output delay per bit adjustment direction signal.
dmi_out_adj_cout_o ^{2,4}	Out	DW/8	DMI output delay per bit adjustment cout signal.
dqs_out_adj_load_n_i ²	In	DW/8	DQS output delay per bit adjustment load signal.
dqs_out_adj_move_i ²	In	DW/8	DQS output delay per bit adjustment move signal.
dqs_out_adj_dir_i ²	In	DW/8	DQS output delay per bit adjustment direction signal.
dqs_out_adj_cout_o ^{2,4}	Out	DW/8	DQS output delay per bit adjustment cout signal.
dq_in_adj_load_n_i ³	In	DW	DQ input delay per bit adjustment load signal.
dq_in_adj_move_i ³	In	DW	DQ input delay per bit adjustment signal.
dq_in_adj_dir_i ³	In	DW	DQ input delay per bit adjustment direction signal.
dq_in_adj_cout_o ^{3,4}	Out	DW	DQ input delay per bit adjustment cout signal.
dmi_in_adj_load_n_i ³	In	DW/8	DMI input delay per bit adjustment load signal.
dmi_in_adj_move_i ³	In	DW/8	DMI input delay per bit adjustment move signal.
dmi_in_adj_dir_i ³	In	DW/8	DMI input delay per bit adjustment direction signal.
dmi_in_adj_cout_o ^{3,4}	Out	DW/8	DMI input delay per bit adjustment cout signal.
Command Bus Training			
cs_adj_load_n_i ³	In	1	CS delay adjustment load signal.
cs_adj_move_i ³	In	1	CS delay adjustment signal.
cs_adj_dir_i ³	In	1	CS delay adjustment direction signal.
cs_adj_cout_o ^{3,4}	Out	1	CS delay adjustment cout signal.
ca_adj_load_n_i ³	In	6	CA delay per bit adjustment load signal.
ca_adj_move_i ³	In	6	CA delay per bit adjustment signal.
ca_adj_dir_i ³	In	6	CA delay per bit adjustment direction signal.
ca_adj_cout_o ^{3,4}	Out	6	CA delay per bit adjustment cout signal.
LPDDR4 Interface			
ddr_ck_o	Out	1	LPDDR4 Clock output (CK). This becomes differential pair signal at the FPGA IO pad.
ddr_cke_o	Out	1	LPDDR4 Clock Enable output (CKE).
ddr_cs_o	Out	1	LPDDR4 Chip Select output (CS).
ddr_ca_o	Out	6	LPDDR4 Command/Address output (CA).
ddr_odt_o	Out	1	LPDDR4 On-Die Termination Control output (ODT).
ddr_reset_n_o	Out	1	LPDDR4 Reset output (RESET_n)
ddr_dq_io	In/Out	DW	LPDDR4 Data Input/Output (DQ).
ddr_dqs_io	In/Out	DW/8	LPDDR4 Data Strobe Input/Output (DQS). This signal becomes differential pair at the IO FPGA pad.
ddr_dmi_io	In/Out	DW/8	LPDDR4 Data Mask Inversion Input/Output (DMI).

Notes:

1. DW = Data Bus Width attribute
2. These signals are connected to OUTDELAYA primitive, refer to the [Signal Description](#) section for the signal description.
3. These signals are connected to DELAYA primitive, refer to the [Signal Description](#) section for the signal description.
4. All the *cout_o and dqwl_o signals must be registered directly before driving other logic.

2.3. Attribute Summary

Table 2.2. Attribute Table

Attribute	Selectable Values	Default	Dependency on Other Attributes
General Tab			
General Group			
I/O Buffer Type	LVSTL_I, LVSTL_II	LVSTL_I	
Gearing Ratio	8:1	8:1	Display Information only
Data Bus Width	16, 32, 64	16	—
Number of DQS Group	Calculated = (Data Bus Width)/8	N/A	Data Bus Width
Enable Dynamic Margin Control on Clock Delay	Checked, Unchecked	Checked	Display Information only
Enable Per Bit Delay Adjustment on DQS Group	Checked, Unchecked	Unchecked	—
Enable Per Bit Delay Adjustment on CA Group	Checked, Unchecked	Checked	—
Clock Settings Group			
DDR Memory Frequency (MHz)	200, 250, 300, 350, 400, 533	400	—
System Clock Frequency (MHz)	Calculated = (DDR Memory Frequency) / Gearing Ratio	N/A	Display Information only
Enable PLL	Checked	Checked	Display Information only
PLL Input Frequency*	10 – 800	100	—
PLL Reference Clock from Pin	Checked, Unchecked	Unchecked	—
PLL Reference Clock I/O Type	LVSTL_I, LVSTL_II, LVSTLD_I, LVSTLD_II, LVTT133, LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS18H, HSTL15D_I, LVCMOS15, LVCMOS15H, LVCMOS12, LVCMOS12H, LVCMOS10H, LVCMOS10, LVCMOS10R	LVSTL_I	PLL Reference Clock from I/O Pin are both Checked
Clock Output Tolerance (%)	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.1	'Enable PLL' is Checked
DDR Memory Actual Frequency (MHz)	Calculated	N/A	Display Information only
Clock/Address/Command			
Number of Clocks	1, 2, 4	1	
Address Width	6	6	Display Information only
Number of Chip Selects	1, 2, 4	1	
Number of Chip ODT	1	1	Display Information only
Dynamic Margin Control			
DQS Read Delay Adjustment Enable	Checked, Unchecked	Unchecked	Enable Dynamic Margin Control on Clock Delay is Checked
DQS Read Delay Adjustment Sign	POSITIVE, COMPLEMENT	POSITIVE	DQS Read Delay Adjustment Enable is Checked
DQS Read Delay Adjustment Value	0 to 255	0	

Attribute	Selectable Values	Default	Dependency on Other Attributes
DQS Read Delay Adjustment Actual Value	Calculated		Display Information only
DQS Write Delay Adjustment Enable	Checked, Unchecked	Unchecked	<i>Enable Dynamic Margin Control on Clock Delay</i> is Checked
DQS Write Delay Adjustment Sign	POSITIVE, COMPLEMENT	POSITIVE	
DQS Write Delay Adjustment Value	0 to 255	0	<i>DQS Write Delay Adjustment</i> Enable is Checked
DQS Write Delay Adjustment Actual Value	Calculated		
DQS Write Delay Adjustment Actual Value	Calculated		Display Information only

***Note:** The *PLL Input Frequency* attribute is fixed to 100 MHz for IP Core v1.0.0. This option to be enabled to support multiple frequencies in the next release.

2.4. DDR Memory Primitives

This section briefly describes the Lattice FPGA primitives that are used to implement the LPDDR4 Memory Module. Please refer to **Reference Guides > FPGA Libraries Reference Guide > Primitive Library - <Device Name>** of the Lattice Radiant Software Help for more information on these FPGA primitives. [Figure 2.3](#) navigates the LFCPNX Primitive Library from the Lattice Radiant Software Help.

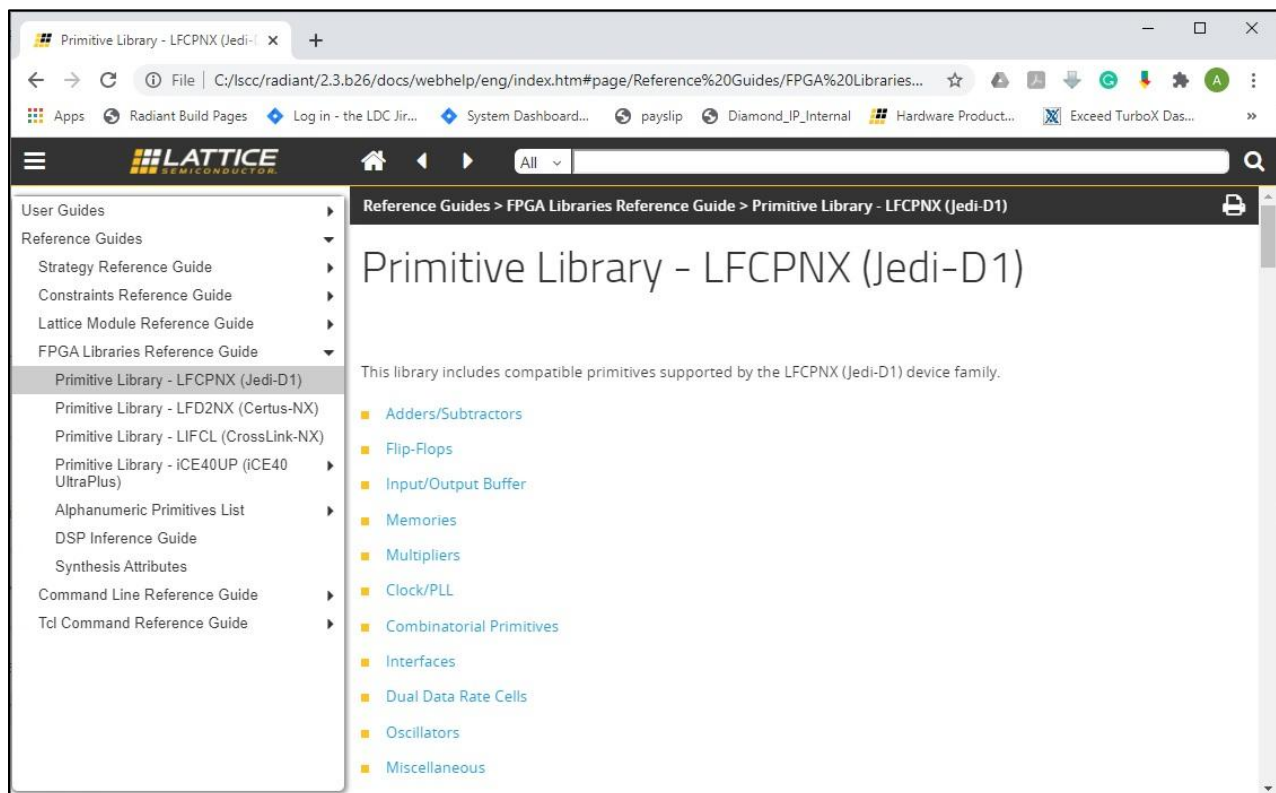


Figure 2.3. Reference Guide for the Primitive Library

2.4.1. Input/Output DELAY

The DELAY blocks can be used to delay the signal in either of the following paths:

- Delay the input signal – from the input pin to the IDDR or IREG or FPGA Fabric
- Delay the output signal – from the ODDR, OREG or FPGA fabric to the output pin

This is useful to adjust for any skews amongst the input or output data bus. It can also be used to generate skew between the bits of output bus to reduce SSO noise. The DELAY block can be used with IDDR or ODDR modules, SDR module, as well as on the direct input pin of the FPGA. The DELAY is shared by the input and output paths and hence can only be used either to delay the input data or the output signal on a given pin.

The delayed value are set using:

- Pre-determined delay value (for Zero Hold time, delay based on Interface Type)
- Fixed delay values entered by user
- Dynamically updated using counter up and down controls

The DELAY block can be completely bypassed as well.

2.4.1.1. DELAYA

By default, the DELAYA primitive is configured to factory delay settings based on the clocking structure. Users can update the DELAY setting using the MOVE and DIRECTION control inputs. The LOAD_N resets the delay back to the default value.

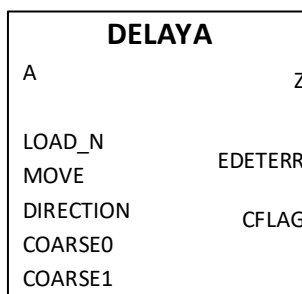


Figure 2.4. DELAYA Block Diagram

2.4.1.2. DELAYB

The DELAYB primitive is configured to factory delay settings based on the clocking structure. Users cannot change the delay when using this module.

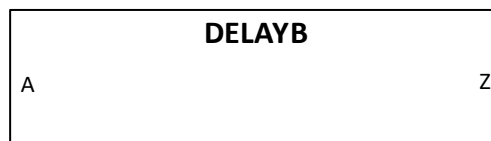


Figure 2.5. DELAYB Block Diagram

2.4.1.3. OUTDELAYA

The OUTDELAYA primitive is used to add static and dynamic delay to DOUT and TOUT for write training in LPDDR4 mode.

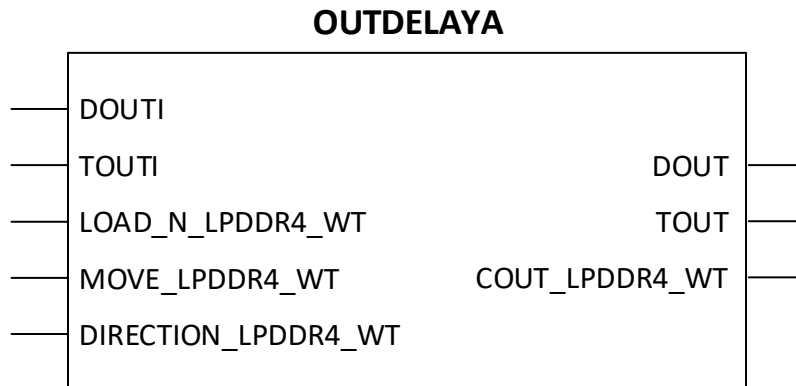


Figure 2.6. OUTDELAYA Block Diagram

2.4.2. IDDR/ODDR

This section describes the primitives used to build LPDDR2, DDR3 and LPDDR4 memory interfaces. Some of these primitives are also used to generate generic DDR functions that use DQS clocking tree. The IDDR/ODDR primitives support 4:1(X2), 8:1(X4), gearing modes that are used to implement the memory functions. The gearing mode defines the ratio of system clock and DDR clock frequencies. The data from the FPGA fabric transferred using the rising edge of system clock (slower frequency) while the data in DDR interface side is transferred in both edges of the DDR clock (faster frequency).

A 4:1 (X2) gearing mode means:

- Clock frequency ratio: 1 system clock : 2 DDR clock
- Data bit width ratio: 4 bits of data in system clock rising edge: 1 bit of data in for each of 4 DDR clock edges

Similarly, an 8:1 (X4) gearing mode means:

- Clock frequency ratio: 1 system clock : 8 DDR clock
- Data bit width ratio: 8 bits of data in system clock rising edge: 1 bit of data in for each of 8 DDR clock edges

Thus, it is recommended to use X4 gearing mode when a low system clock frequency or a high DDR clock frequency is desired.

2.4.2.1. IDDRX4DQ

This primitive is used to capture the data bits from DDR3 or LPDDR4 interface for X4 gearing mode.

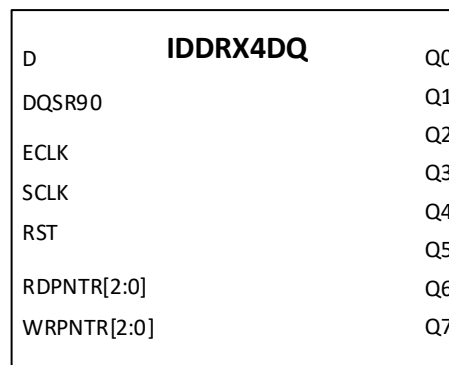


Figure 2.7. IDDRX2DQ Block Diagram

2.4.2.2. ODDRX4DQ

This primitive is used to generate the DQ data output of DDR3 or LPDDR4 memory interface for X4 gearing mode.

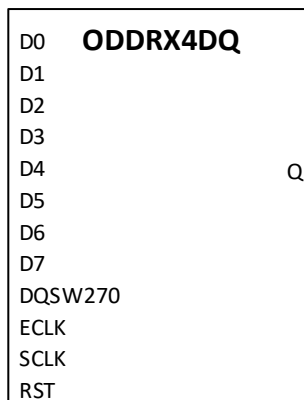


Figure 2.8. ODDRX4DQ Block Diagram

2.4.3. ODDRX4DQS

This primitive is used to generate DQS clock output of DDR3 and LPDDR4 memory for X4 gearing.

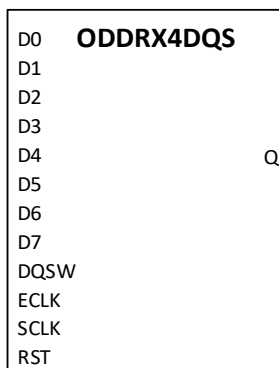


Figure 2.9. ODDRX4DQS Block Diagram

2.4.4. Memory Output DDR Primitives for Tristate Output Control

The following are the primitives used to implement tristate control for the outputs to the DDR memory.

2.4.4.1. TSHX4DQ

This primitive is used to generate the tristate control for DQ data output of DDR3 memory interface with x4 gearing mode.

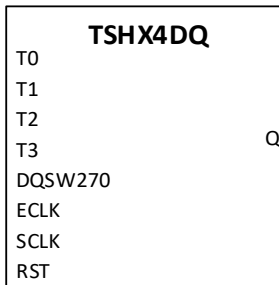


Figure 2.10. TSHX4DQ Block Diagram

2.4.4.2. TSHX4DQS

This primitive is used to generate the tristate control to DQS output for DDR3 or LPDDR4 memory interface for x4 gearing mode.

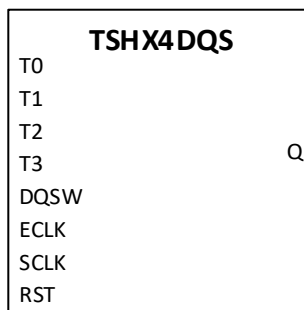


Figure 2.11. TSHX4DQS Block Diagram

2.4.5. OSHX4

This primitive is used to generate the address and command signals of DDR3 or LPDDR4 memory interface for x4 gearing mode and write leveling.

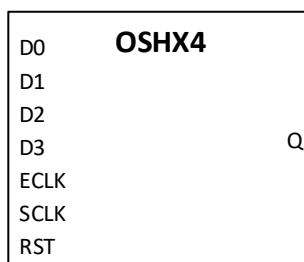


Figure 2.12. OSHX4 Block Diagram

2.4.6. DDRDLL

DDRDLL generates a phase shift code (90 degrees) according to its running frequency. This code is provided to every individual DQS block and DLLDEL slave delay element located in 2 adjacent sides if available.

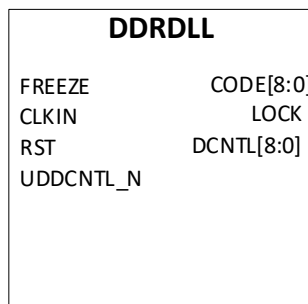


Figure 2.13. DDRDLL Block Diagram

2.4.7. ECLKDIV

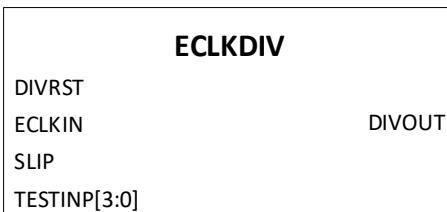


Figure 2.14. ECLKDIV Block Diagram

2.4.8. ECLKSYNC

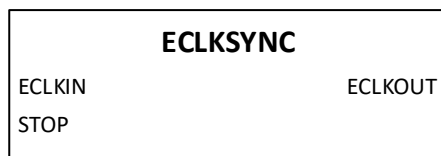


Figure 2.15. ECLKSYNC Block Diagram

2.4.9. DQSBUF_IVREF

To support DDR memory interfaces (DDR2/3, LPDDR2/3/4), the DQS strobe signal from the memory must be used to capture the data (DQ) in the PIC registers during memory reads.

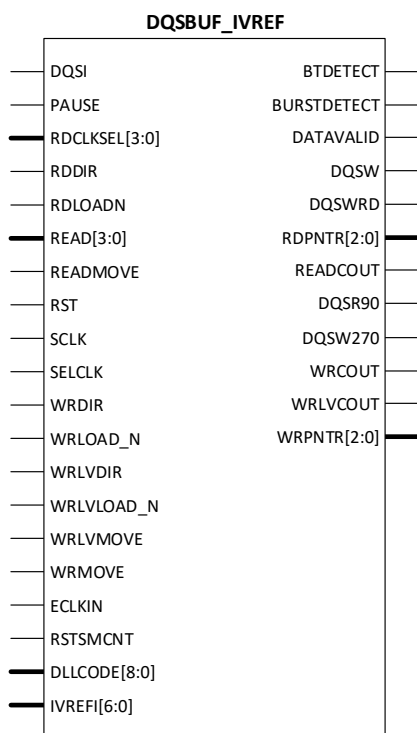


Figure 2.16. DQSBUF_IVREF Block Diagram

2.5. Clock Synchronization Logic

Figure 2.17 shows the block diagram of the clock synchronization logic. The PLL generates the following clocks:

- CLKOP – a slow clock equal to the frequency of the reference lock – set by *PLL Input Frequency* attribute. This clock is used to clock the MEM_SYNC.
- CLKOS – used for sclk_o and eclk_o (internal DDR clock that clocks the DDR primitives) generation. The CLKOS frequency is set by *DDR Memory Frequency* attribute, which is the operating clock. The CLKOS frequency is selectable between operating clock (*DDR Memory Frequency*) and 50 MHz which is the command bus training clock.

The ECLKSYNC and ECLKDIV generates the eclk_o and sclk_o respectively. These clocks are routed to a dedicated clock route resource which ensures correct timing. The DDRDLL generates the DLL code which is used by the DQSBUF for generating the correct phase delays of the internal clocks for the data path. These DDR clock primitives are controlled by MEM_SYNC which is implemented in soft logic.

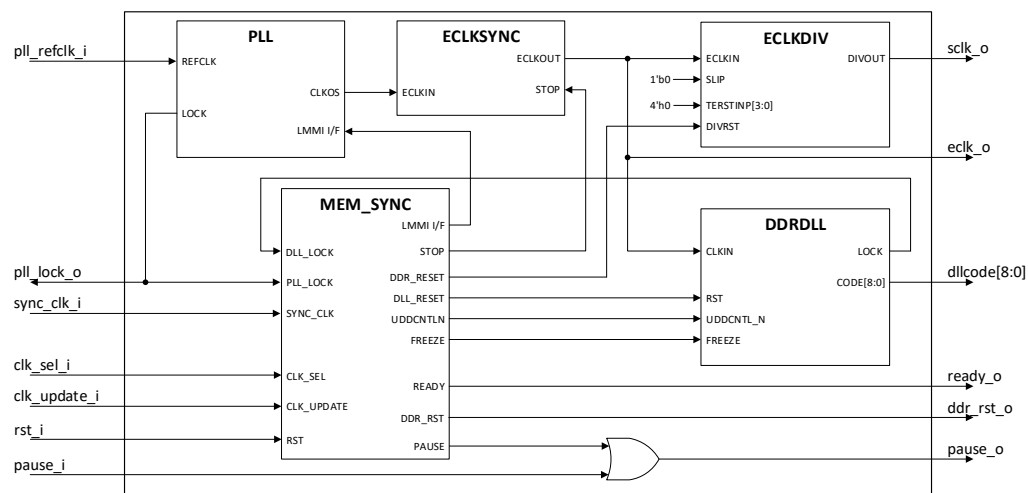


Figure 2.17. Clock Synchronization Logic Block Diagram

An example of clock frequency change timing diagram is shown in Figure 2.18, the order of signal transitions are accurate but the cycle-by-cycle relationship is not. This is because the signals are operating in 3 clock – sclk_o, internal eclk_o (CLKOS) and CLKOP (not shown). The signals moving across clock domains avoid metastability by implementing clock crossing logic or only allow transition during freeze/stop/pause signals are asserted.

The clock frequency change is initiated by a Low to High transition on clk_update_i signal and the ddr_clk_sel_i specifies the target new frequency: 1'b0 for Operating Frequency and 1'b1 for command bus training frequency. The ready_o signal negates in the next sclk_o cycle, indicating that the sclk_o and ddr_ck_o are not yet ready. The controller must not issue any request - command/data/delay code update while ready_o signal is low. The MEM_SYNC.freeze signal asserts and is followed by MEM_SYNC.stop assertion to properly stop the sclk_o and eclk_o before the actual PLL clock change which occurs during both freeze and stop signals asserted. When these signals are negated, the PLL is already generating the target frequency. The sclk_o and ddr_ck_o resumes at the target frequencies.

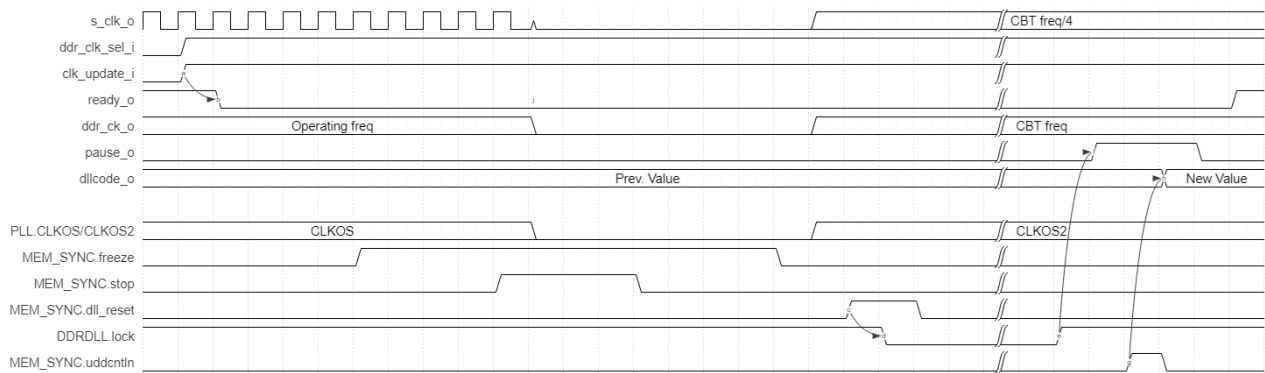


Figure 2.18. Clock Synchronization Logic Block Diagram

The DLL code need to be updated for the new frequency. This is done by asserting the `MEM_SYNC.dll_reset`, the pause signal is asserted before and after `dll_reset` to avoid metastability. The `DDRDLL.lock` negates after the `dll_reset` and asserts after the `DDRDLL` locks to the new frequency. The `MEM_SYNC.uddcntIn` pulses to update the DLL code, the `pause_o` is also asserted before and after this pulse for the DQSBUF to properly update for the new DLL code.

The `ready_o` signal asserts to indicate that the clocks are now stable and the module is now ready to receive new requests from the controller. The `clk_update_i` may be negated any time after the `ready_o` signal asserts.

2.6. Data Input/Output Path

The block diagram of Data Input/Output Path describing the DDR primitive connections are shown in Figure 2.19.

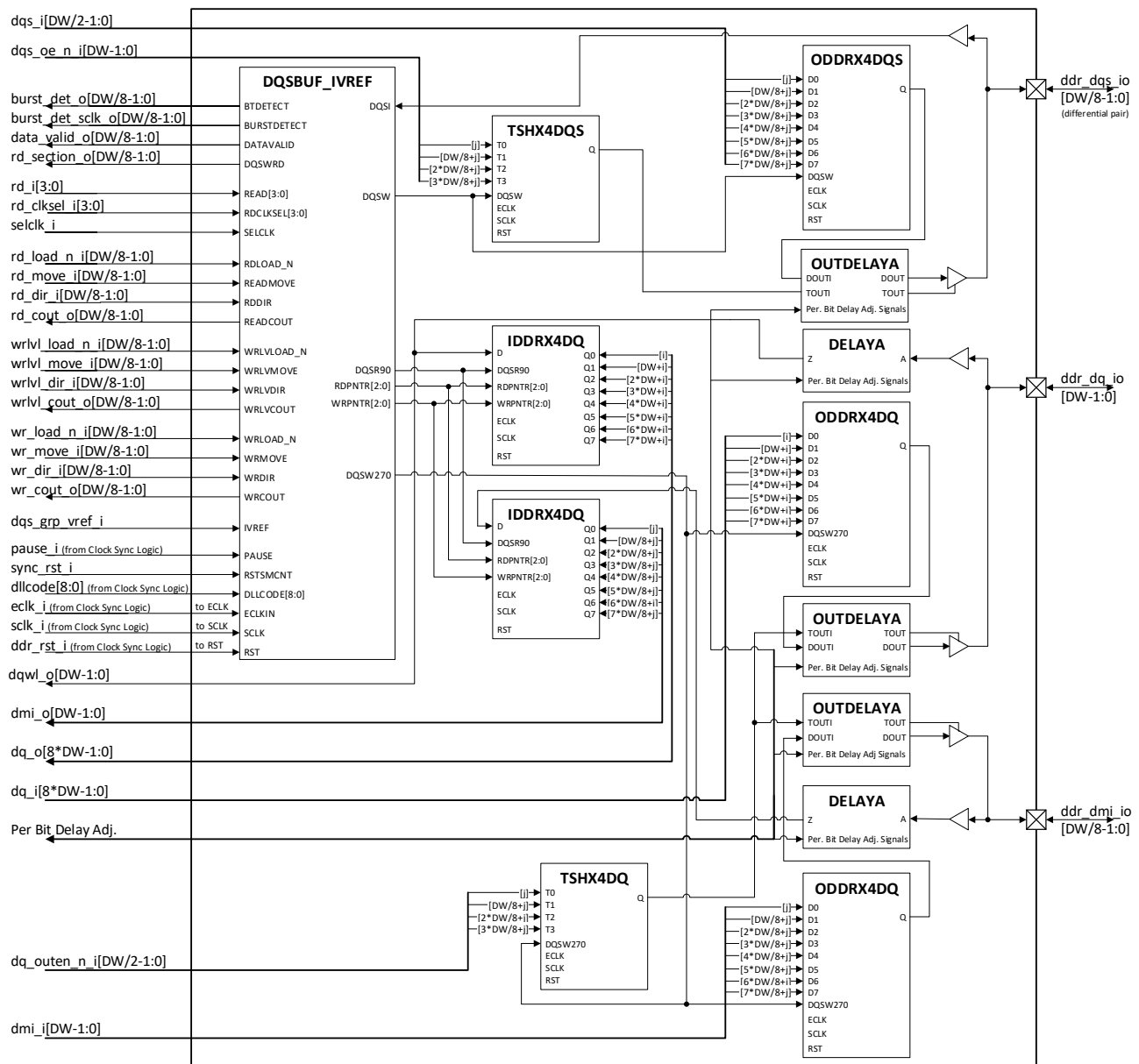


Figure 2.19. Data Input/Output Path Block Diagram

The DQSBUF_IVREF generates the following 3 internal clocks which are generated according to the DLL code:

- DQSW – clocks the write DQS. The delay is controlled by the wrlvl_<load_n/move/dir> signals (write leveling).
- DQSW270 – clocks the write DQ/DMI. The delay is controlled by the wr_<load_n/move/dir> signals
- DQSR90 – clocks the read DQ/DMI. The delay is controlled by the rd_<load_n/move/dir> signals

This internal clock is only activated when a read pre-amble DQS=0 followed by DQS pulse has been successfully captured. A read BL16 produces eight DQSR90 pulses.

Important Note:

The delay codes of DQSBUF_IVREF is not initialized during reset. The controller or user logic must reset the delay codes by driving 0 logic to wrlvl_load_n, wr_load_n and rd_load_n before performing any training.

2.6.1. Write Data access

Figure 2.20 shows the timing diagram example for write BL16 data access with *Data Bus Width* = 32 (DQS group = 4). The pre-amble is 2nCK cycles and the postamble for this example is 0.5nCK. The 2nCK pre-amble is generated by setting `write_dqs_i` = 0xFF. F during disable cycle and enabling DQS 1 CK earlier than the DQ/DML. The input is internally registered at the first rising edge of `sclk_o` and is output serially after 2 `sclk_o` cycles. Thus, the latency is 3 `sclk_o` cycles. The output enable signal `write_dqs_oe_n_i` and `write_dq_oe_n_i` are 4x the bit width of the DQS group (DW/8). The first DW/8 bits controls the output enable for the DDR clock cycle that crosses the `sclk_o` rising edge. The least significant bytes/bits of `write_dq_i`/`write_dmi_i` are transmitted first in the DDR side. In the diagram, the lower bits of `write_dq_i`/`write_dmi_i` are do not care because the corresponding `write_dq_oe_n_i` bits are “1” (output disabled). Adding 1 `ddr_ck_o` cycle latency in DDR side means shifting the inputs in the User Interface side. For example, the Figure 2.21 is equal to Figure 2.20 with added 2nCK latency.

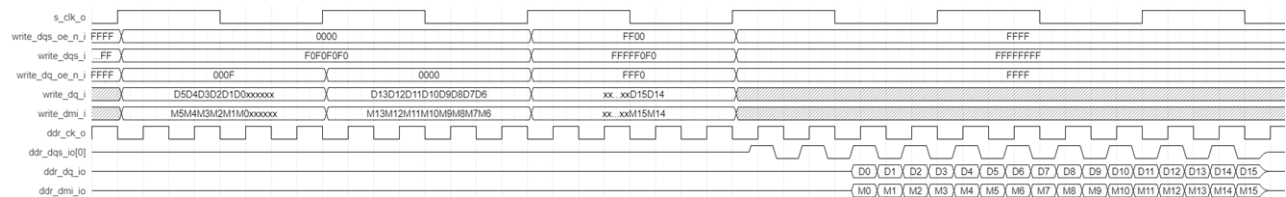


Figure 2.20. Write BL16 timing diagram

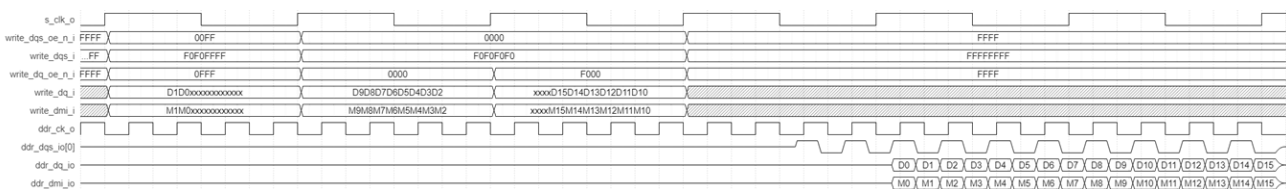


Figure 2.21. Write BL16 timing diagram with added 2nCK latency

2.6.2. Read Data access

The `read_dqs_ie_i` signals the DQSBUF to capture the incoming read burst, the bits[3:0] indicates that corresponding DDR clock cycle is captured. In Figure 2.22, `read_dqs_ie_i` = 4'hF for 2 consecutive SCLK cycles, starting from `sclk_o` rising edge at tick 9. This means the 8x2=16 samples of DQ/DML are captured continuously starting from next rising edge of internal DDR clock for read path after the `sclk_o` rising edge at tick 25 – this is 3 SCLK rising edge including the 1st, `read_dqs_ie_i` = 4'hF sample. The pre-amble with a DQS pulse must be provided before this time.

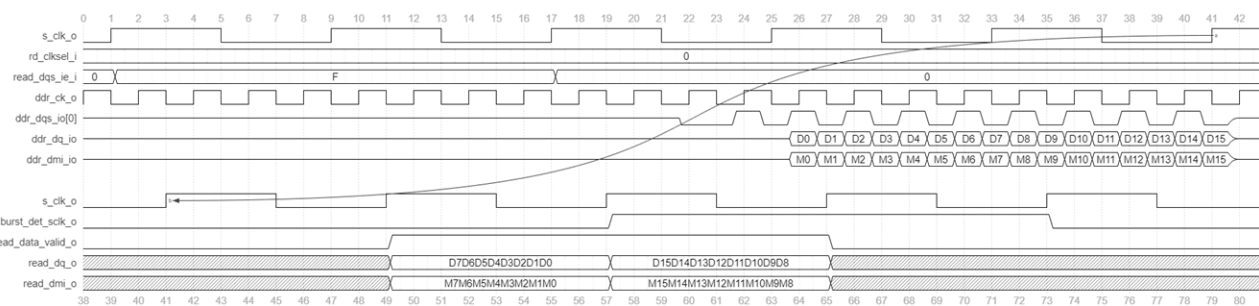


Figure 2.22. Read BL16 timing diagram (`rd_clkssel_i` = 4'h0)

When the read preamble and eight DQS pulses are captured properly, `burst_det_sclk_o` asserts for two `sclk_o` cycles or more. If `burst_det_sclk_o` did not assert or has asserted for only 1 `sclk_o` cycle, that means the captured read DQ/DML is unreliable. The timing of assertion of `burst_det_sclk_o` depends on asynchronous (with respect to `sclk_o`) reception of the incoming burst. Thus, actual assertion time is difficult to predict. Because of this, this signal is only used during DQS Read Training and Read Training.

The read_data_valid_o asserts when eight DQ/DMI samples has been captured, this signal should assert for 2 sclk_o cycles when a read BL16 has been captured properly. The read_dq_o and read_dmi_o are do not care when read_data_valid_o is negated. The eight DQ/DMI samples are transferred in 1 sclk_o cycles in the read_dq_o/read_dmi_o signals with the first sample on the LSB side.

The Figure 2.23 is similar to Figure 2.22 except that the read_clktsel_i is 4'hF which shifts the correct BL16 position by 2 ddr_ck_o cycles. The read_clktsel_i is like a phase delay adjustment of $2 \times 360^\circ / 16 = 45^\circ$ for each step.

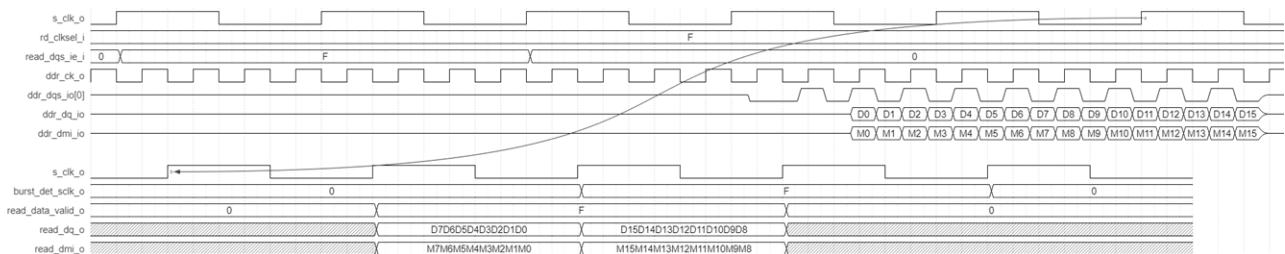


Figure 2.23. Read BL16 timing diagram (rd_clktsel_i = 4'hF)

2.6.3. DQS Read Training

For every read operation, the LPDDR4 Memory Interface Module must be initialized at the appropriate time to identify the incoming read DQS pre-amble. Upon a proper detection of the pre-amble, the DQSBUF extracts a clean signal out of the incoming DQS signal from the memory and generates the necessary control signals to the other primitives for capturing the DQ/DMI. The incoming read data is asynchronous to sclk_o but upon proper positioning of the read_dqs_ie_i and rd_clktsel_i, the captured data is aligned to BL8. Thus, no more data alignment is needed on the outputs read_dq_o and read_dmi_o.

The DQS read training process sweeps all the read_dqs_ie_i and rd_clktsel_i combination to determine this appropriate time that the incoming preamble and eight DQS pulses can be cleanly captured. Below is the DQS read training procedure:

1. Update the rd_clktsel_i value, each increment is approximately 45° phase shift. The following steps must be followed when updating the rd_clktsel_i to avoid metastability:
 - a. Asset the pause_i signal to stop the internal clocks.
 - b. Wait for 4 sclk_o cycles
 - c. Change the rd_clktsel_i to new value. This is usually increment by 1.
 - d. Wait for 4 sclk_o cycles
 - e. Negate the pause_i signal to enable the internal clocks.
2. Trigger the LPDDR4 memory to send BL16.
3. Capture the incoming BL16 using the read_dqs_ie_i signal. Each bit of the read_dqs_ie_i[3:0] signal aligns to one ddr_ck_o cycle. For a burst length of 16 (BL16), a total of 2 SCLK cycle of read_dqs_ie_i=4'F needs to be provided. Below are the valid sequences, depending on the position of the incoming pre-amble, each have consecutive 8 bits asserted:
 - a. Cycle 1: 4'b1000, Cycle 2: 4'b1111, Cycle 3: 4'b0111
 - b. Cycle 1: 4'b1100, Cycle 2: 4'b1111, Cycle 3: 4'b0011
 - c. Cycle 1: 4'b1110, Cycle 2: 4'b1111, Cycle 3: 4'b0001
 - d. Cycle 1: 4'b1111, Cycle 2: 4'b1111
4. If burst_det_sclk_o asserts for at least 2 sclk_o cycles, that means the preamble and BL16 has been properly captured. If the read data is correct, DQS Read Training is done, otherwise, proceed to next step.
5. Repeat Steps 2-4 until all the possible rd_clktsel_i sequences are swept or until DQS Read Training is done in Step 4.
6. Repeat Steps 2-4 until all the possible read_dqs_ie_i values are swept or until DQS Read Training is done in Step 4.

2.7. Command/Address Path

Figure 2.24 shows the block diagram of the command/address path. The Clock is generated by the ODDR4, the toggle starts from '0' so that values in other signals are shifted out in the ddr_ck_o falling edge and are sampled by the external LPDDR4 memory device on the rising edge for better skew tolerance. Other signals are connected to OSHX4 which means each bit (6 bits for CA) are shifted out in each clock cycle. The DELAYB is for correcting the internal skew after silicon validation. The DELAYA is used for delay adjustment during command bus training. The timing diagram of command/address path is shown in Figure 2.25.

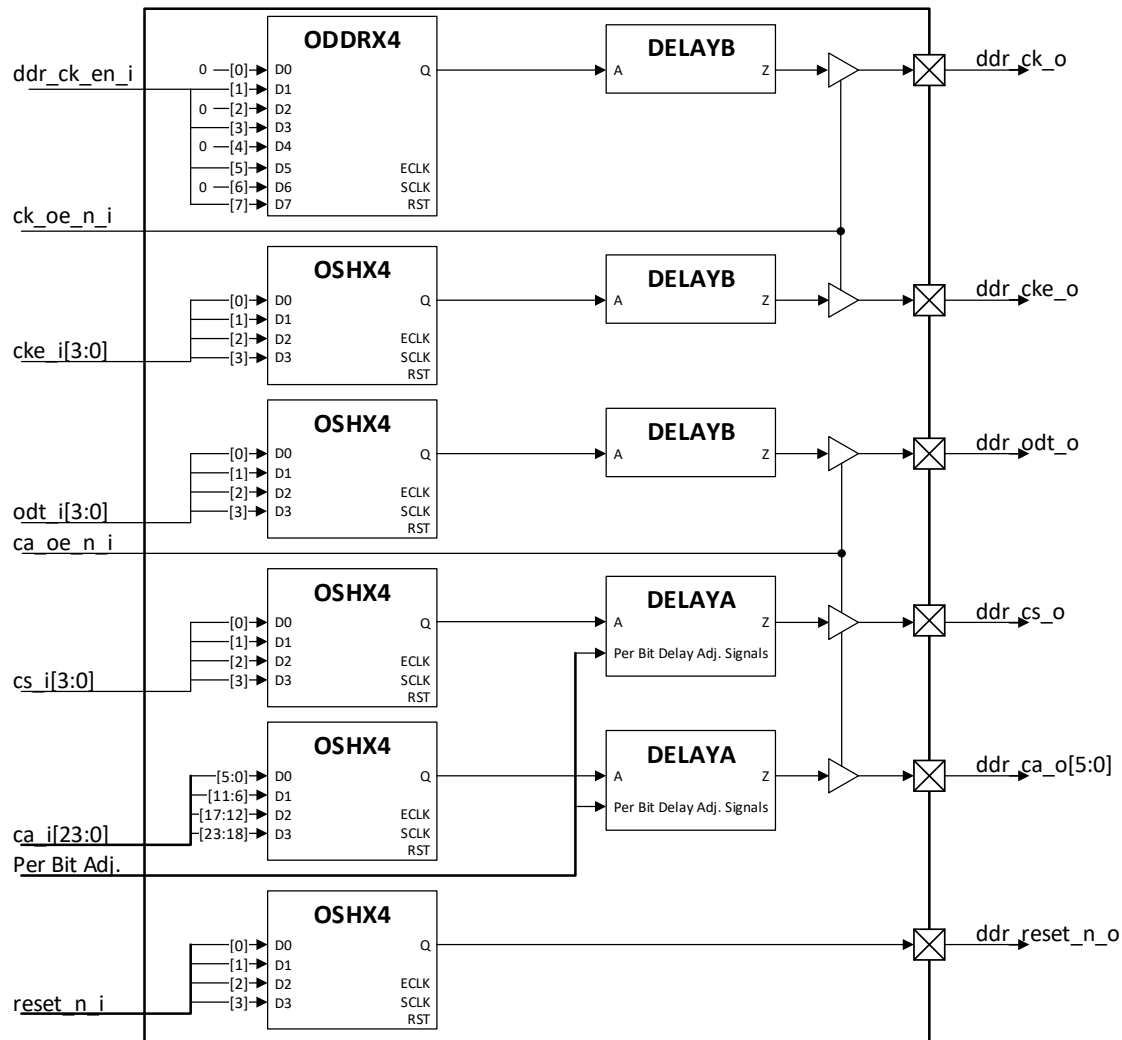


Figure 2.24. Command/Address Path Block Diagram

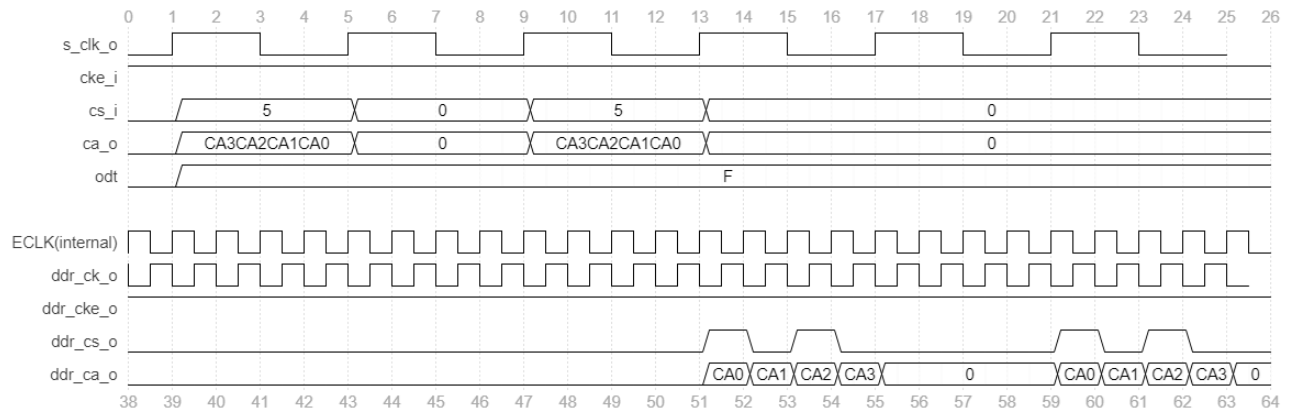


Figure 2.25. Command/Address Path Timing diagram

2.8. Training Support

2.8.1. Command Bus Training

This section describes the LPDDR4 Memory Module's support for command bus training, focusing on the Step 5 of Training Sequence for single-rank systems as described in the section 4.9.1 of the LPDDR4 Standard which says: "Perform Command Bus Training (V_{REFCA} , CS, and CA)".

This step is broken down to the following sub-steps:

- Program the LPDDR4 Memory V_{REFCA} – refer to Figure 2.26
- Send CA pattern and capture the feedback value from LPDDR4 Memory – refer to Figure 2.27
- Adjustment of the CA Bus Delay Code

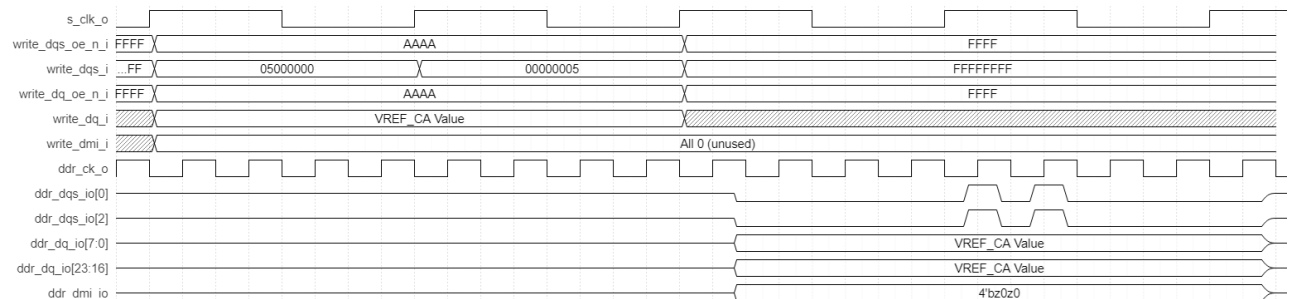


Figure 2.26. LPDDR4 Memory V_{REFCA} Programming Timing Diagram

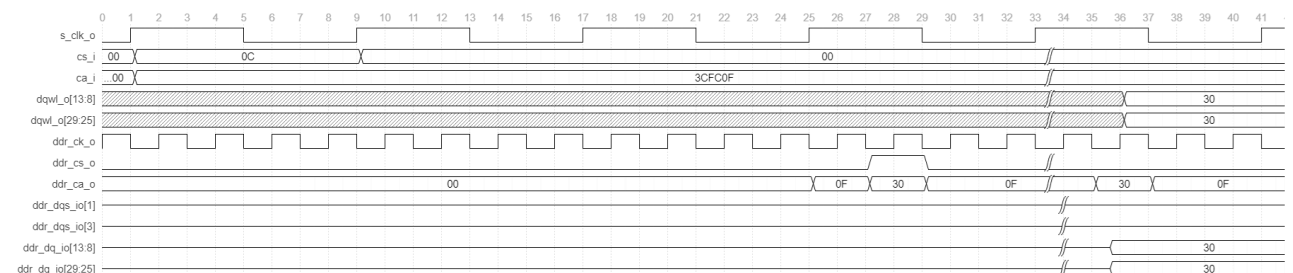


Figure 2.27. Sending/Capture a CA Pattern Timing Diagram

3. Core Generation, Simulation, and Validation

This section provides information on how to generate the LPDDR4 Memory Module using the Lattice Radiant software and how to run simulation and synthesis.

3.1. Licensing the IP

No license is required for this module.

3.2. Generation and Synthesis

Lattice Radiant Software allows you to generate and customize modules and IPs and integrate them into the device architecture.

To generate the LPDDR4 Memory Module in Lattice Radiant Software:

1. In the **Module/IP Block Wizard** create a new Lattice Radiant Software project for the LPDDR4 Memory module.
2. In the **IP Catalog** tab, double-click on **LPDDR4_MEM** under **Module, Architecture_Modules, I/O** category. The **Module/IP Block Wizard** opens as shown in [Figure 3.1](#). Enter values in the **Component name** and the **Create in** fields and click **Next**.

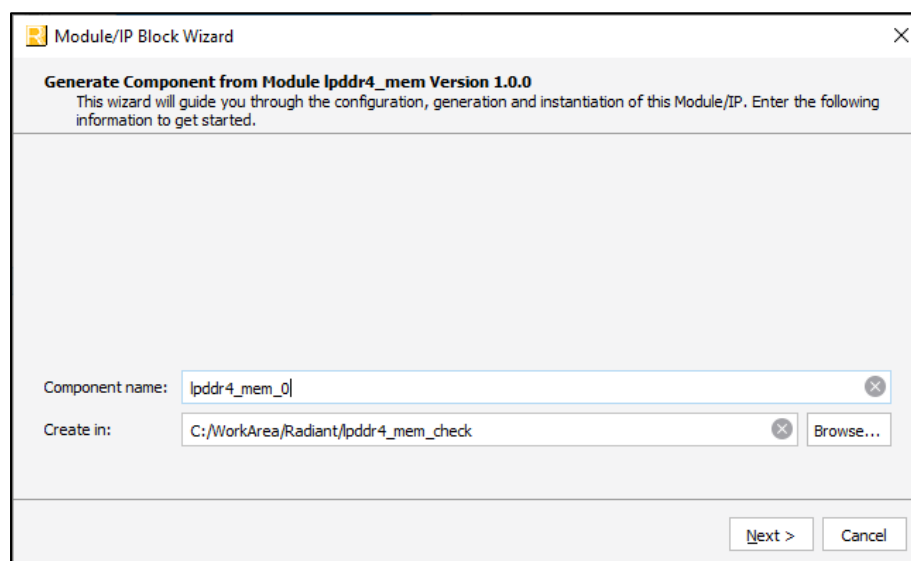


Figure 3.1. Module/IP Block Wizard

3. In the dialog box of the **Module/IP Block Wizard** window, configure LPDDR4 Memory module according to custom specifications using drop-down menus and check boxes. As a sample configuration, see [Figure 3.2](#). For configuration options, see [Table 2.2](#).

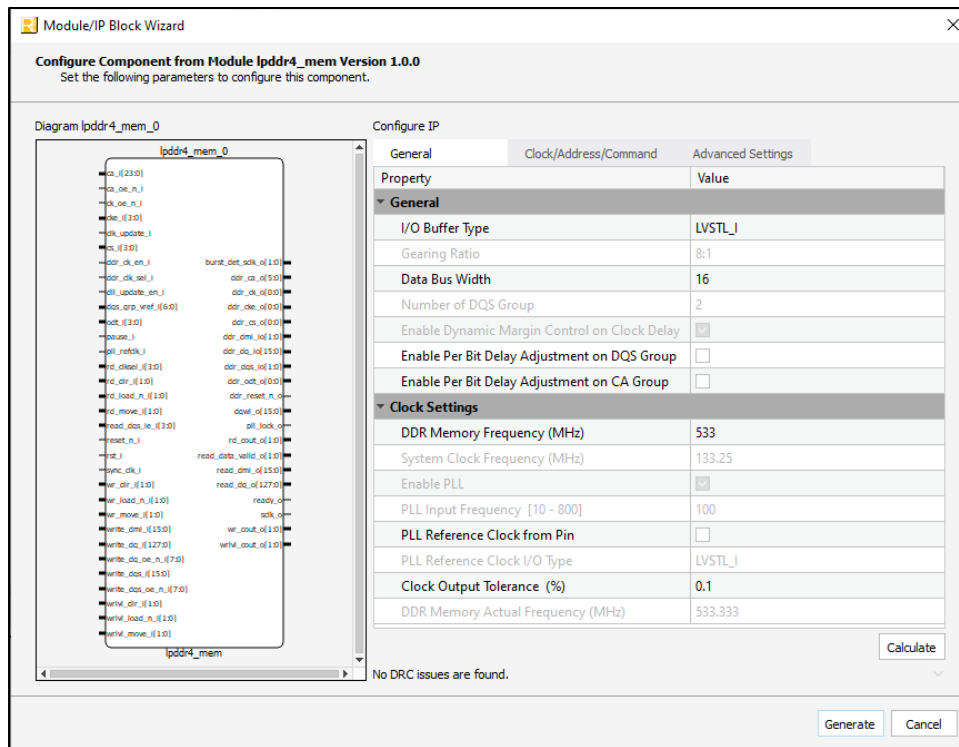


Figure 3.2. Configure Block of DDR Memory Module

- Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 3.3.

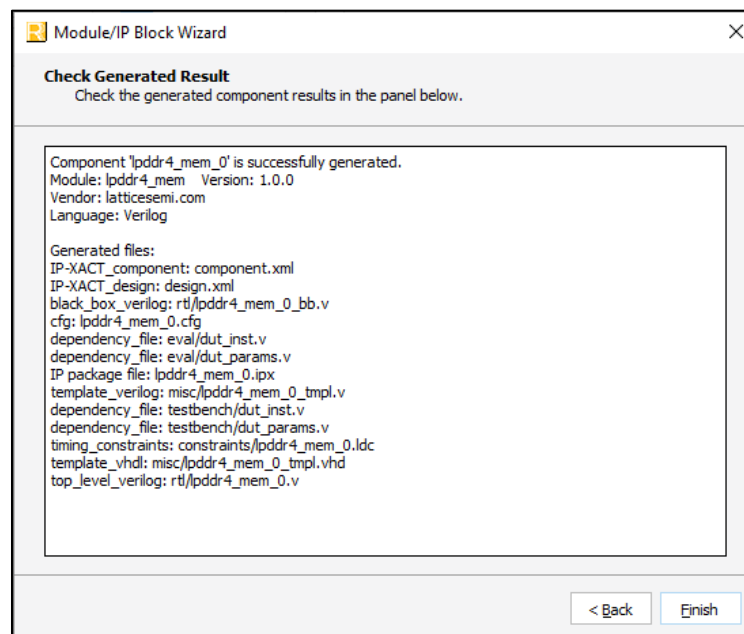


Figure 3.3. Check Generating Result

- Click **Finish** to generate the Verilog file. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 3.1.


The generated LPDDR4 Memory Module package includes the black box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.v) that can be used as an instantiation template for the IP core is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design. The generated files are listed in [Table 3.1](#).

Table 3.1. Generated File List

Attribute	Description
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the IP core.
rtl/<Component name>_bb.v	This file provides the synthesis black box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the IP core.
eval/eval_top.v	Top level RTL files that may be used for running Lattice Radiant software flow check (synthesis to export) on the generated IP. Without this, the Radiant software map process fails due to not having enough I/O. This is mainly used for checking resource utilization and fmax for the selected IP configuration, this is not for implementation.
eval/lscs_simple_lfsr.v	A simple linear-feedback shift register.
eval/dut_inst.v	A sample instantiation of the generated IP. This is included by the eval_top.v.
eval/dut_params.v	Lists the equivalent localparams of the user settings. This is included by the eval_top.v.

3.3. Running Functional Simulation

Below are the steps for running simulation.

1. Add the tb_top.v top level testbench file in the project as a simulation file. Click the **File** tab and select **Add** in the drop-down menu. Click **Existing Simulation File** and select <Component name>/testbench/tb_top.v.
2. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in [Figure 3.4](#).

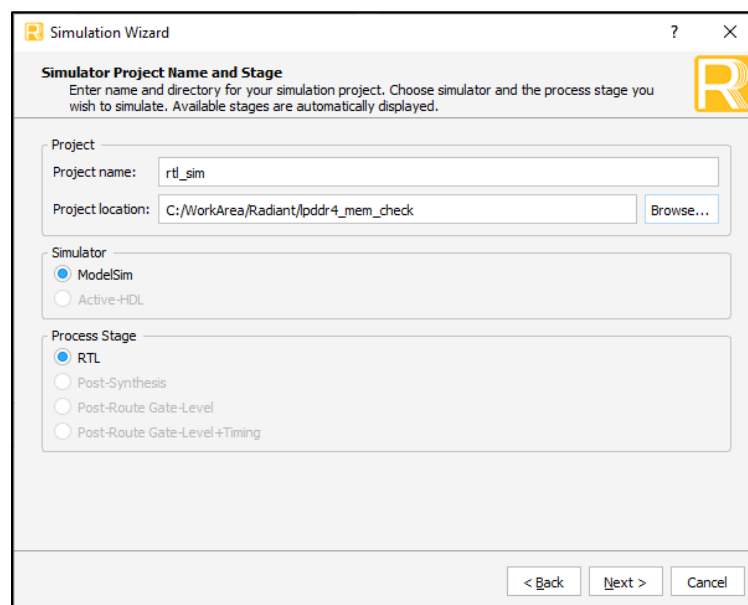


Figure 3.4. Simulation Wizard

- Click **Next** to open the **Add and Reorder Source** window as shown in Figure 3.5. Notice that the **Source Files** area only contain the generated IP (<Component name>.v) and the tb_top.v, which is added in Step 1. The tb_top.v includes all the necessary test files for simulation.

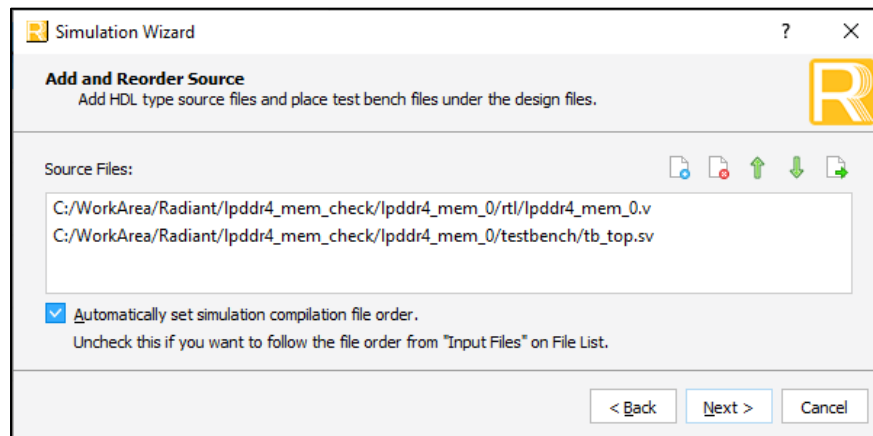


Figure 3.5. Adding and Reordering Source

- Click **Next**. The **Summary** window is shown. Click **Finish** to run the simulation.
The results of the simulation in our example are provided in Figure 3.6.

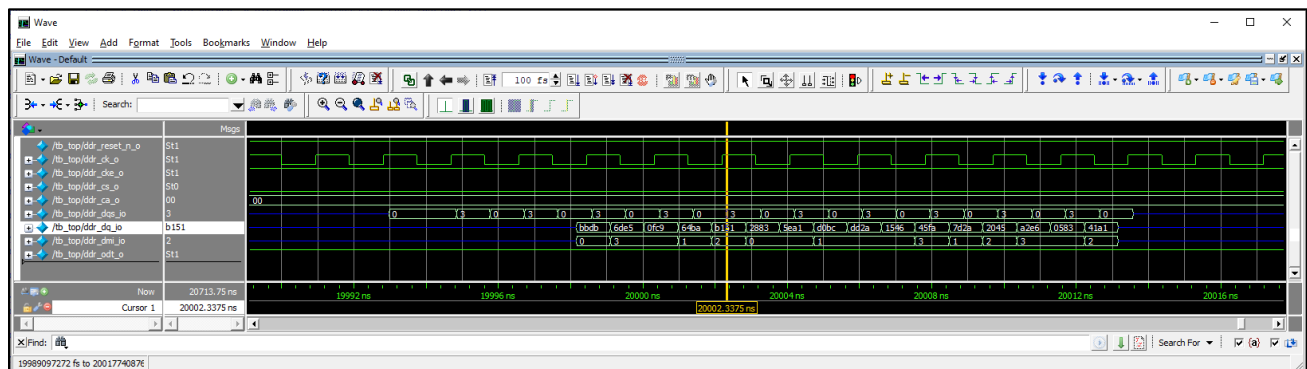


Figure 3.6. Simulation Waveform

Appendix A. Resource Utilization

Table A.1 shows the configuration and resource utilization for LCPNX-100-9FFG672I using Synplify Pro of Lattice Radiant software 2.3.

Table A.1. Resource Utilization

Configuration	sclk_o Fmax (MHz)*	Registers	LUTs	EBRs	IDDR/ODDR/TDDR
Default	200	49	118	0	18/30/20
Data Bus Width = 32, Others = Default	200	49	120	0	36/50/40
Data Bus Width = 64, Others = Default	200	49	118	0	72/90/80
Data Bus Width = 32, DDR Memory Frequency = 533 MHz, Others = Default	200	49	111	0	36/50/40

***Note:** The sclk_o Fmax is generated using a design that only contains the DDR Memory Module and a few linear-feedback shift registers. These values may be reduced when the IP Core is used with the user logic.

References

- [CertusPro-NX](#) web page
- [LPDDR4 Memory Interface Module IP Release Notes \(FPGA-RN-02103\)](#)
- [Lattice Radiant Software User Guide](#)
- [Lattice Propel Builder User Guide](#)
- [Lattice Radiant Timing Constraints Methodology User Guide \(FPGA-AN-02059\)](#)
- [Lattice Radiant](#) FPGA design software
- [Lattice Insights](#) for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, please refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Note: In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

Revision 1.2, IP v1.5.0, December 2025

Section	Change Summary
All	Added IP ver.1.5.0.
Abbreviation in This Document	<ul style="list-style-type: none"> Replaced <i>Acronyms</i> with <i>Abbreviations</i>. Updated section contents.
Introduction	<ul style="list-style-type: none"> Updated Table 1.1. Quick Facts. <ul style="list-style-type: none"> Added <i>MachXO5-NX</i>, <i>LFCPNX-50</i>, <i>LFCPNX-100AUTODIE</i>, <i>LFMXO5-55T</i>, and <i>LFMXO5-100T</i> under the support devices row. Added <i>IP Release Notes</i> reference. Updated IP version information. Added table notes 1 and 2.
Revision History	Added the note, <i>In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.</i>

Revision 1.1, June 2021

Section	Change Summary
Introduction	Updated content including Table 1.1 to include CertusPro-NX.
References	Updated reference to CertusPro-NX.

Revision 1.0, December 2020

Section	Change Summary
All	Initial release



www.latticesemi.com