

Logos2 DDR IP Application Guide

(AN04018, V1.1)

(20.08.2024)

Shenzhen Pango Microsystems Co., Ltd.

All Rights Reserved. Any infringement will be subject to legal action.

Revisions History

Document Revisions

Version	Date of Release	Revisions
V1.1	20.08.2024	Initial release.

Application Example for Reference Only

About this Manual

Terms and Abbreviations

Terms and Abbreviations	Meaning
APB	Advanced Peripheral Bus
AXI	AdvancedeXtensible Interface
CA	Control and Address
DDR	Double Data Rate
DFI	DDR PHY Interface
DCD	DDR Command Decode
DDC	Dedicated DQS Circuit
DCP	DDR Command Procedure
HMIC	High performance Memory Interface Controller
IPC	IP Compiler
LP	Low Power
MC	Memory Controller
MR	Mode Register
MRS	Mode Register Set
PCE	Physical Constraint Editor
PDS	Pango Design Suite
PHY	Physical
RD	Read
UCE	User Constraint Editor
UI	User Interface
WR	Write

Tables of Contents

Revisions History	1
About this Manual	2
Tables of Contents.....	3
Tables	4
Figures	5
Chapter 1 Overview.....	6
1.1 IP Application Introduction	6
Chapter 2 Function Description	7
2.1 IP Generation Process.....	7
2.1.1 Installing IP	7
2.1.2 Selecting IP	8
2.1.3 IP Parameter Configuration.....	8
2.1.4 Generating IP	19
2.1.5 GTP Hard Core Unit Constraints	19
2.1.6 Timing Constraint.....	20
2.2 IP Debug Interface.....	21
2.2.1 Debugging Signals	21
2.2.2 Description of Debug Signals	23
2.3 Typical Applications	27
2.3.1 Single BANK-x16 Typical Application.....	27
2.3.2 Multi BANK-x64 Typical Application.....	28
Chapter 3 Application Debugging FAQ.....	29
3.1 Debugging Tools	29
3.1.1 Debugger GUI Interface.....	29
3.1.2 Example Design	30
3.1.3 Debug Signals	34
3.1.4 Test Boards.....	34
3.2 Debugging Methods	34
3.2.1 Debug Using Example Design	34
3.2.2 Common Software Issues.....	35
3.2.3 General Inspection	36
3.2.4 Calibration Issues	40
3.2.5 Bit Error Issues.....	53
Disclaimer.....	58

Tables

Table 1-1 Logos2 Family DDR Application Summary	6
Table 2-1 Debug Interface	21
Table 2-2 Definitions of debug_data Bits	23
Table 2-3 Definition of dbg_calib_ctrl Bits	24
Table 2-4 Definition of dbg_slice_status Bits	25
Table 2-5 Definitions of dbg_slice_state Bits	26
Table 3-1 Bist Mode Control Signals	32
Table 3-2 Bist Mode Indicator Signals	33
Table 3-3 Master Status Information during PHY Initialization Calibration	43
Table 3-4 PHY Initialization-Related Signals	43
Table 3-5 DQS Gate Manual Adjustment Interface Signal	51

Figures

Figure 2-1 Opening [Update IP]	8
Figure 2-2 Adding IP	9
Figure 2-3 Logos2 HMIC_S IP Parameter Configuration Interface	9
Figure 2-4 Basic Options Page	10
Diagram 2-5 Structure Selection	11
Diagram 2-6 AXI Interface Mode Selection	11
Diagram 2-7 Memory Address Mapping Selection	11
Diagram 2-8 Total DQ Width Selection	12
Diagram 2-9 Input Clock and Data Rate Selection	12
Diagram 2-10 Read/Write Latency Selection	13
Figure 2-11 Memory Options Page	13
Figure 2-12 Memory Chip Selection	13
Figure 2-13 Reference Model Selection	14
Figure 2-14 SDRAM Customisation	15
Figure 2-15 Driver Options Supported by DDR3	15
Figure 2-16 Pin/Bank Options Page	16
Figure 2-17 Importing fdc File	16
Figure 2-18 Address and Control Bus Configuration	17
Figure 2-19 Data Bus Configuration	18
Figure 2-20 Summary Page	19
Figure 2-21 Logos2 HMIC_S IP Generation Report Interface	19
Figure 2-22 Single BANK-x16 Structure Diagram	27
Figure 2-23 Multi-bank-x64 Structure Diagram	28
Figure 3-1 Debug Signal Debugging Waveform	29
Figure 3-2 Example Design System Block Diagram	30
Figure 3-3 Example Design Test Flowchart	31
Figure 3-4 PHY Initialization Calibration Process	42
Figure 3-5 Master Status Information	44
Figure 3-6 Training Calibration Values of 4 Groups	45
Figure 3-7 Phase Relationship of CK and DQS Before Write Leveling Calibration	46
Figure 3-8 Phase Relationship Between CK and DQS at the SDRAM Pin	46
Figure 3-9 Write Leveling Mode Calibration Diagram	47
Figure 3-10 DQS Delay Greater than CK Delay	48
Figure 3-11 Read DQS Gate Training Diagram	50
Figure 3-12 Read Calibration State Value	52
Figure 3-13 Eye Calibration State Value	53
Figure 3-14 Error Code Example	54

Chapter 1 Overview

This chapter describes the DDR-related application information of the Logos2 Family FPGAs to help users understand the IP specifications and parameters, primarily including IP application introduction, DDR application block diagram, IP specification description, and other contents.

1.1 IP Application Introduction

The DDR3 IP—HMIC_S (High-performance Memory Interface Controller Soft core) IP based on the Logos2 Family FPGA devices is a soft core that combines the MC controller with physical layer (PHY) interfaces, providing users with simplified AXI4, standard AXI4, and DFI3.1 interfaces for DDR3 SDRAM high-speed system designs.

[Table 1-1](#) briefs the DDR features and application information for the Logos2 family.

Table 1-1 Logos2 Family DDR Application Summary

Core Features	
Supported Devices	PG2L25H, PG2L50H, PG2L100H, PG2L100HX, and PG2L200H
Scenario Coverage	HR BANK supports DDR3.
Maximum Speed	Refer to " <i>DS04001 Logos2 Family FPGA Device Datasheet</i> "
Available Resources	
IP Design Files	DDR3 IP Core (RTL)
Example Design	User Design (Verilog)
Simulation	Test Bench(systemverilog) Simulation Library(RTL) DDR3 systemverilog models
Constraint File	FDC include timing and position constrains
Supported Versions	
PDS	For the PDS version used with the IP, refer to " <i>UG042003_HMIC_S_IP</i> "
HMIC_S IP	DDR3 IP 1.14 for HR BANK
Modelsim	ModelSim 10.2c or above/QuartaSim 10.6c or above
VCS	VCS 2018.09-SP2
Verdi	Verdi 2018.09-SP2

Note: Besides the Logos2 Family devices, this IP also supports other devices using HRIO DDR IPs: PG2K400, PG2T70H, PG2T390HX, and PG2T160H.

Chapter 2 Function Description

2.1 IP Generation Process

This section provides a step-by-step guide to generate the IP for users and detailedly explains all the parameters required to be defined in this process. Upon completing all the steps, users will obtain the RTL codes and constraint files that are consistent with the memory interface design, as well as the corresponding Example Design and the necessary scripts for simulation, which can be used for reference and debugging.

Attention:

The screenshots in this section are for reference only, and the specific interface may vary depending on the software version.

Click the menu item [Tools > IP Compiler] or directly click the [IP Compiler] button in the toolbar to launch the IPC tool. Customized configurations of HMIC_S IP can be completed through the IPC tool, instantiating the required IP modules. For detailed instructions on using the IPC tool, please refer to "*IP_Compiler_User_Guide*".

The main steps for instantiating the HMIC_S IP module are described below.

2.1.1 Installing IP

Start by opening the IPC tool in a new or existing project. Click [File > Update] in the main window to open the [Update IP] dialogue box, and click [Add Package] in the top-left corner, select the .jar file to add the corresponding version of the IP.

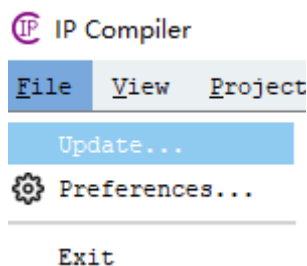


Figure 2-1 Opening [Update IP]

IPs that have never been installed before will be selected by default. Click [Install] to install. If not selected, it means that an IP with the same version number (but possibly not identical) has already been installed. Users can manually tick the checkbox to override the previous IP.

Upon successful IP installation, [Installed 1 IP] will appear in the [Message] dialogue box of the IPC tool. After completion, all the installed IPs will display in the [Catalog] interface on the left side.

2.1.2 Selecting IP

In the IPC interface, select [DDR3 Interface] under the "System/DDR/Soft" directory in the left window, then set the [Pathname] and [Instance Name] on the right page. The default path for generating the IP is "/ipcore/<Instance Name>/<Instance Name>.idf" under the current project directory.

2.1.3 IP Parameter Configuration

After selecting the IP, click [Customize] to enter the Logos2 HMIC_S IP parameter configuration interface. The [Add IP] dialogue box pops up. Click [Yes] to add the generated IP to the project. If selecting [No], the IP is still generated but not associated with the project, then users need to manually add the idf file of the existing IP to use it.

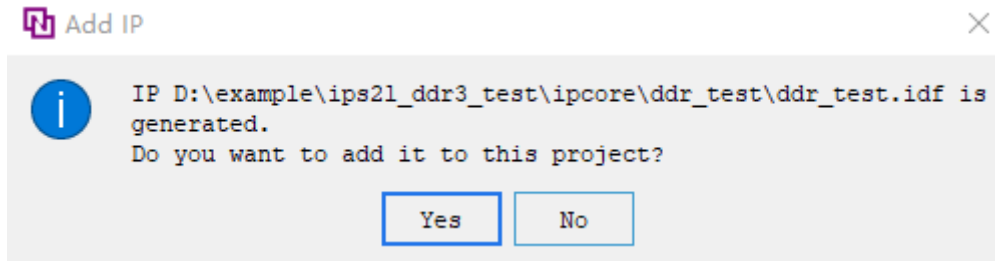


Figure 2-2 Adding IP

The [Symbol] window on the left of the [Configure] interface shows the interface block diagram, and the parameter configuration window is on the right.

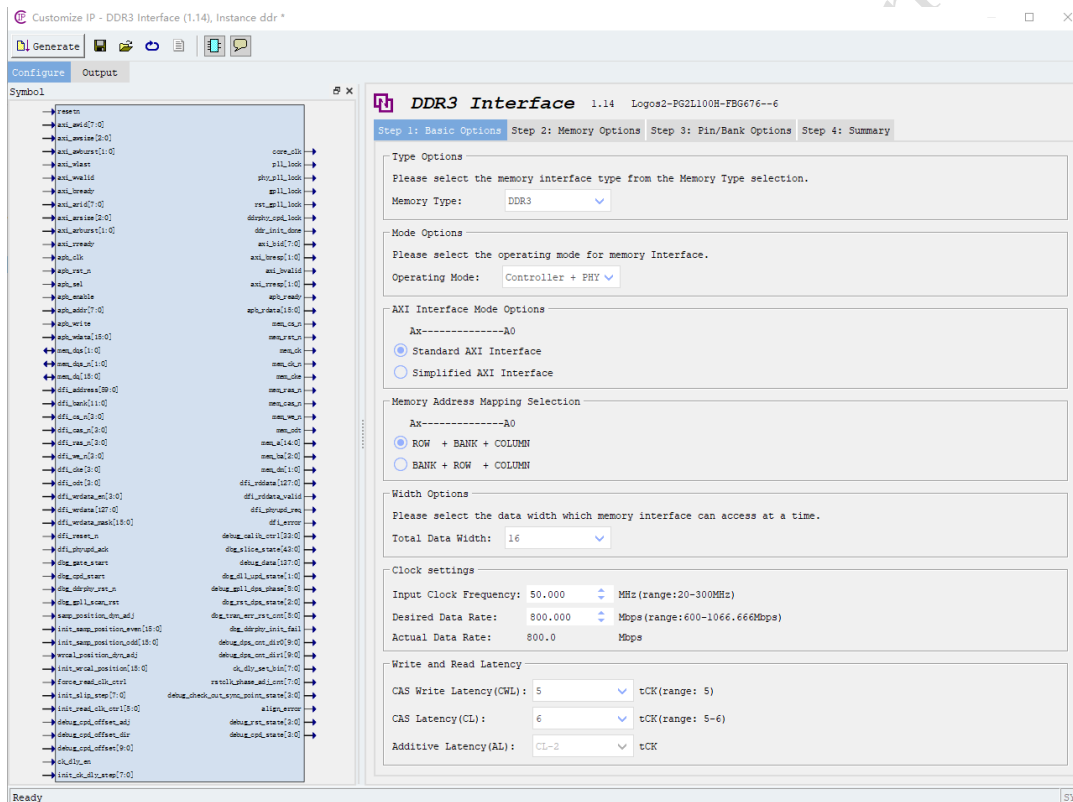


Figure 2-3 Logos2 HMIC_S IP Parameter Configuration Interface

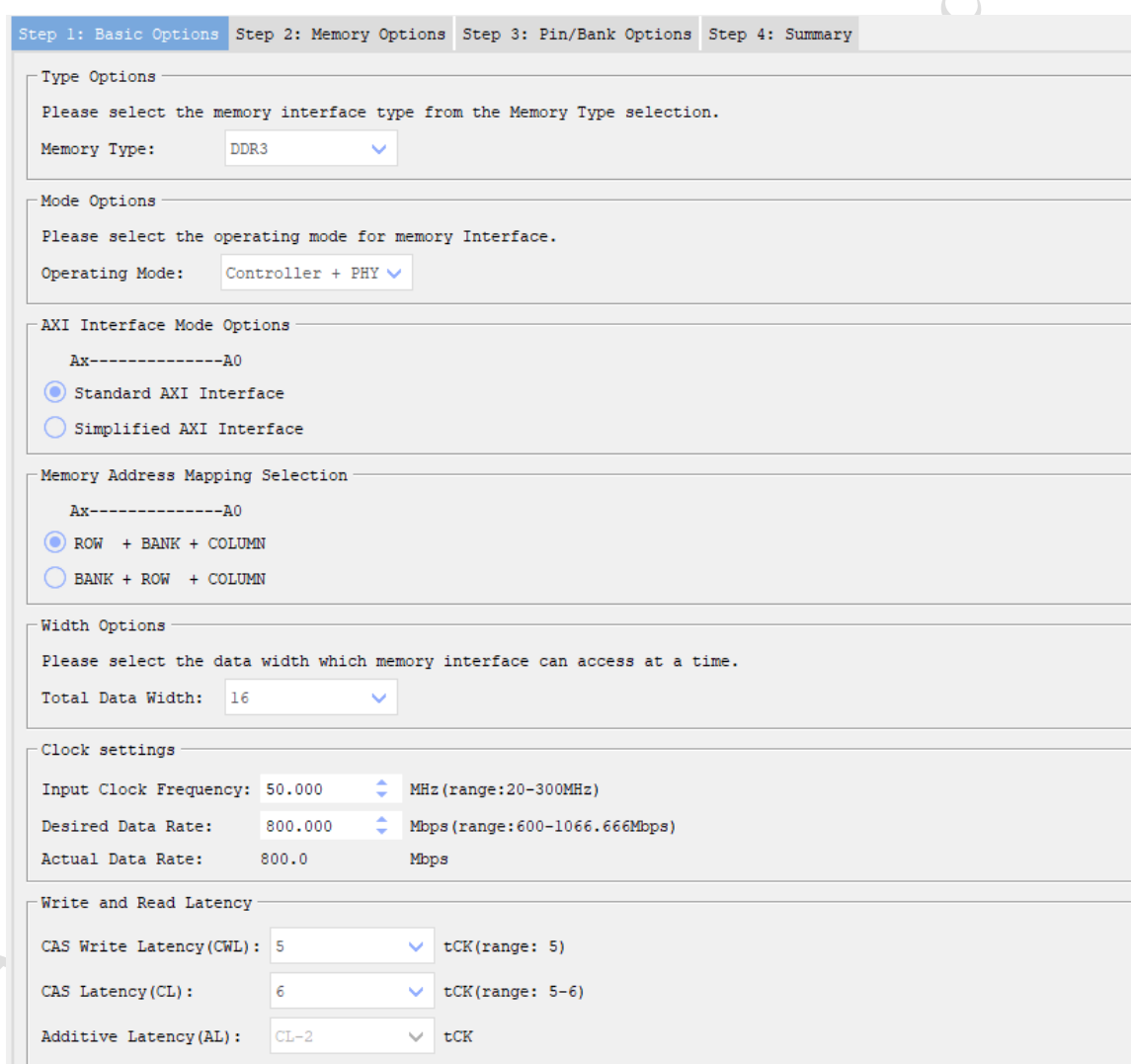
Parameter configuration includes four pages, namely Step1: Basic Options, Step 2: Memory Options, Step 3: Pin/Bank Options, Step 4: Summary. The configuration steps for HMIC_S IP are described as follows.

Description:

Please configure the IP parameters in the order of the pages, following Step 1 → Step 2 → Step 3 → Step 4.

Step 1: Basic Options

[Basic Options] is the basic configuration page for the IP that allows defining the basic parameters of the HMIC_S IP, including memory types selection, structure selection, AXI mode selection, memory address mapping method selection, data width selection, clock settings, and read/write latency.



Step 1: Basic Options | Step 2: Memory Options | Step 3: Pin/Bank Options | Step 4: Summary

Type Options
Please select the memory interface type from the Memory Type selection.
Memory Type:

Mode Options
Please select the operating mode for memory Interface.
Operating Mode:

AXI Interface Mode Options
Ax-----A0
☒ Standard AXI Interface
☐ Simplified AXI Interface

Memory Address Mapping Selection
Ax-----A0
☒ ROW + BANK + COLUMN
☐ BANK + ROW + COLUMN

Width Options
Please select the data width which memory interface can access at a time.
Total Data Width:

Clock settings
Input Clock Frequency: MHz (range:20-300MHz)
Desired Data Rate: Mbps (range:600-1066.666Mbps)
Actual Data Rate: Mbps

Write and Read Latency
CAS Write Latency(CWL): tCK (range: 5)
CAS Latency(CL): tCK (range: 5-6)
Additive Latency(AL): tCK

Figure 2-4 Basic Options Page

Select the memory type for HMIC_S IP from the [Memory Type] pull-down menu, with DDR3

supported currently.

Select the structure for the HMIC_S IP from the [Operating Mode] pull-down menu. If selecting [Controller + PHY], the generated IP codes incorporate the controller and PHY, while only the PHY is included for the [PHY Only] option.

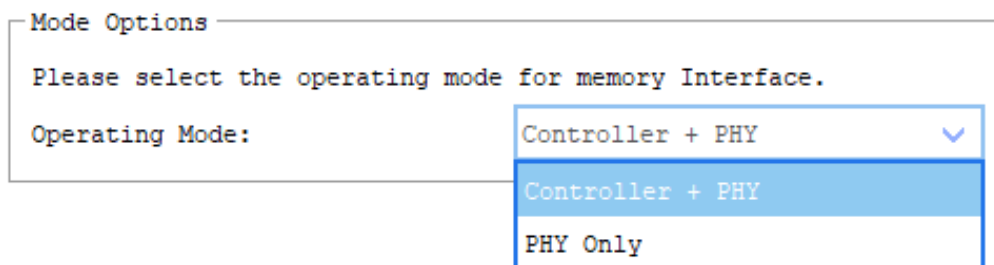


Diagram 2-5 Structure Selection

In Controller + PHY mode, users can choose between the [Simplified AXI interface] and the [Standard AXI interface]. In PHY Only mode, this selection is not required.

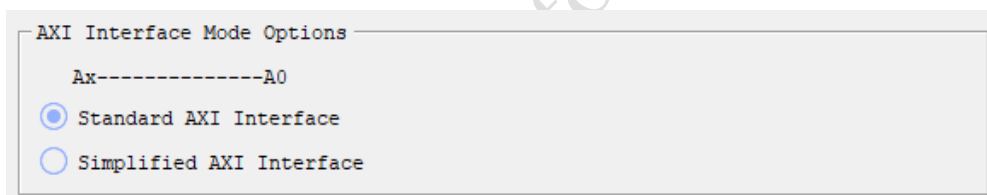


Diagram 2-6 AXI Interface Mode Selection

In Controller + PHY mode, users can select between the two read/write address mapping methods of 'ROW + BANK + COLUMN' and 'BANK + ROW + COLUMN' for the Controller AXI interface. In PHY Only mode, this selection is not required.

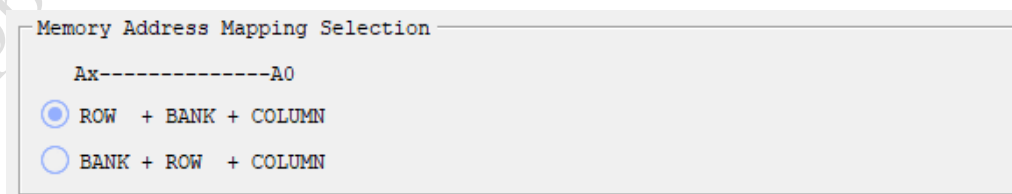


Diagram 2-7 Memory Address Mapping Selection

From the [Total Data Width] pull-down menu, select the total DQ width of the off-chip SDRAM

connected to HMIC_S (with a maximum bit width of 72 bits).

Width Options

Please select the data width which memory interface can access at a time.

Total Data Width:

16

72

64

56

48

40

32

24

16

8

Diagram 2-8 Total DQ Width Selection

Users can directly enter the input clock frequency and desired data rate of HMIC_S IP in the numeric setting box or adjust the values using the arrow. The input values must be within the supported range, and the currently supported maximum data rate is 1066Mbps for DDR3. The actual data rate is calculated by the software, which should be as close to the desired rate as possible.

Clock settings

Input Clock Frequency: 50.000 MHz (range:20-300MHz)

Desired Data Rate: 800.000 Mbps (range:600-1066.666Mbps)

Actual Data Rate: 800.0 Mbps

Diagram 2-9 Input Clock and Data Rate Selection

The read/write latency length varies based on the selected memory data rate, which can be changed using the pull-down menu. The default values are recommended for optimal performance.

Write and Read Latency

CAS Write Latency(CWL):

5

▼

tCK(range: 5)

CAS Latency(CL):

6

▼

tCK(range: 5-6)

Additive Latency(AL):

CL-2

▼

tCK

Diagram 2-10 Read/Write Latency Selection

Step 2: Memory Options

The [Memory Options] page is dedicated to memory configuration, which includes memory chip selection and customisation as well as options for the memory device driver.

Step 1: Basic Options

Step 2: Memory Options

Step 3: Pin/Bank Options

Step 4: Summary

Memory Part

Please select the memory part. Find an equivalent part or create a part using the 'Create Custom Part' button if the part you want is not listed here.

☐ Create Custom Part

MT41K128M16XX

▼

Drive Options

To calibrate the output driver impedance, an external precision resistor (RZQ) is connected between the ZQ ball and VSSQ. The value of the resistor must be 240ohm +/-1 percent.

Output Driver Impedance Control: RZQ/6

▼

The ODT feature is designed to improve signal integrity of the memory channel by enabling the DDR3 SDRAM controller to independently turn on/off ODT.

RTT(nominal)-ODT: RZQ/4

▼

Figure 2-11 Memory Options Page

Select the memory chip model used from the pull-down menu. The supported memory chips depend on the selected memory type. [Figure 2-12](#) shows the optional memory chips of DDR3 SDRAM.

Memory Part

Please select the memory part. Find an equivalent part or create a part using the 'Create

☐ Create Custom Part

MT41K128M16XX

▼

MT41K128M8XX

MT41K64M16XX

MT41K256M8XX

MT41K128M16XX

MT41K512M8XX

MT41K256M16XX

Figure 2-12 Memory Chip Selection

If the optional memory chips do not meet the requirements, users can check [Create Custom Part], select a reference model from the pull-down menu of the [Custom Memory Part] option box below, and then customise a new SDRAM type based on the reference model. The timing parameters and the row, column, and bank address bits can be modified as needed. Likewise, the reference models are determined by the selected memory type.

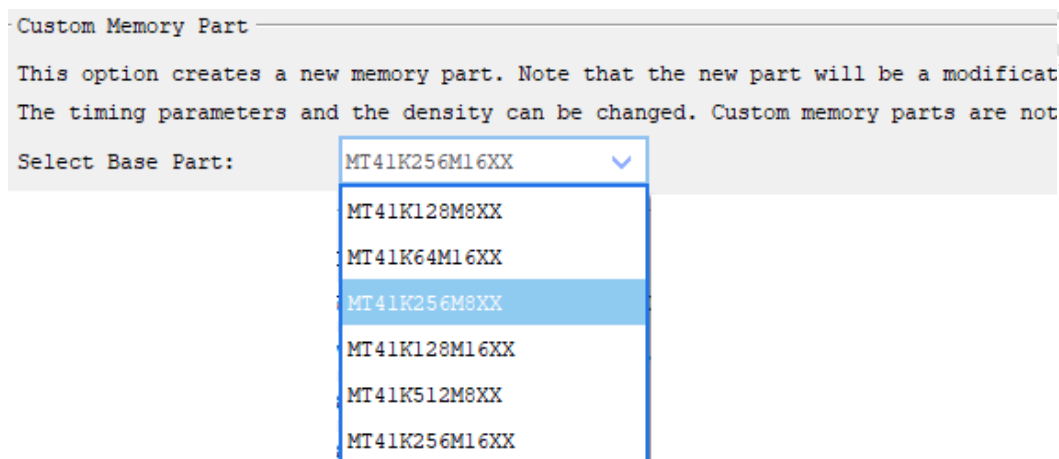


Figure 2-13 Reference Model Selection

The Hynix HMT325S6BFR8C-G7 memory chip is taken as an example to explain how to customise SDRAM. According to the memory datasheet, the Hynix HMT325S6BFR8C-G7 is a DDR3 SDRAM memory chip with a density of 2Gbit, a maximum data rate of 1066Mbps, 64-bit width, row address bit A0-A14, column address bit A0-A9, and bank address bit BA0-BA2. Therefore, the Micron MT41K256M8XX is selected as the reference model, set as 2Gbit memory density, 160ns tRFC, 37.5ns tRAS, 15-bit row address, 10-bit column address, and 3-bit bank address, which are the same as the Hynix HMT325S6BFR8C. The Hynix HMT325S6BFR8C-G7 has a maximum data rate of 1066Mbps, and the setting is modified as 13.125ns tRCD and 13.125ns tRP based on the memory datasheet. The tREFI is set to 7.8ns under normal operating conditions but should be halved to 3.9ns for high-temperature operations (85°C<T<120°C). Now the customisation of the SDRAM is completed.

Custom Memory Part

This option creates a new memory part. Note that the new part will be a modification of the 'Base Part' you select below. The timing parameters and the density can be changed. Custom memory parts are not simulated and not hardware validated.

Select Base Part:

Parameter	Value	Range	Units	Descriptions
trfc	160.000	90-350	ns	Refresh to Active or Refresh to Refresh
tras	37.500	35-37.5	ns	Active to Precharge command
trp	13.125	10-15	ns	Precharge command period
tred	13.125	10-15	ns	Active to Read or Write delay
twr	15.000	15-15	ns	Write recovery time
trefi	7.800	1-7.8	us	Average periodic refresh interval
trtp	7.500	7.5-7.5	ns	Internal Read to Precharge command delay
twtr	7.500	7.5-7.5	ns	Read following a Write to the same device

Row Address:

Column Address:

Bank Address:

Figure 2-14 SDRAM Customisation

The driver options are determined by the selected memory type and will be applied to the internal mode register during memory initialization. Users can choose the driver and ODT values based on their needs. The recommended default values are $DRV = RZQ/6$ and $ODT = RZQ/4$.

For the detailed definitions and address values of the relevant register, please refer to *JESD79-3D*, *DDR3 SDRAM Standard* and "*UG042003_HMIC_S_IP*".

Drive Options

To calibrate the output driver impedance, an external precision resistor (RZQ) is connected between the ZQ ball and VSSQ. The value of the resistor must be 240ohm +/-1 percent.

Output Driver Impedance Control:

The ODT feature is designed to improve signal integrity of the memory channel by enabling the DDR3 SDRAM controller to independently turn on/off ODT.

RTI (nominal)-ODT:

Figure 2-15 Driver Options Supported by DDR3

Step 3: Pin/Bank Options

The [Pin/Bank Options] page is the configuration page for interface parameters.

Step 1: Basic Options Step 2: Memory Options **Step 3: Pin/Bank Options** Step 4: Summary

Memory Pin Constraint File Select
Please select a fdc file which contains default memory pins constraint.
☐ Enable fdc file select

PLL Reference Clock Pin Options
Please select the banks for the PLL Reference Clock in the architectural view below.
PLL Reference Clock Bank: L5

Control/Address Pin Options
Please select the banks for the Control/Address in the architectural view below.
Control/Address Bank: L5

Please select the pins for the Control/Address in the architectural view below.
☒ Enable CS_n(if cs_n is disabled,it should be considered NF maintained LOW through an external resister to GND)

Please select the groups for the Control/Address in the architectural view below.
☐ Custom Control/Address Group

Data Pin Options
Please select the banks and groups for the data in the architectural view below.

Signal Name	Bank Number	Group Number
DQ[0-7]	L6	G1
DQ[8-15]	L6	G0

Figure 2-16 Pin/Bank Options Page

Check [Enable fdc file select] to import the custom fdc file. Users can input the fdc file path in the text box or click [Browse] to select the fdc file, then the IP will automatically read the memory interface constraints from the fdc file and configure the Control/Address and Data Pins in the UI.

Memory Pin Constraint File Select
Please select a fdc file which contains default memory pins constraint.
☒ Enable fdc file select

Please select a fdc file

Figure 2-17 Importing fdc File

Attention:

The memory interface signal name in the user's fdc file must match that in the Example Design.

Select the bank where the PLL reference clock is located from the [PLL Reference Clock Bank] pull-down menu. The bank tables vary depending on the selected FPGA chip. The PLL reference

clock must be placed in the same bank as the input reference clock.

From the [Control/Address Bank] pull-down menu, select the bank and group where the address and control bus are located. The mem_cs_n signal can be enabled by checking [Enable CS_n]. The default grouping is used when [Custom Control/Address Group] is unchecked. Due to different PCB traces in actual use, users are advised to check [Custom Control/Address Group] for using custom pins. The default pin allocation is based on the P04I100KF01_A2 estimation board. Users can refer to the P04I100KF01_A2 schematic diagram for hardware design.

When configuring pins using custom groups, the GUI will perform a legality check on the pins. If two ports use the same pin, the port pins will be highlighted in red as a warning to the user.

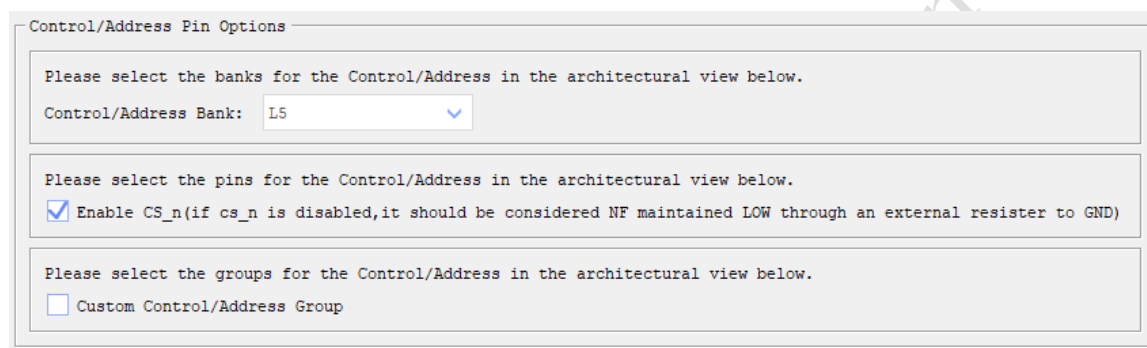


Figure 2-18 Address and Control Bus Configuration

Attention:

Set the location constraints of the DDR address and control bus IO using the pin number during custom grouping via the IP configuration page. The flow test may fail if the LOC position constraints of the DDR address and control bus IO are modified via fdc or the UCE interface.

Select the bank and group where the data bus is located from the pull-down menu. The total length of the DQ is consistent with the data width selected in "[Step 1: Basic Options](#)". Due to different PCB traces in actual use, users still need to constrain the DQ pins according to the actual traces after generating the IP.

Data Pin Options

Please select the banks and groups for the data in the architectural view below.

Signal Name	Bank Number	Group Number
DQ[0-7]	L6	G1
DQ[8-15]	L6	G0

Figure 2-19 Data Bus Configuration

Attention:

All configuration items in "Step 3: Pin/Bank Options" must be configured according to the actual pin assignments on the circuit board. After generating the IP, the DQ pins, reset pins and status pins should be constrained according to actual pin assignments on the circuit board; otherwise, errors may occur when running the flow test.

The PLL reference clock must be placed in the same bank as the input reference clock. To minimize the impact of noises on signal quality, it is recommended to place the control/address and the reference clock in the same bank.

A maximum of 3 consecutive banks on the same side is allowed while cross-bank utilization is prohibited.

Step 4: Summary

The [Summary] page is used to print the current configuration information without parameter configuration required. Users can review their selections here.

Basic Options	
Memory Type	: DDR3
Operating Mode	: Controller + PHY
Total Data Width	: 16
Density	: 4Gb
Volt	: 1.5V
Input Clock Frequency	: 50.0MHz
Data Rate	: 800.0Mbps

Memory Options	
Memory Part	: MT41K256M16XX
Row Address	: 15
Column Address	: 10
Bank Address	: 3
Output Driver Impedance Control	: RZQ/6
RIT(nominal)-ODT	: RZQ/4

Pin/Bank Options	
PLL Reference Clock Bank	: L5
Control/Address Bank	: L5
CS_n	: Enabled
RESET	: L5
DQ[0-7] Bank	: L6
DQ[8-15] Bank	: L6

Figure 2-20 Summary Page

2.1.4 Generating IP

Upon completion of parameter configuration, click the [Generate] button in the top left corner to generate the IP, in order to generate HMIC_S IP codes according to users' specific requirements. The information report interface for IP generation is shown in [Figure 2-21](#).

Done: 0 error(s), 0 warning(s)

Figure 2-21 Logos2 HMIC_S IP Generation Report Interface

2.1.5 GTP Hard Core Unit Constraints

Certain GTP hard core units in DDR projects require physical constraints. Position constraints for some GTP hard core units (e.g. CPD) need to be consistent with the location constraints provided by the .fdc file, the constraint location cannot be changed, and the traces should be modified

according to specific usage. Other GTP hard core units can be directly edited through fdc, or manually constrained using the PCE (Physical Constraint Editor) tool. The constraint information is stored in the .fdc file in the "/pnr" directory and will ultimately be transferred to the pcf constraint file during the placement. For the specific configuration method of constraint files, please refer to the relevant help documents in the PDS installation path: "*User_Constraint_Editor_User_Guide*", "*Physical_Constraint_Editor_User_Guide*".

For example: `define_attribute {i:I_ips_ddr_top.u_ddrphy_top.u_ddrphy_cpd} {PAP_LOC} {DDRPHY_CPD_9_466}`

2.1.6 Timing Constraint

Multiple clocks within the HMIC_S IP require constraints, namely ref_clk, rst_clk, ddrphy_sysclk, and phy_dq_sysclk (one constraint required for each bank). For specific constraint methods, please refer to the .fdc files in the "IP /pnr" directory.

In timing constraints, the create_clock command defines the master clock from outside of the FPGA chip, and the create_generated_clock command is used to define generated clocks.

The set_multicycle_path command is used to apply multi-cycle path constraints to adjust the positional relationship between the source clock edge and the target clock edge in timing analysis.

The set_clock_uncertainty command defines clock uncertainty to represent clock jitter and clock skew, thus increasing system tolerance.

The set_clock_groups command is used to set clock groups so that the software does not analyse timing paths between different clock groups.

For specific meanings of parameters in timing constraint commands, please refer to "*Pango_Design_Suite_User_Guide*".

Attention:

The timing constraints for `ddrphy_sysclk` and `phy_dq_sysclk` provided in the `.fdc` file are by default supported only by the ads synthesis tool. To use the OEM synthesis tool, the constraints must be switched to the OEM format constraints stated in the `.fdc` file.

2.2 IP Debug Interface

2.2.1 Debugging Signals

HMIC_S IP provides a rich set of debug interfaces. The input port can be debugged by modifying the register through the serial port or directly via manual input, and the output port can be debugged by printing information through the serial port or via JTAG. When the debug function is not used, the input end of the debug interface should be assigned to a fixed value according to the reference design and should not be left floating.

Description:

For detailed usage of the debug interface, please refer to "[Chapter 3 Application Debugging FAQ](#)"

Table 2-1 Debug Interface

Port Name	Direction	Bit width	Description
<code>dbg_gate_start</code>	Input	1	Reset sequence control, active on the rising edge.
<code>dbg_cpd_start</code>	Input	1	Reset sequence control, active on the rising edge.
<code>dbg_ddrphy_rst_n</code>	Input	1	Reset sequence control, active low.
<code>dbg_gpll_scan_rst</code>	Input	1	Reset sequence control, active high.
<code>samp_position_dyn_adj</code>	Input	1	dqsi and dqsin sampling position dynamic adjustment enable after the initialization is completed When a rising edge is detected, an offset is added to the current sampling position, and the amount of offset is determined by <code>init_samp_position_even</code> and <code>init_samp_position_odd</code> .
<code>init_samp_position_even</code>	Input	8*DQS_WIDTH	The offset step of dqsi initial sampling position relative to 90° delay. The highest bit indicates the sign bit, requiring the addition operation for the value of 0 and the subtraction operation for the value of 1. There are a total of DQS_WIDTH 8-bit groups that can be configured independently. It is also valid when dynamically adjusting the dqsi and dqsin sampling positions.

Port Name	Direction	Bit width	Description
init_samp_position_odd	Input	8*DQS_WIDTH	The offset step of dqsi initial sampling position relative to 90° delay. The highest bit indicates the sign bit, requiring the addition operation for the value of 0 and the subtraction operation for the value of 1. There are a total of DQS_WIDTH 8-bit groups that can be configured independently. It is also valid when dynamically adjusting the dqsi and dqsin sampling positions.
wrcal_position_dynamic	Input	1	Enables write direction DQS and DQ phase dynamic adjustment When a rising edge is detected, an offset is applied to the current phase, determined by init_wrcal_position.
init_wrcal_position	Input	MEM_DQS_WIDTH*8	The offset step of DQS and DQ phase initial position relative to 90°. The highest bit indicates the sign bit, requiring the addition operation for the value of 0 and the subtraction operation for the value of 1. There are a total of DQS_WIDTH 8-bit groups that can be configured independently. It is also active when dynamically adjusting the write direction DQS and DQ phases.
force_read_clk_ctrl	Input	1	Enables DQS gate position fixing 0: DQS gate position changes during the training process. 1: DQS gate position remains unchanged, always at the initial value
init_slip_step	Input	4*DQS_WIDTH	Initial value for DQS gate coarse adjustment position. There are a total of DQS_WIDTH 4-bit groups that can be configured independently.
init_read_clk_ctrl	Input	3*DQS_WIDTH	Initial value for DQS gate fine adjustment position. There are a total of DQS_WIDTH 3-bit groups that can be configured independently.
debug_cpd_offset_adj	Input	1	Enables GPLL phase adjustment, active on the rising edge. When a rising edge is detected internally, the GPLL phase is adjusted based on debug_cpd_offset_dir and debug_cpd_offset relative to the current GPLL phase.
debug_cpd_offset_dir	Input	1	GPLL phase adjustment direction 0: Decrease; 1: Increase;
debug_cpd_offset	Input	10	GPLL phase adjustment step.
init_ck_dly_step	Input	8	The initial step of CK when performing write leveling.
ck_dly_en	Input	1	Enable command and address signal output delay adjustment for the memory interface, active high. 0: The command and address signal output delay are generated by the training process. 1: The command and address signal output delay remains unchanged, always at the setting value of init_ck_dly_step.
ck_dly_set_bin	Output	8	The delay adjustment port for the CK signal output of the memory interface. The configuration value of this port multiplied by 5ps is the additional delay for CK.
debug_data	Output	69*DQS_WIDTH	Debug data for each set of DDRPHY with 8-bit DQ sharing a single DDRPHY.
dbg_calib_ctrl	Output	34	Debug data for Training status.
dbg_slice_status	Output	17*MEM_D	Training status.

Port Name	Direction	Bit width	Description
		QS_WIDTH	
dbg_slice_state	Output	22*MEM_D QS_WIDTH	Training status.
dbg_dll_upd_state	Output	2	DLL update control state.
debug_gpll_dps_phase	Output	9	GPLL current phase.
dbg_rst_dps_state	Output	3	Reset clock phase adjustment status.
dbg_tran_err_rst_cnt	Output	6	Reset sequence signal.
dbg_ddrphy_init_fail	Output	1	DDRPHY initialization failed.
debug_dps_cnt_dir0	Output	10	GPLL phase adjustment count.
debug_dps_cnt_dir1	Output	10	GPLL phase adjustment count.
align_error	Output	1	Standard for data alignment errors between different groups
debug_rst_state	Output	4	Reset sequence status.
debug_cpd_state	Output	4	CPD alignment state.

2.2.2 Description of Debug Signals

2.2.2.1 Description of debug_data

debug_data is the debug data for each set of DDRPHY with 8-bit DQ sharing a single DDRPHY and outputting debug data of 69 bits. The debug data from different DQ groups is concatenated into debug_data, with the lower DQ's debug data in the lower bits of debug_data. Users can perform debugging by printing information through the serial port or via JTAG. For the description of the meanings of each field of debug_data, please refer to [Table 2-2](#).

Table 2-2 Definitions of debug_data Bits

Bits	Debug Port	Description
3:0	coarse_slip_step	Coarse adjustment position obtained from DQS gate training, expressed in 1 CK clock cycle
6:4	read_clk_ctrl	Fine adjustment position obtained from DQS gate training, expressed in 1/8 CK clock cycle
10:7	gate_win_size	Size of the valid window for DQS gate training
11	gate_check_pass	DQS gate training completion indicator (set to 0 after completion)
12	rddata_check_pass	Read calibration data completion confirmation indicator (set to 0 after completion)
20:13	dqs_even_bin	DQS rising edge delay step value obtained after read eye diagram calibration, expressed in 10ps
28:21	dqs_odd_bin	DQS falling edge delay step value obtained after read eye diagram calibration, expressed in 10ps
36:29	total_margin_even	Margin for DQS rising edge window
45:38	total_margin_odd	Margin for DQS falling edge window
55:48	wrlvl_step	DQS delay step value obtained when write leveling is complete, expressed in 10ps

Bits	Debug Port	Description
56	Reserved	Reserved
57	wrlvl_dq	DQ value returned during write leveling
65:58	wl_p_dll_bin	Write leveling delay code plus 90-degree code
66	this_group_ca_dly	The number of mem_ck clock cycles that the CA signals of this group are delayed.
68:67	ck_dqs_diff	The difference between the mem_ck clock cycles for the CA and DQS signals of this group

2.2.2.2 Description of dbg_calib_ctrl

dbg_calib_ctrl is the state machine register during the PHY initialisation calibration process. Users can perform debugging by printing information through the serial port or via JTAG. For the description of dbg_calib_ctrl fields, please refer to [Table 2-3](#).

Table 2-3 Definition of dbg_calib_ctrl Bits

Bits	Debug Port	Description
0	calib_error	Training error indicator, generated due to read calibration errors during write leveling
4:1	dbg_upcal	Update control state machine
8:5	dbg_eyecal	Read eye calibration control state machine
12:9	dbg_wrcal	Write calibration control state machine
17:13	dbg_rdcap	MPR read calibration control state machine
21:18	dbg_wrlvl	write leveling calibration control state machine
25:22	dbg_init	Power-up initialization process control state machine
29:26	dbg_main	Master state machine for training
33:30	dbg_error_status	The training state when an error occurs. 0: No error; 1: Write leveling generates an error; 2: Read pattern generates an error; 3: Gate training generates an error; 4: Read training generates an error; 5: Write training generates an error; 6: Eye training generates an error.

2.2.2.3 dbg_slice_status Description

dbg_slice_status is the slice module status signal register for each set of DDRPHY with 8-bit DQ sharing a single DDRPHY and outputting a 17-bit slice module status register. The slice module status signals of different DQ groups are concatenated into multi-bit slice status signals, then concatenated into dbg_slice_status. Users can perform debugging by printing information through the serial port or via JTAG. For the description of dbg_slice_status fields (taking 16bit DQ as an

example), please refer to [Table 2-4](#).

Table 2-4 Definition of dbg_slice_status Bits

Bits	Debug Port	Description
0	dll_update_code_done	Indicator signal for the completion of DLL update. 1: Complete; 0: Not complete.
1	eyecal_move_done	Indicator signal for the completion of a single delay adjustment in eye diagram calibration. 1: Adjustment complete; 0: Adjustment not complete.
2	eyecal_check_pass	Indicator signal for the completion of eye diagram calibration 1: Eye diagram calibration completed; 0: Eye diagram calibration not completed.
3	wrcal_move_done	Indicator signal for the completion of a single delay adjustment in write calibration. 1: Adjustment complete; 0: Adjustment not complete.
4	wrcal_check_pass	Indicator signal for the completion of write calibration. 1: Write calibration completed; 0: Write calibration not completed.
5	dll_lock_tmp	DLL lock indicator, with each bit corresponding to a group, and higher bits for higher groups. 1: Locked; 0: Unlocked.
6	dqs_gate_comp_done	Indicator signal for the completion of DQS gate update. 1: Complete; 0: Not complete.
7	rddata_check_pass_tmp	Data comparison passed after DQS gating, with each bit corresponding to a group; higher bits correspond to higher groups. 1: Passed; 0: Not passed.
8	gate_cal_error	Indicator signal for DQS gate error. 1: DQS gate error; 0: No DQS gate error.
9	gate_adj_done	Indicator signal for the completion of a single position adjustment for DQS gate. 1: Adjustment complete; 0: Adjustment not complete.
10	gate_check_pass	Indicator signal for the completion of DQS gate. 1: DQS gate complete; 0: DQS gate not complete.
11	rdel_move_done	Indicator signal for the completion of a single delay adjustment in read calibration. 1: Adjustment complete; 0: Adjustment not complete.
12	rdel_calib_error	Indicator signal for read calibration error. 1: Calibration error; 0: No calibration error.
13	rdel_calib_done	Indicator signal for the completion of read calibration. 1: Calibration complete; 0: Calibration incomplete.
14	adj_rdel_done	Indicator signal for the completion of initial adjustment in read calibration. 1: Adjustment complete; 0: Adjustment not complete.
15	wrlvl_dqs_resp_tmp	Write leveling calibration completion indicator, with each bit corresponding

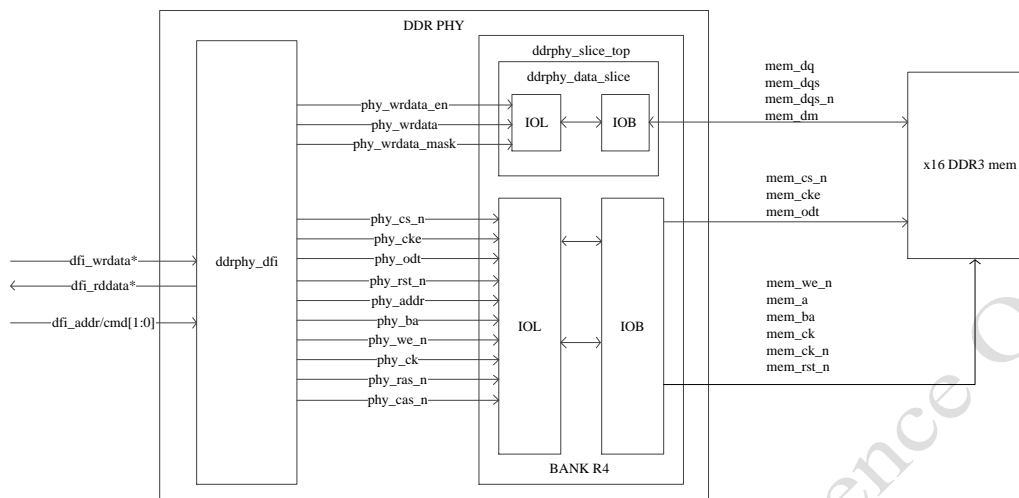
Bits	Debug Port	Description
		to a group, and higher bits for higher groups. 1: Calibration complete; 0: Calibration incomplete.
16	wrlvl_error_tmp	Write leveling calibration error indicator, with each bit corresponding to a group, and higher bits for higher groups. 1: Calibration error; 0: No calibration error.

2.2.2.4 Description of dbg_slice_state

dbg_slice_state is the slice module state machine register for each set of DDRPHY with 8-bit DQ sharing a single DDRPHY and outputting a 22-bit slice module state machine register. The slice module state machine registers of different DQ groups are concatenated into dbg_slice_stat, with the lower DQ's register data in the lower bits of dbg_slice_state. Users can debug by printing information through the serial port or via JTAG. For the description of dbg_slice_state fields, please refer to [Table 2-5](#).

Table 2-5 Definitions of dbg_slice_state Bits

Bits	Debug Port	Description
7:0	dbg_slice_rdchk_state	State output of the read calibration data comparison state machine.
11:8	dbg_slice_rpbld_ctrl_state	State output of the read calibration control DQS delay state machine.
14:12	dbg_slice_gate_state	State output of the adjustment of DQS gate delay state machine.
18:15	dbg_slice_dq_wrcal_state[3:0]	State output of the control write direction DQ delay state machine.
21:19	dbg_slice_wrlvl_state	State output of the control state machine for write leveling delay.



The typical applications of Multi-Bank-x16 are shown in [Figure 2-22](#). Single BANK-x16 is based on a specific HR BANK (R4 in the diagram), achieving pin-saving by removing certain functional pins. Non-sensitive signals are preferred. For example, CKE and RESET_n are allocated to the unused SIO pins of the FPGA.

2.3.2 Multi BANK-x64 Typical Application

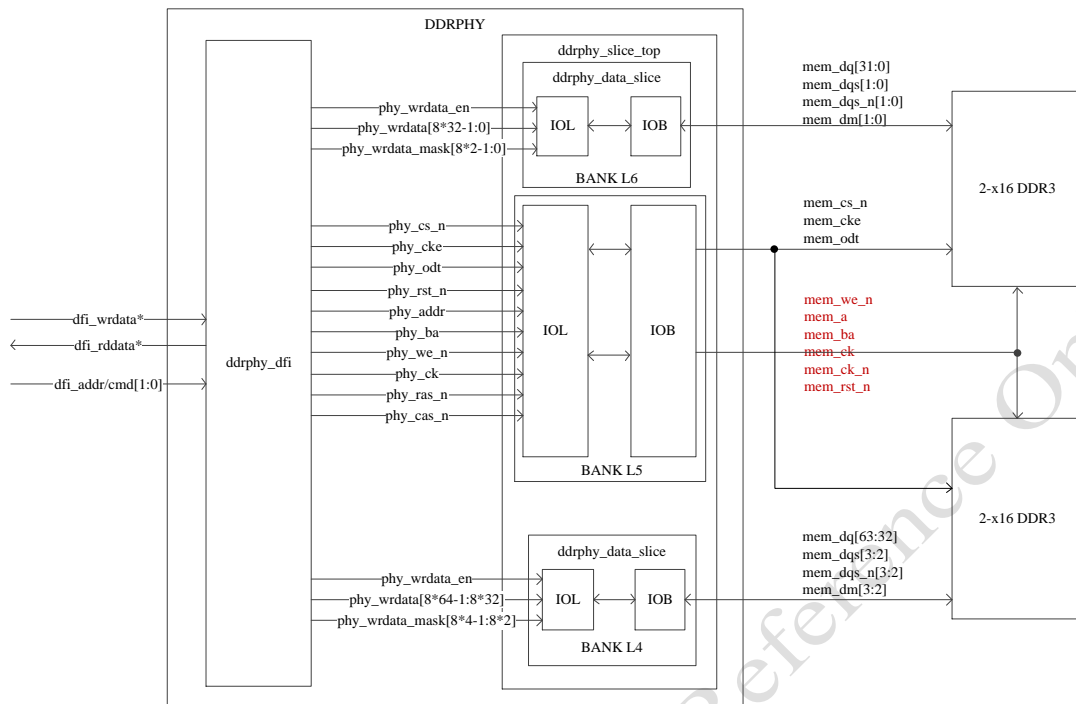


Figure 2-23 Multi-bank-x64 Structure Diagram

The typical applications of multi BANK-x64 are shown in [Figure 2-23](#).

The Multi-Bank-x64 is based on three consecutive HR banks on the same side (L4, L5, and L6 in the diagram), with Bank L5 used for the CA channel and Bank L4 and Bank L6 used for the DQ channel.

The following precautions apply to Multi-Bank-x64 applications:

- A maximum of 3 consecutive banks on the same side is allowed while cross-bank arrangement is prohibited.
- The CA signal is arranged to the middle bank (Bank L5), and the reference clock is placed in the same bank to ensure a shorter delay difference in the clock distribution path.
- In the Fly-by topology, ensure that the far-end stub tracing conforms to the design requirements as much as possible to cover AC parameters.

Chapter 3 Application Debugging FAQ

This chapter discusses the application debugging of the HMIC_S IP to help resolve relevant issues users may encounter during debugging.

3.1 Debugging Tools

The HMIC_S IP provides a variety of debugging methods, so it is essential to understand how to debug in different environments.

3.1.1 Debugger GUI Interface

The HMIC_S IP supports Debuggers that are similar to common debugging approaches. Users can add the debug signals to the .fic file, and capture waveforms for debugging using the Debugger tool.

For the specific steps to add a debugger and the debugging methods, please refer to the "*Fabric Debugger User Guide*".

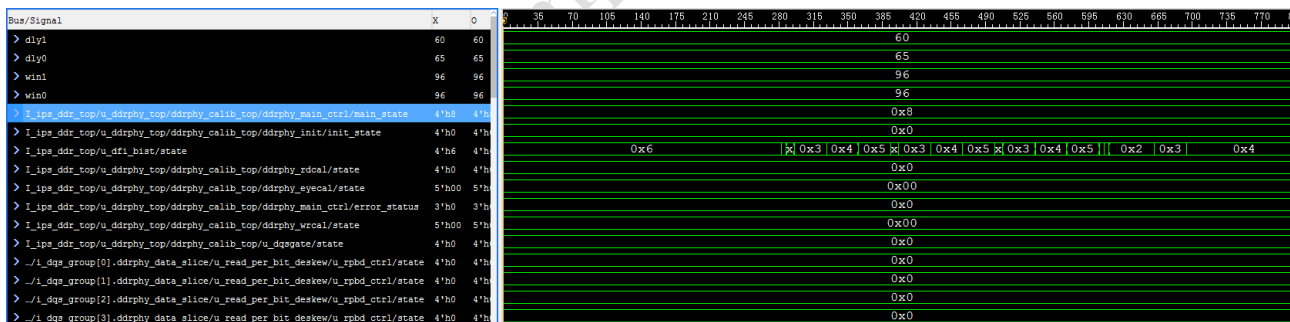


Figure 3-1 Debug Signal Debugging Waveform

The HMIC_S IP is internally embedded with certain key status signals, and other debug signals need to be loaded manually. For specific descriptions of debug signals, please refer to "[3.1.3 Debug](#)".

3.1.2 Example Design

This section mainly introduces the Example Design scheme based on HMIC_S IP (Controller + PHY structure). In this scheme, the user logic acts as AXI Master with HMIC_H IP as AXI Slave. The user logic writes data through the Write channel of the AXI interface and receives data through the Read channel for data comparison. If a data error occurs, err_flag_led will be pulled high.

3.1.2.1 Design Block Diagram

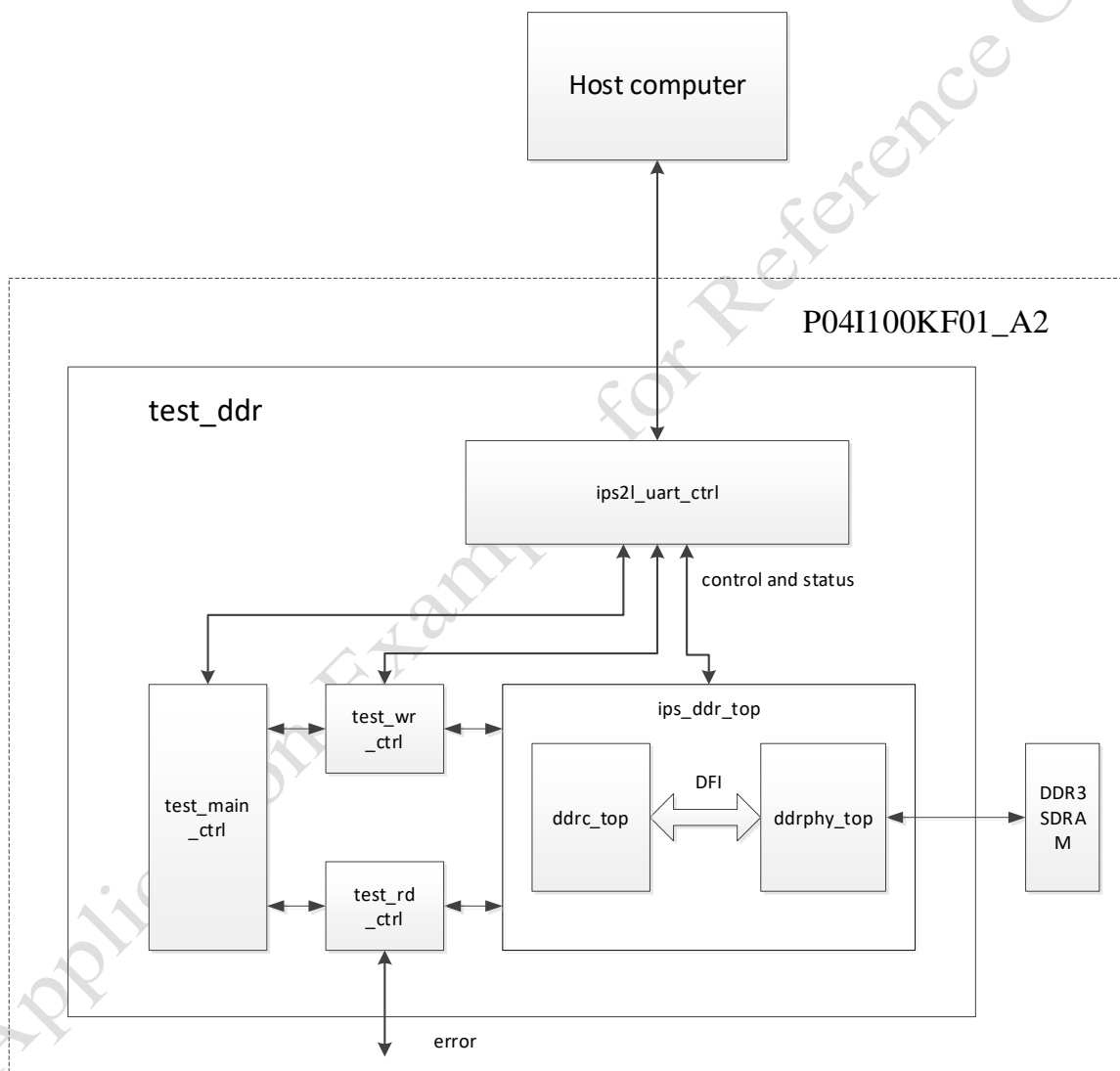


Figure 3-2 Example Design System Block Diagram

The system block diagram of the Example Design is shown in [Figure 3-2](#), where test_main_ctrl is the control module for AXI read and write instructions, test_wr_ctrl is the control module for AXI write instructions and data writing, test_rd_ctrl is the control module for AXI read instructions and

data reading, and ips2l_uart_ctrl is the serial port conversion module for easy control and internal state reading during debugging.

3.1.2.2 Test Method

In the Example Design, the user logic performs read and write operations on HMIC_S IP and checks the readback data. The detailed test process is as shown in [Figure 3-5](#).

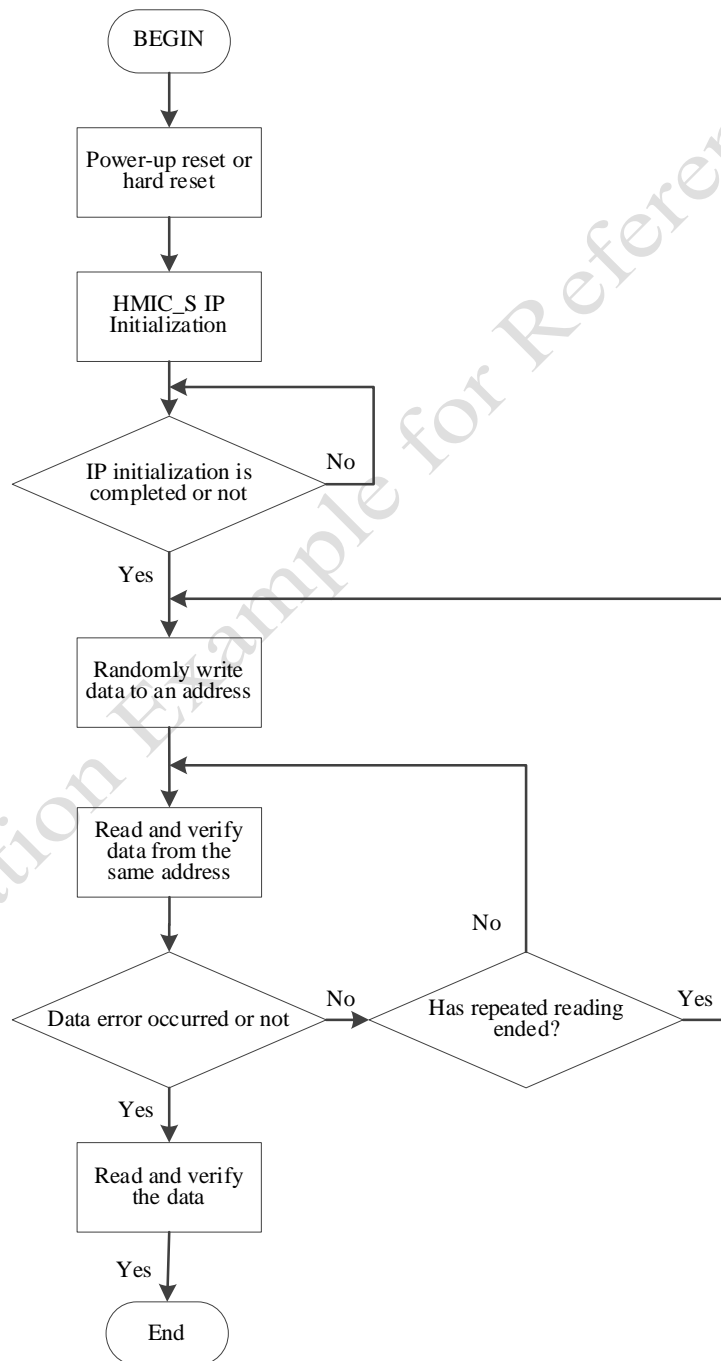


Figure 3-3 Example Design Test Flowchart

After the system is powered up or a hard reset is initiated, the HMIC_S IP begins initialization. Once initialization is completed (indicated by the `ddrc_init_done` signal going high), the `test_main_ctrl` module controls the `test_wr_ctrl` module to generate write instructions and write data to initialize the DDR chips. After write full, `test_main_ctrl` starts random read and write operations, and `test_rd_ctrl` checks the readback data to determine if there are any errors.

Attention:

Do not directly use the Example Design generated by the IP for Flow on-board testing. Constrain pins according to the actual pin connections of the single board, then proceed with running flow and on-board testing.

By using the PDS software IPC tool, a typical DDR3 application example project can be directly generated through the HMIC_S IP interface. This example project includes synthesizable test stimuli, and has undergone simulation and on-board verification. Various modes are available within the stimuli, with related control signals referenced in [Table 3-1](#). It can be used to study and analyse HMIC_S IP-based user designs, and also help in identifying board-level issues.

Table 3-1 Bist Mode Control Signals

Signal Name	Bit width	Description
<code>wr_mode</code>	2	Read/write mode 2'b00: Random read/write; 2'b01: Random read/write, read multiple times, number of reads is <code>read_repeat_num+1</code> ; 2'b10: Write first, then read, address remains unchanged; 2'b11: Write First, then read, read multiple times, number of reads is <code>read_repeat_num+1</code> .
<code>data_mode</code>	2	Data mode 2'b00: PRBS; 2'b01: PRBS + increment; 2'b10: Crosstalk mode, crosstalk position controlled by <code>dq_inversion</code> ; 2'b11: Fixed pattern, pattern configured by parameters.
<code>read_repeat_num</code>	4	Number of repeated reads, valid when <code>wr_mode=2'b01</code> or <code>2'b11</code>
<code>dq_inversion</code>	8	Data inversion
<code>data_order</code>	1	Arrangement of write data 1'b0: Horizontal 1'b1: Vertical
<code>insert_err</code>	1	Active on the rising edge, each rising edge triggers an erroneous write data.
<code>manu_clear</code>	1	Clears error flag, active high
<code>len_random_en</code>	1	Random enable of DFI read/write len 1: Random 0: Not random, DFI read/write len is <code>fix_wr_len</code>
<code>fix_axi_len</code>	4	When <code>len_random_en</code> is 0, DFI read/write len is fixed at <code>fix_axi_len</code> . When

Signal Name	Bit width	Description
		len_random_en is 1, fix_axi_len is invalid.

In addition, this example project also outputs the results of the Bist test and related indicator signals for read/write errors to facilitate user debugging. Refer to the indicator signals at [Table 3-2](#).

Table 3-2 Bist Mode Indicator Signals

Signal Name	Bit width	Description
test_rd_state	3	test_rd_ctrl module state machine
test_wr_state	3	test_wr_ctrl module state machine
test_main_state	4	test_main_ctrl module state machine
err_cnt	8	Number of readback data errors
err_flag_led	1	Readback data error flag
err_data_out	mem_dq_width*8	Readback error data
exp_data_out	mem_dq_width*8	Desired data
err_flag_out	mem_dq_width*8	Error flag for each bit of error data
next_err_data	mem_dq_width*8	Readback data with reoccurrence error
next_err_data	1	Error flag for readback data with reoccurrence error
err_data_pre	mem_dq_width*8	The previous beat of data of the error data
err_data_aft	mem_dq_width*8	The next beat of data of the error data

Whenever there is readback data, the test_rd_ctrl module verifies the readback data. If the verification fails, the error flag, err_flag_led, is pulled high and latched. The first erroneous data is latched to err_data_out, and its corresponding error flag is latched to err_flag_out. Additionally, the two beats of valid data before and after the first erroneous data are stored in err_data_pre and err_data_aft, respectively. If further errors occur after the first erroneous data, the second erroneous data is latched to next_err_data. Setting manu_clear to high clear the error flags err_flag_led and next_err_flag and error counters like err_cnt.

Users can directly set the reset initial value of the Bist Mode control signal to configure different read-write test modes or dynamically control the Bist Mode control signal value via the serial port. The Bist Mode indicator signal can also be read via the serial port or captured by a Debugger tool for waveforms.

The following content will demonstrate the hardware-related verification based on this example project.

3.1.3 Debug Signals

The HMIC_S IP design features a complete set of debug interfaces that can quickly identify calibration status, read/write window margins, and other information. These debug interfaces can be brought out and mounted to a debugger tool for capturing waveforms, or be used for debugging purposes through the serial port bus.

3.1.4 Test Boards

P04I100KF01_A2 is a high-performance estimation board developed by Pango Microsystems and equipped with an FPGA and a 64-bit (4 x16 chip) wide DDR3 interface.

3.2 Debugging Methods

Due to different application environments, hardware-related issues vary from initialization to reliability testing. This section provides general debugging steps and methods that enable designers to effectively perform hardware debugging using a debugger or debug signal resources. Specific signal characteristics and usage methods will be described in sections relevant to debugging issues.

3.2.1 Debug Using Example Design

Using the Example Design provided by the IP, isolate any unrelated logic interference, and add only the necessary logic, generate a small version of the project, ensure that pin constraints are correct and timing is converged, open the debug signal resources and execute testing on the target board, and pay close attention to the debug information in case of calibration or read/write data errors as much as possible.

Attention:

When using the debugger tool, it is not mandatory to capture calibration and window results by using the HMIC_S IP Example Design and enabling the debug interface. Instead, focus more on how to perform debugging based on known issues in operation.

Generally, designers connect debug signals to the serial port or JTAG interface. During debugging, they can read the calibration status, phase adjustment, and window calculation results, and debug the functions of the HMIC_S IP by enabling specific signals.

3.2.2 Common Software Issues

After generating the IP, if you encounter errors while running the flow test based on the Example Design, you can handle them by querying the "*Pango_Design_Suite_User_Guide*". Below are methods for addressing common DDR software issues.

3.2.2.1.1 CA group constraint error

When the software reports the following errors, it indicates an error in setting up the grouping for the CA signals.

E: Place-0054: The IO of inst I_ips_ddr_top/u_ddrphy_top/ddrphy_slice_top/i_ca_group[1].u_ddc_ca/ddc_inst must be constrained in the same DQS group.

When configuring the IP, it is recommended to manually set the location constraints for the CA signal. At this point, the IPC will automatically set the grouping in the IP's underlying code based on the CA signal location. Once the IP is generated, the grouping is fixed. Modifying the CA signal location constraints through the fdc file at this point will result in group changes, causing errors. Therefore, if you need to modify the CA signal location constraints after IP generation, it is recommended to repeat the IP generation, as detailed in the module instantiation section.

3.2.2.1.2 DQ Group Location Error

When the software reports the following errors, it indicates an error in setting up the grouping for the DQ signals.

E: Place-0054: The IO of inst I_ips_ddr_top/u_ddrphy_top/ddrphy_slice_top//i_dqs_group[1].u_ddrphy_data_slice/u_ddc_dqs/opit_0 must be constrained in the same DQS group.

Therefore, it is necessary to check whether each 8-bit DQ signal and its DQS and DM signals are correctly constrained into one group.

3.2.2.1.3 CPD Location Error

When the software reports the following error, it indicates an error in the CPD location constraint.

E: Place-0006: The I_ips_ddr_top/u_ddrphy_top/u_ddrphy_cpd/cpd_inst cannot be placed.

In the fdc file generated by the IP, the location constraints for CPD and other GTP hard cores match the IO location constraints within the IP. Manually modifying the CPD or IO position constraints after IP generation may prevent the modified CPD output from reaching the modified IO clock domain and cause the aforementioned error. Therefore, it is not recommended to modify the GTP hard core location constraints, only the hierarchy of the location constraints according to the IP's location in the project.

3.2.3 General Inspection

Ensure that all core timing constraints included in the Example Design are properly incorporated into the existing design and that all timing constraints have been closed after implementation. In view of this, it is necessary to conduct a basic inspection on the following items.

3.2.3.1 Checking External Clock Source

It is recommended to use a clean and reliable external clock input source as an effective reference and ensure its stability and proper drive strength. For information about the characteristics of clock inputs, please refer to the AC/DC Switching Characteristics section in the "*DS04001_Logos2 Family FPGA Device Data Sheet*". Ensure that all GPLL and PPLL output clocks are locked and that the CPD output is locked.

3.2.3.2 Following User and Design Guides

Ensure that all design references follow the PCB Design Guide to meet the requirements for trace matching, topology & trace, noise, termination impedance, IO standards, and other aspects. According to these principles, the key to successful high-speed memory interface design lies in accurate hardware design and signal integrity analysis.

3.2.3.3 Ensuring Rational Voltage Settings

Measure all the voltages on the single board during both operation and non-operation periods of the whole system, ensuring that the voltages are set within an appropriate range and the voltage noises are within a controlled range.

- Ensure that the pull-up voltage V_{TT} for the termination matching resistor is $V_{CCIO}/2$.
- Ensure that the V_{REF} of the DQ bank on the FPGA side is set to half of the bank voltage/DDR3 chip supply voltage. When using an internal reference voltage, you need to set the $VREF_MODE_VALUE$ to 0.5 via UCE or fdc files; when using an external reference voltage, $VREF$ can be generated by voltage dividing with two 1K ohm resistors or provided by a dedicated power supply chip;
- Ensure the voltage ripple is within the permissible range. If significant ripple changes are observed, promptly analyse the generation paths of the voltage ripples and perform targeted decoupling, such as increasing the decoupling capacitance or altering the near-end and far-end feedback paths.
- Check for voltage dip under full load. If so, ensure that the VCC adheres to the normal operating conditions under full load.

3.2.3.4 Checking External Calibration Resistance RZQ

In actual applications, please check if the external calibration resistance RZQ of the FPGA and memory is $240\ \Omega$ with an accuracy of 1%.

Attention:

If the drive strength in the write direction does not meet the design requirements, please verify the value, accuracy, and temperature rating of the external calibration resistor of the FPGA, then try reducing the value by 5–10%.

Similarly, if the drive strength in the read direction does not meet the design requirements, verify the value, accuracy, and temperature rating of the external calibration resistor of the memory, then try reducing the value by 5–10%.

3.2.3.5 Checking Reset Signal

Check the system reset to ensure the duration of the reset signal is not less than 3 system clock cycles and verify that the reset polarity of the DDR PHY is low.

3.2.3.6 Measuring Clock Signal

Check if the CK/CK_n, DQS/DQS_n, and system clock duty cycle meet the specification requirements, and assess the general signal integrity of these signals. CK jitter is an integrated manifestation of ref_clk phase noise, core noise, PPLL performance, and IO driving load capacity, and also serves as the basis for the reliable operation of the chip.

3.2.3.7 Checking Physical Memory Allocation

Please verify that the physical memory for the single board is correctly assigned in the GUI design, including device timing parameters and signal bit width, such as row address width, column address width, and bank address width. Mismatches may lead to read or write errors.

3.2.3.8 Checking DM Pin

The DM signal of DDR3 does not support the multiplexing function. If this port is not used in the

design, please ensure that the DM pin is correctly pulled down to earth (recommended pull-down to ground via a 120 ohm resistor down). If the DM pin of the memory chip is left floating, it may lead to unpredictable failures during the calibration process.

3.2.3.9 Checking CS_n Pin

For DDR3 devices, the FPGA drive CS_n chip select signal is not mandatory in the single bank design. Designers can consider using an external pull-down resistor to keep it at a low level. The pull-down resistor value should be based on the memory manufacturer's recommendations.

Accordingly, ensure proper handling of the CS_n signal during the HMIC_S IP configuration. If the pin has been pulled down to a low level and remains in that state, make sure to disable the CS_n signal during IP configuration.

3.2.3.10 Checking ODT Configuration

ODT is an essential control signal for the normal operation of DDR3 devices. The optional configurations of ODT can be done through the HMIC_S IP GUI. Initially, the ideal ODT settings can be configured based on simulation, including both the FPGA end and the chip end. For the ODT features, please refer to the "*UG040012 Logos2 Single Board Hardware Design Guide*" and the memory manufacturer device manual.

3.2.3.11 Checking for Floating Pins

Check for potentially floating pins. Floating reset_n or address pins may cause inconsistencies during repeated reset or power-up/power-down operations. If inconsistency errors arise in the calibration process, please check whether the reset_n and address pins are properly and reliably connected.

3.2.3.12 Analysing Signal Integrity

Perform a general signal integrity analysis on the DDR system, for example:

- Set the ideal ODT value for the device based on the HMIC_S IP GUI, corresponding to the configuration register MR1 in the RTL code. Simultaneously, set the appropriate output drive strength, run hardware simulation, and analyse the simulation results to determine the most suitable ODT configuration and output drive strength.
- For the DDR3 memory system, it is necessary to use an oscilloscope to analyse the phase alignment, V_{IL}/V_{IH} , jitter, slew rate, and window of the DQ/DQS signal during read or write processes for signal integrity.
- If possible, observe the address, command signal phase alignment, and V_{IL}/V_{IH} using an oscilloscope, and analyse the signal integrity.

3.2.3.13 Frequency Reduction Test

The purpose of frequency reduction testing is to further differentiate between a hardware problem and a software problem by reducing the system operating speed. Generally, decreasing the operating speed leads to an expanded window margin, simplifying the closure of system timing. If the frequency reduction test succeeds, please verify the signal integrity and the hardware design's compliance with "*UG040012_Logos2 Single Board Hardware Design Guide*". For the Controller+PHY application scenarios, please check the AC parameters in the Controller configurations. Otherwise, if the test fails, please confirm the validity of the functional simulation and check for other issues.

3.2.4 Calibration Issues

The most common issue during DDR interface debugging is initialization failure. Most of these failures are caused by design rule conflicts. Therefore, when initialization fails, it's essential to conduct a general design check, as described in section "[3.2.3 General Inspection](#)". After ruling out design issues, you can then identify the status and causes of the initialization failure.

3.2.4.1 Initialization Calibration State

[Figure 3-4](#) shows the initialization of the IP layer PHY and the different stages of the calibration process. The data access between the PHY and memory follows the JEDEC DDR3 protocol standard initialization sequence. For DDR3 SDRAM chips, a series of mode registers need to be set

through MRS instructions to complete the initialization. These mode registers define various SDRAM functions such as burst length and read/write CL and AL. For detailed descriptions of the mode register functions, please refer to the memory manufacturer device manual. It should be noted that the HMIC_S IP does not report any calibration failures during the memory initialization. All other initialization calibration issues and analyses will be introduced in the following sections.

Attention:

Before performing DDR3 initialization calibration, take measures to keep other components of the system in a silent state.

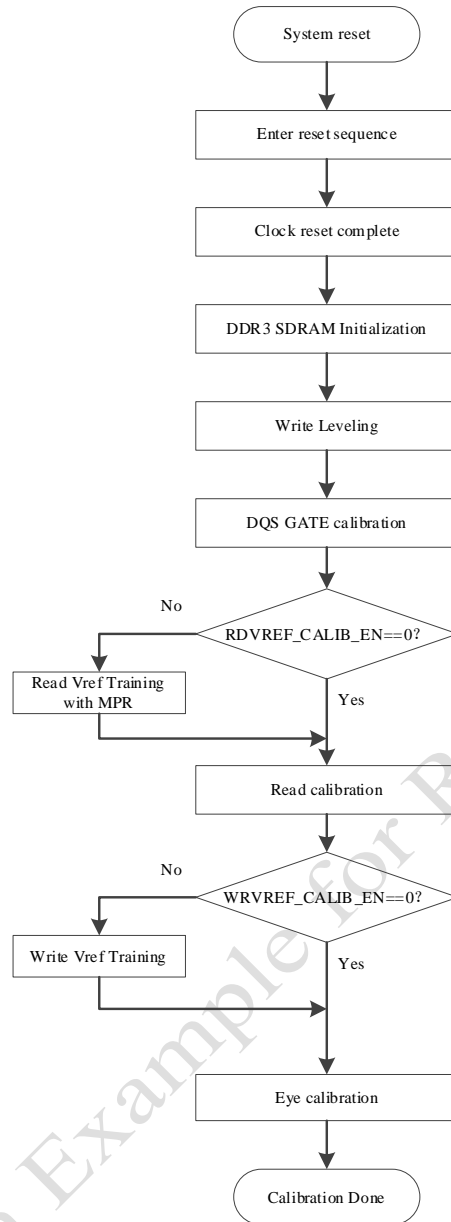


Figure 3-4 PHY Initialization Calibration Process

3.2.4.2 Calibration Error State Identification

dfi_init_complete (phy_init_done) is the PHY initialization calibration success flag. If dfi_init_complete remains 0 after the bitstream download and reset release, the PHY initialization fails, and the cause of the failed initialization can be pinpointed through the following steps.

1. Master Status Information during PHY Initialization Calibration

The calibration failure status can be quickly identified through the Debug Signals interface of the HMIC_S IP. Table 3-3 provides the common debug information of the PHY initialization calibration state, corresponding to debug_calib_ctrl[29:26]. The current calibration failure state can

be determined by reading the dbg_main status signal value.

Table 3-3 Master Status Information during PHY Initialization Calibration

Debug Signal	Direction	Bit width	Value	State	Description
dbg_main	Output	4	4'd0	IDLE_PHASE	IDLE state
			4'd1	INIT_PHASE	Initialization process
			4'd2	WRLVL_PHASE	Write leveling process
			4'd3	RDCAL_PHASE	Read calibration process
			4'd4	WRCAL_PHASE	Write calibration process
			4'd5	EYECAL_PHASE	Eye calibration process
			4'd6	UPDATE_PHASE	DQS gate adjustment process
			4'd7	DDRC_PHASE	Initialization completed Update calibration request handling

2. Debug Output Signals and Manual Debug Functions

"2.2.1 Debugging Signals" provides the commonly used debug output signals including window values and delay values, as well as reset control, DQS phase adjustment and read direction manual window adjustment function interfaces. These output information and debug function interfaces can assist users in quickly debugging and analysing DDR systems.

When the PHY initialization fails, DDRPHY will reset, and after resetting the PLL and DLL, initialization will start again. The relevant signals are detailed in <instname>_ddrpht_top.v, as described in Table 3-4. Users can capture relevant signals through the Debugger tool to determine at which stage the PHY initialization process failed.

Table 3-4 PHY Initialization-Related Signals

Port Name	Bit width	Description
phy_pll_rst	1	PLL reset signal
dfi_error	1	DFI interface error flag signal
ddrphy_dll_rst_n	1	DLL reset signal
ddrphy_rst_n	1	DDRPHY reset signal
wrlvl_error	1	Writer leveling failure flag
gate_cal_error	1	DQS gate calibration failure flag, valid window less than 3
rdel_calib_error	1	Read calibration failure flag
wrcal_error	1	Write calibration failure flag

3. Serial Port Debug Information Output Example

Based on the P04I100KF01_A2 estimation board, run the 1066Mbps x32 Example Design, print the debug information using serial port scripts, and read debug_data (the corresponding bus address in the Example Design is status_bus_90 ~ status_bus_a1) and other debugging signals via the appropriate register bus address. The script processes the fields of debug_data and other data according to the section "[2.2.2 Description of Debug Signals](#)", resulting in status information, window values, delay values, etc. When executed properly, the expected results are as follows, provided for reference only.

The master state value for the initialization calibration is 7, indicating that the initialization calibration is completed. The initialization calibration fails if the master state value is not 7. The initialization calibration failure state is determined based on the master state values given in [Table 3-3](#).

```
calib_main_state is: 7
calib_init_state is: 0
calib_wrlvl_state is: 0
calib_rdccl_state is: 0
calib_wrcal_state is: 0
calib_eyecal_state is: 0
calib_upcal_state is: 0
calib_error is: 0
```

Figure 3-5 Master Status Information

```

g0_coarse_slip_step is: 3   g2_coarse_slip_step is: 3
g0_read_clk_ctrl is: 4     g2_read_clk_ctrl is: 4
g0_gate_win_size is: 4     g2_gate_win_size is: 3
g0_gate_check_pass is: 0   g2_gate_check_pass is: 0
g0_rddata_check_pass is: 1 g2_rddata_check_pass is: 1
g0_dqs_even_bin is: 38     g2_dqs_even_bin is: 43
g0_dqs_odd_bin is: 76      g2_dqs_odd_bin is: 72
g0_total_margin_even is: 96 g2_total_margin_even is: 89
g0_total_margin_odd is: 118 g2_total_margin_odd is: 106
g0_eyecal_check_pass is: 0 g2_eyecal_check_pass is: 0
g0_wrlvl_step is: 45       g2_wrlvl_step is: 58
g0_wl_p_ov is: 0           g2_wl_p_ov is: 0
g0_wrlvl_dq is: 1          g2_wrlvl_dq is: 1
g0_wl_p_dll_bin is: 105    g2_wl_p_dll_bin is: 118

g1_coarse_slip_step is: 3   g3_coarse_slip_step is: 3
g1_read_clk_ctrl is: 4     g3_read_clk_ctrl is: 4
g1_gate_win_size is: 4     g3_gate_win_size is: 4
g1_gate_check_pass is: 0   g3_gate_check_pass is: 0
g1_rddata_check_pass is: 1 g3_rddata_check_pass is: 1
g1_dqs_even_bin is: 37     g3_dqs_even_bin is: 45
g1_dqs_odd_bin is: 73      g3_dqs_odd_bin is: 78
g1_total_margin_even is: 89 g3_total_margin_even is: 97
g1_total_margin_odd is: 106 g3_total_margin_odd is: 117
g1_eyecal_check_pass is: 0 g3_eyecal_check_pass is: 0
g1_wrlvl_step is: 46       g3_wrlvl_step is: 58
g1_wl_p_ov is: 0           g3_wl_p_ov is: 0
g1_wrlvl_dq is: 1          g3_wrlvl_dq is: 1
g1_wl_p_dll_bin is: 106    g3_wl_p_dll_bin is: 118

```

Figure 3-6 Training Calibration Values of 4 Groups

For the fly-by topology, the write leveling training values for each group are slightly different based on the actual routing. For example, [Figure 3-6](#) shows that the wrlvl_step delay values exhibit a relationship of $g0 < g1 < g2 < g3$. The DQS phase adjustment values obtained from the training of the 4 groups are not significantly different. For example, Group g0 has values of 3, 4, 4, and Group g2 has values of 3, 4, 3.

3.2.4.3 Write Leveling Calibration Issues

The output paths and drive clocks of CK and DQS inside the FPGA are fully consistent. Therefore, when `clk_dly_set_bin=8'd0`, the CK and DQS waveforms are edge-aligned before write leveling training begins, as shown in [Figure 3-7](#).

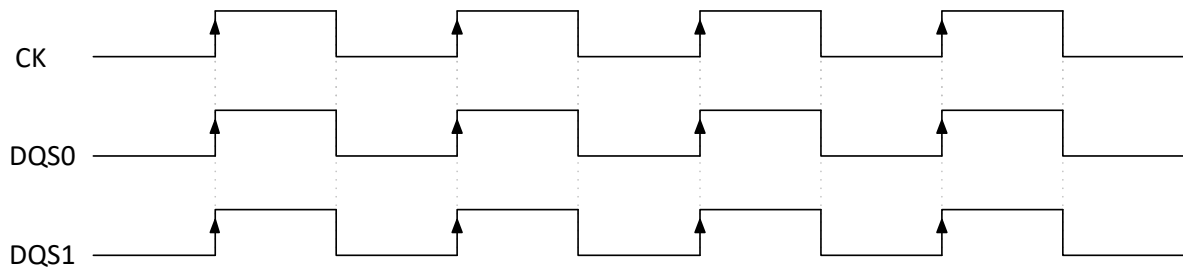


Figure 3-7 Phase Relationship of CK and DQS Before Write Leveling Calibration

To improve signal integrity, the fly-by topology is generally used for the clock, address, and command signals of DDR3 SDRAM. This topology may cause a fixed offset for the DQS and CK of each chip (the offsets of the default clock, address, and command signals are the same). The aim of write leveling is to separately adjust the phase shift of DQS using the controller to compensate for its offset and meet tDQSS timing requirements. For other timing requirements, please refer to the memory manufacturer datasheet.

According to the fly-by design requirements for DDR PCB topology, the delay of the CK signal from the FPGA pin to the SDRAM pin must be greater than the delay of all DQS signals from the FPGA pin to the SDRAM pin. The delay includes the FPGA's package delay and PCB trace delay. For example, if the trace and packaging delays for CK are 300 ps and 175 ps, respectively, and for DQS0 they are 240 ps and 160 ps, while for DQS1 they are 200 ps and 125 ps, then $T_{ck_delay} = 475ps > T_{dq0_delay} = 400 > T_{dq1_delay} = 325$. Therefore, the phase relationship between CK and DQS at the SDRAM pin is shown below.

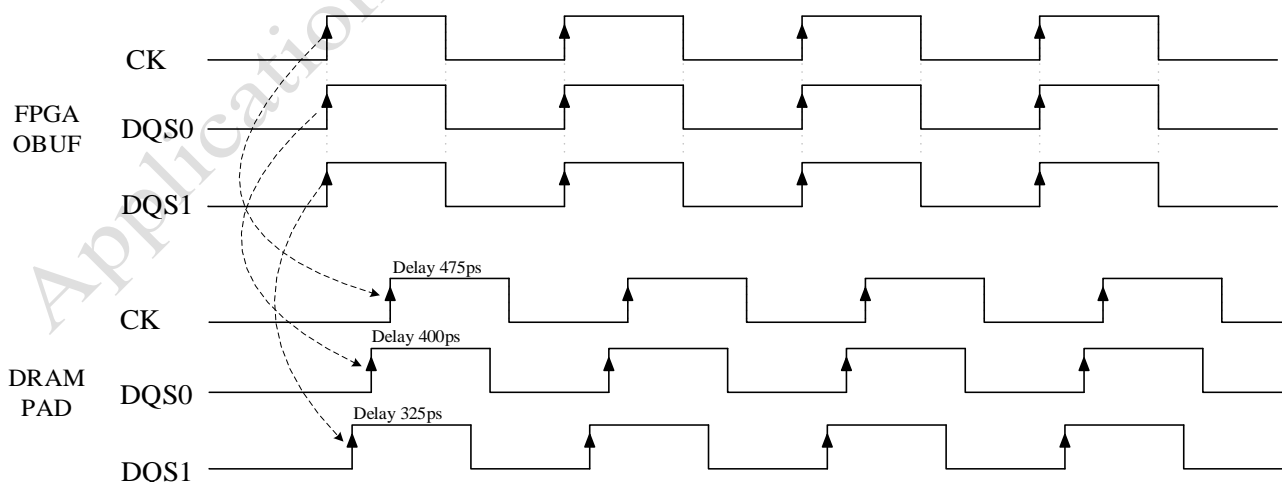


Figure 3-8 Phase Relationship Between CK and DQS at the SDRAM Pin

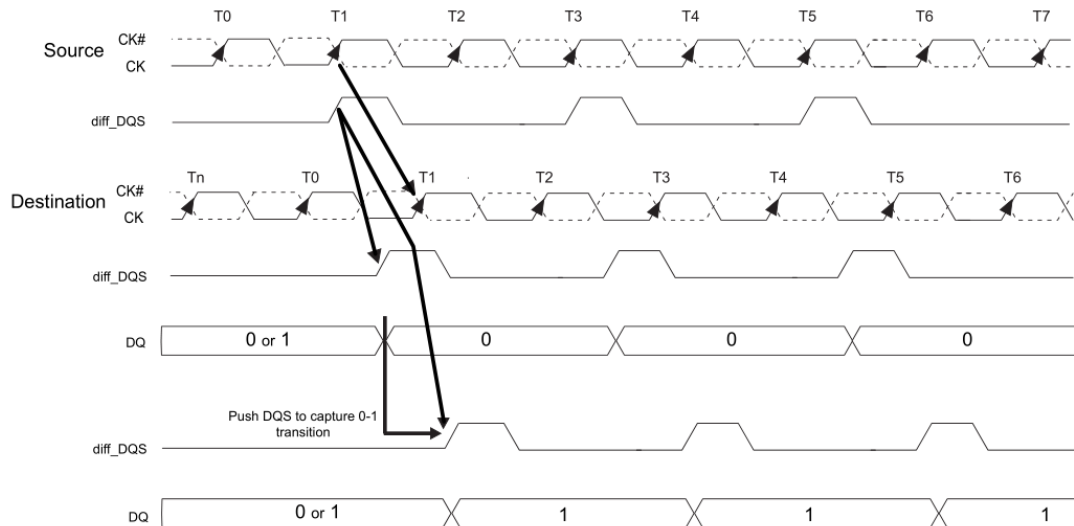


Figure 3-9 Write Leveling Mode Calibration Diagram

During the entire write leveling process, the controller enables the write leveling function on the chip using the MRS instruction. It then sends a single pulse DQS and drives the DQS phase shift. The DDR3 SDRAM internally uses DQS to sample CK. If a transition from 0 to 1 is sampled, it feeds back the sampling signal by pulling DQ high to indicate that write leveling is complete. When the PHY detects DQ=1 (DQS0 samples CK with values output through DQ0~DQ7, and DQS1 samples CK with values output through DQ8~DQ15, and so on; the completion of write leveling can be determined by checking if either a single bit DQ or all DQs equal 1), it exits write leveling mode, completing the write leveling calibration process. After completing the entire training process, the MRS instruction is used again to disable this function. The training result guarantees the basic alignment between the DQS and CK on each chip.

According to the phase relationship in [Figure 3-8](#), the initial value of DQS sampling CK is 0. During calibration, DDRPHY continuously increases the delay of DQS by increasing the step value of DDC's DELAY_STEP0. DELAY_STEP0 controls the delay of WCLK. The increment of DQS delay is equal to the increment of WCLK delay with a precision of 10ps; when DQS samples that CK's value changes from 0 to 1, it is considered that the write leveling calibration of this DQS group is complete, and the step value at this time is recorded as the wrlvl_step delay value of this DQS group.

Generally, it is required that the PCB trace CK be delayed by about 50ps compared to DQS. However, some PCB designs may not strictly adhere to the requirement that "The delay of the CK signal must be greater than that of all DQS signals." In such cases, the delay of one or all DQS signals may exceed that of CK. At this time, the phase relationship between CK and DQS at the DRAM pin is shown below.

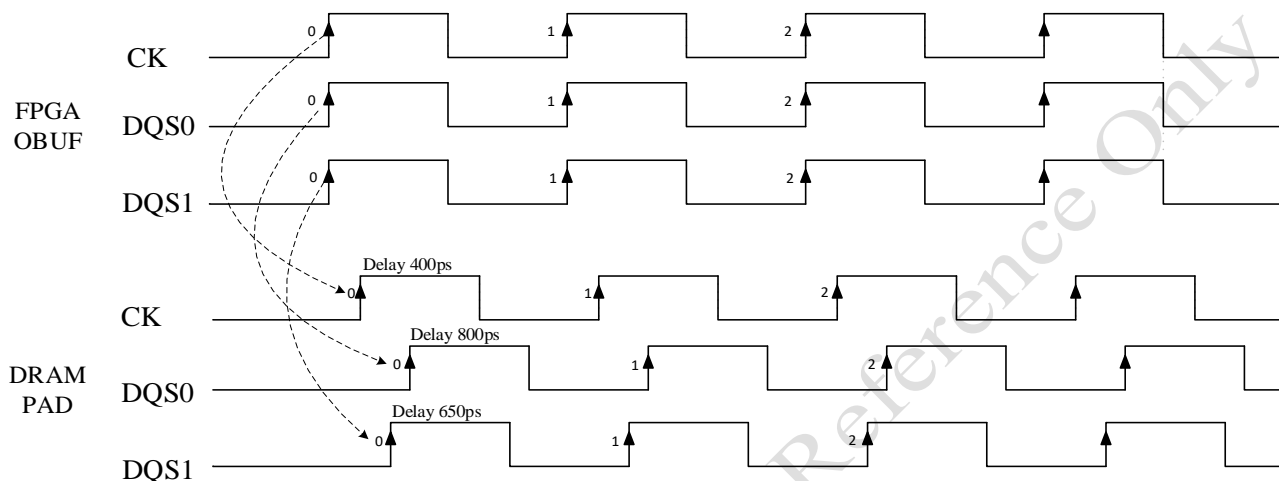


Figure 3-10 DQS Delay Greater than CK Delay

At this time, the initial value of CK sampled by DQS during write leveling calibration initialization is set to 1. The PHY will automatically adjust the CK delay (all CA signals will undergo the same adjustment). Under normal circumstances, within the delay control range (0~127*5ps), the CK low level should be stably detected through DQS sampling, allowing the initial value of CK to be reset to 0 before proceeding with the subsequent write leveling calibration.

For example, at a rate of 1066Mbps, the setup and hold time of CK and DQS should be at least 245ps. When the difference between CK delay and DQS trace delay is lower than this value, a calibration failure will occur during write leveling.

The HMIC_S IP provides users with the init_ck_dly_step debugging interface at the DDR PHY layer port. Users can adjust the initial delays of CK (for controlling the overall offset of the CA signal) with an accuracy of approximately 5ps.

The trace delay difference between CK and DQS should not exceed 1 TCK. For application scenarios that adopt the fly-by topology expansion, the CK routing length is usually much greater than the DQS routing length on the furthest-end chip. Therefore, under certain circumstances, there

might be a delay difference exceeding 1 TCK between CK and DQS, which may also lead to a write leveling calibration failure.

For example, if the DDR rate is 800Mbps, then the trace time difference between CK and DQS should not exceed 2500ps. Suppose the PCB trace delay is 6mil/ps, the CK trace length must not exceed the DQS length of 15000mil.

To solve the problem of excessive CK delay, the HMIC_S IP provides the `init_wrlvl_step` interface for each DQS group for presetting the initial value of write leveling.

When the write leveling calibration fails, resulting in PHY initialization failure, the serial script may read a `dbg_main (debug_calib_ctrl[21:19])` value of 2. This is due to the limited delay adjustment range of CK and DQS, with maximum adjustment values of 127*5ps and 255*10ps, respectively. At this time, the `wrlvl_step` delay value in the write leveling calibration is at its maximum value of 8'hff. When write leveling calibration is not completed yet, i.e. write leveling calibration overflows, if the DQS delay adjustment reaches its maximum value but still fails to sample the CK rising edge, the potential causes could be the following:

1. The DDR interface rate is low;
2. The CA pin is incorrectly connected or left floating. This would lead to the chip failing to complete power-on initialization correctly and unable to enter Write Leveling mode;
3. DQ lines are left floating or incorrectly connected, for instance, DQ is connected to DM. Note that swapping within a DQ group does not count as incorrect DQ wiring. For example, within the DQS0 group, DQ0 can be connected to DQn (n=0, 1, 2, 3, 4, 5, 6, 7).

Since the write leveling calibration is not mandatory for single x4, x8, and x16 devices, users can verify the presence of any problems during write leveling by disabling the `WRLVL_EN` function in the `<object>_ddrphy_top.v` file. Setting the `wrlvl_en` interface to 1'b0 disables the write leveling function.

3.2.4.4 DQS Gate Issues

1. DQS Gate Initialization Calibration

During read operations, the DQS signal is in a high impedance state before the preamble, while the low level of the preamble part cannot reach its most stable state. Therefore, the gate signal for read DQS needs to be trained to filter out the preceding high impedance state and preamble, ensuring the

effective DQS for the entire read operation. This is known as read DQS gate training.

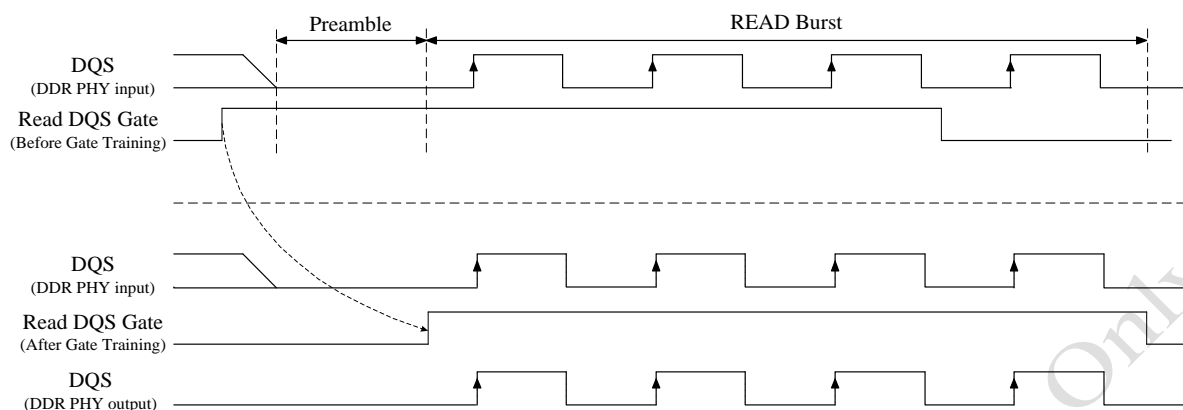


Figure 3-11 Read DQS Gate Training Diagram

During the entire read operation, the calibration of DQS aims to determine the valid gate value for each group, enabling accurate sampling of each byte during the PHY-based read process.

During the first gate training, once the DQS gate position is found, a pattern sequence of 64'h1b1b_1b1b_1b1b_1b1b is written, and a readback is performed. The read data is compared against the pattern for consistency.

If the readback data is only misaligned with the pattern, the number of misaligned bits will be used to adjust the cycle difference between the CA signal and the DQS signal; otherwise, the read training will fail. After the read training is completed, a second gate training will be performed to verify whether the gate position is correct.

The DDR3 PHY uses an internal clock to sample the DQS signal and return a valid value. As part of the training algorithm, this sampling process can determine the phase relationship between the valid rising edge and the sampling clock of the DQS.

The PHY adjusts the sampling of the DQS signal using varying granularities, including coarse adjustment (coarse_slip_step) and fine adjustment (read_clk_ctrl). These adjustments ultimately provide feedback on the valid sampling value of the DQS through the hardware module and determine the DQS preamble through sequential checking. The low-level duration of the preamble in typical DDR3 is approximately 3/4 of an SDRAM clock cycle and begins in a tri-state. Please refer to *JESD79-3D, DDR3 SDRAM Standard* for details.

2. DLL Update

During the system operation, the step values of the internal delay chain of the DLL change accordingly with the temperature and voltage, resulting in changes in the generated 90 ° delay code. Thus, positions using this 90 ° delay code will inevitably be affected, such as DQS gate calibration and write direction calibration.

If the system experiences initialization calibration failures or bit errors when external conditions change, this could be attributed to a deviation in the 90 ° delay code generated by the DLL. In light of this, HMIC_IP provides the debug_dll_update_en debug interface signal for enabling the timed detection function of PHY's internal DLL, ensuring that the output 90 ° delay code meets the design requirements.

3. Manual Adjustment of DQS Gate Position

If the initialization calibration fails and stays in the DQS gate state (for example, the dbg_main value is fixed to 6), users can manually adjust the DQS gate window sampling position through the Debug interface described in [Table 3-5](#), thus promptly discovering problems involving DQS gate window recognition and potential issues with DQS signal integrity during the initialization calibration process. The DQS gate position fixing can be enabled via a serial port script. Adjust the coarse and fine positions of the DQS gate incrementally from small to large. After each adjustment, a reset is performed, and the status signal is observed to determine whether the initial calibration is successful and the DQS gate status passes.

Table 3-5 DQS Gate Manual Adjustment Interface Signal

Debug Signal Name	Direction	Bit width	Description
force_read_clk_ctrl	Input	1	Enables DQS gate position fixing 0: DQS gate position changes during the training process. 1: DQS gate position remains unchanged, always at the initial value
init_slip_step	Input	4*DQS_WIDTH	Initial value for dqs gate coarse adjustment position. There are a total of DQS_WIDTH 4-bit groups that can be configured independently.
init_read_clk_ctrl	Input	3*DQS_WIDTH	Initial value for DQS gate fine adjustment position. There are a total of DQS_WIDTH 3-bit groups that can be configured independently.


3.2.4.5 Read Calibration Issues

After completing the DQS gate calibration and the write leveling calibration, the next step is to

ensure reliable DQ data sampling by DQS, which is the read calibration stage. During the read calibration, the PHY returns the pre-written 1-0-1-0 alternating fixed data using the DDR3 multi-purpose register (MPR), which contains the training patterns for the sampling points between DQS and DQ during read operations.

The read calibration phase features DQS centring calibration. The PHY separately calculates the boundaries and sizes of the DQS sampling windows on the rising edge and falling edge. Through single-step adjustment of the DQS delay, the optimal sampling position is reached, and the DQS centring calibration is complete.

When the initialization calibration fails and stays in the read calibration state, the status of the read calibration state machine `dbg_rdc` can be viewed as shown in [Figure 3-12](#).



```
calib_rdc_state is: 0
```

Figure 3-12 Read Calibration State Value

If it is in the `RDCAL_ERROR` state, users need to check the status of the sub-state machine `dbg_slice_rdc_state` and verify the status consistency of all groups.

Generally, large eye widths are supported for specific pattern types of MPR. Read calibration failures may result from voltage and temperature changes or SI distortions due to termination matching resistors. In this case, try changing the V_{REF} or V_{CCIO} voltages or improving the slew rates of DQ and DQS, such as adjusting FPGA termination matching resistors and enhancing the memory chip driving strength.

If conditions permit, it is recommended to use high-quality probes and oscilloscopes to capture the DQ and DQS signals in the proximity of the FPGA and memory interface balls during the read calibration, and observe the RX Mask and other signal integrity-related characteristic parameters of the DQ and DQS signals.

3.2.4.6 Eye Calibration Issues

On the basis of read calibration, eye calibration further calibrates the DQS sampling points by reading back more complex pseudo-random sequence pattern data to derive more accurate DQS sampling positions. Similar to read calibration, eye calibration also features DQS centring

calibration. The PHY separately calculates the boundaries and sizes of the DQS sampling windows on the rising edge and falling edge. Through single-step adjustment of the DQS delay, the optimal sampling position is reached, and the DQS centring calibration is complete.

When the initialization calibration fails and stays in the eye calibration state, the status of the eye calibration state machine `dbg_eyecal` can be viewed as shown in [Figure 3-13](#).

```
calib_main_state is: 6
calib_init_state is: 0
calib_wrlvl_state is: 0
calib_rdcval_state is: 0
calib_wrcal_state is: 0
calib_eyecal_state is: 8
calib_upcal_state is: 0
calib_error is: 0
```

Figure 3-13 Eye Calibration State Value

In the event of eye calibration failures, please verify that the calibration values of the previous stage are in the normal range. If so, the problem may be caused by changes in voltage and temperature or SI distortions stemming from termination matching resistors. In this case, try changing the V_{REF} or V_{CCIO} voltages or improving the slew rates of DQ and DQS, such as adjusting FPGA termination matching resistors and enhancing the memory chip driving strength. Otherwise, it may be attributed to abnormalities in the write leveling or DQS gate calibration stage. Please refer to "[3.2.4.3 Write Leveling Calibration Issues](#)" and "[3.2.4.4 DQS Gate Issues](#)".

If conditions permit, it is recommended to use high-quality probes and oscilloscopes to capture the DQ and DQS signals in the proximity of the FPGA and memory interface balls during the read calibration, and observe the RX Mask and other signal integrity-related characteristic parameters of the DQ and DQS signals.

3.2.5 Bit Error Issues

3.2.5.1 Rules Review

Usually, a routine design check should be performed when a failure occurs during the calibration, as described in section "[3.2.3 General Inspection](#)" of "[Chapter 3 Application Debugging FAQ](#)". High-speed memory interface signals generally rely heavily on reliable PCB board-level designs. If

design rule conflicts exist, they could be the root cause of data bit errors.

3.2.5.2 Bit Error Reproduction

Generally, when bit errors occur during normal read and write processes, the example project provided by the IP can be used to reproduce the bit error. For example, verify by configuring the Example Design to send different patterns and the debugging status information after all current initialization calibrations complete is output. In addition, one of the key steps for bit error debugging and analysis is determining the positions and types of bit errors based on a comparison between written and read data.

The repeat read function of the Example Project is often used to find the specific cause of DDR read/write errors. By comparing data read multiple times from the same address, it can be determined whether the data at that address is a write error or a read error. The fixed pattern of data can trigger errors with codes prone to errors, improving the efficiency of locating bugs. See [Table 3-1](#) and [Table 3-2](#) for BIST-related debug ports.

Using [Figure 3-14](#) as an example for error code reproduction, the example has a bit width of x16 and uses a counter pattern. The printed data is burst data with a burst length of 8. When the bit position of each error is fixed, as shown in [Figure 3-14](#), it can be initially determined as a single bit error of DQ2 and occurs on the 4th beat of a burst. The meaning of err_bit is to mark the error bit in a burst as 1.

hope_data_0:	hope_data_1:
0x4a4a4949	0x4e4e4d4d
0x4c4c4b4b	0x50504f4f
err_data_0:	err_data_1:
0x4a4a4949	0x4e4e4d4d
0x4c4c4b4b	0x50504f4f
err_bit_0:	err_bit_1:
0x0	0x0
0x40000	0x0

Figure 3-14 Error Code Example

3.2.5.3 Bit Error Type Identification

Whether using the Example Design provided by the IP or the user design, it is necessary to determine the positions and time points of bit errors and compare the known actual data with the

expected data. The bit error types can be distinguished based on the test conditions from the following aspects:

➤ Determine the characteristics of the bit-error data

Does the error in the data lie in single bits or bytes? Does it occur repeatedly? Classification, phenomena analysis, and potential debugging or resolution solutions.

Does the error belong to a specific DQS group or only appear in a specific DQ bit?

Sampling data shift? Half-byte interleaving? Completely irregular?

➤ Does the bit error only occur when accessing a specific address or segment within a bank or a certain bank in the memory?

The causes are possibly that, the user design may have overlooked the logic switching resulting from address expansion or the activation time fails to meet the timing requirements outlined in the memory device datasheet.

➤ Does the error only occur with a certain data pattern or sequence? Is it fixed to flip from 0 to 1, or 1 to 0? Is the error code fixed at a constant 0 or 1?

This phenomenon indicates that the memory interface signals on the PCB may have shorted or open connections, or the presence of crosstalk or other simultaneous switching output noises.

➤ Determine the conditions and probability of bit error recurrence

Does the bit error occur at every power-up initialization calibration? What is the probability of the bit error?

Does the bit error only occur after every soft or hard reset? What is the probability of the bit error?

Does it only occur under a specific temperature or voltage?

➤ Can the bit error be recovered through some means?

For example: repeated writes, repeated reads, FPGA reconfiguration, reinitialization, temperature or voltage alternation, etc.

3.2.5.4 Distinguishing Bit Error Sources

As errors in the write direction can cause incorrect data during readback, it becomes challenging to ascertain the specific source of the errors. Additionally, when the timing requirements for addresses or control commands are not met, read/write bit errors may also appear. The following methods can help designers discover and identify existing issues:

➤ Confirmation of initialization calibration results

The hardware environment may cause errors in the initialization calibration process and in some cases may even result in passing the initialization process but obtaining incorrect initialization calibration results. When error codes occur, key debugging information from debug_data should be read to ensure that all initialization calibration results are as expected and accurate, primarily comparing differences in calibration results between groups in terms of window values, delay values, DQS adjustment values, etc., see [Table 2-2](#).

➤ Eliminate "false error codes"

When conditions allow, use the example design provided by the IP as much as possible instead of debugging error code issues based on user designs, in order to improve debugging efficiency and eliminate "false error codes" caused by user logic errors. Additionally, localisation can be performed by simplifying the design, reducing the DQ width, and decreasing the interface rate. Simplifying the design includes testing only the DDR interface as much as possible and removing unrelated logic to eliminate other unnecessary interference. If error codes disappear after reducing the DQ width or the interface rate, then the focus of debugging should be on how to improve signal integrity.

➤ Read-write separation

In the event of intermittent errors, the repeated reading approach can be used for read-write separation. The recommended option is to write a few burst data segments of appropriate length at the initial stage. The pattern can be based on either the data observed with bit errors or the data that can reproduce the same error. Subsequently, continuous data access at this position may result in the following two circumstances:

If bit errors appear intermittently in the readback data, it is highly indicative of a potential issue in the read direction;

If the readback data consistently shows the same error, it implies a potential problem in the write direction.

➤ Capturing DQ and DQS Signals

If conditions permit, use high-quality probes and oscilloscopes to capture the DQ and DQS signals in the proximity of the memory during write operations to the fullest extent, or the DQ and DQS signals in the proximity of the FPGA and memory interface balls during read operations. To capture appropriate DQS and DQ waveforms, it is usually necessary to capture from the segment between the DQS preamble and the valid data state, and observe the data sampling accuracy, the phase relationship between DQS and DQ, and other signal integrity-related parameters. By adjusting the values of DCI and ODT (e.g., enhancing drive

strength and reducing termination impedance) and comparing different measurement results, the optimal eye diagrams can be obtained at the FPGA receiver.

➤ **Analysing Read and Write Margins**

The debug information provided through the debugging interface of the IP allows for easy acquisition of read or write margins during the initialization calibration. As is seen from the window margins of the DQS rising and falling edges during the read calibration that are shown in the debugging results previously described, accurate window margins are usually derived by using complex patterns during the calibration. Users can try adjusting reference voltage, output impedance, and termination impedance to increase the eye diagram window. However, before making adjustments, it is essential to determine whether it is a read error code or a write error code to minimize adjustment variables.

Application Example for Reference Only

Disclaimer

Copyright Notice

This document is copyrighted by Shenzhen Pango Microsystems Co., Ltd., and all rights are reserved. Without prior written approval, no company or individual may disclose, reproduce, or otherwise make available any part of this document to any third party. Non-compliance will result in the Company initiating legal proceedings.

Disclaimer

1. This document only provides information in stages and may be updated at any time based on the actual situation of the products without further notice. The Company assumes no legal responsibility for any direct or indirect losses caused by improper use of this document.
2. This document is provided "as is" without any warranties, including but not limited to warranties of merchantability, fitness for a particular purpose, non-infringement, or any other warranties mentioned in proposals, specifications, or samples. This document does not grant any explicit or implied intellectual property usage license, whether by estoppel or otherwise.
3. The Company reserves the right to modify any documents related to its family's products at any time without prior notice.
4. The information provided in this document is intended to assist users in resolving application issues; however, it does not guarantee flawlessness. The Company disclaims any responsibility for functional abnormalities or performance degradation that may occur due to the user's failure to follow the methods outlined in this document, and such issues cannot be acknowledged as product-related. Moreover, the solutions presented in this document are merely one of several possible options, and no guarantee is provided for their applicability to all application scenarios. In the event of any functional abnormalities or performance degradation resulting from user implementation of the instructions outlined in this document, the Company will not accept responsibility or acknowledge them as product-related issues.