DDR3 Soft Controller Core User Guide

UG-CORE-DDR3-CONTROLLER-v1.6
October 2023
www.elitestek.com

Contents

Introduction

The DDR3 Soft Controller Core is a memory controller core that interfaces with industry standard DDR SDRAM modules. The core also handles the timing parameters, priority, and other memory operations with DDR3 modules.

Features

- Mode register set customization
- Fully parameterized to be compatible with any DDR3 device
- Supports x8/x16 DRAM memory
- Data rate of up to 800 MT/s (x16 devices)
- Supports burst length of 8
- Native and AXI4 interface
- DQ input auto-calibration
- Supports:
  - 1 rank support
  - Bank interleaving
  - Half-rate mode
  - Random access within the same row
- Verilog HDL RTL and simulation testbench

Device Support

Table 1: DDR3 Soft Controller Core Device Support

FPGA Family	Supported Device
Trion	–
钛金系列	All

Resource Utilization and Performance

Note: The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and change depending on the device resource utilization, design congestion, and user design.

Table 2: 钛金系列 Resource Utilization and Performance

Using TIMING_2 optimization in Efinity®'s Place-and-Route.

FPGA	Mode	Logic and Adders	Flip-flops	Memory Blocks	Efinity® Version
Ti180 M484 C4	AXI4	2,883	2,129	27	2022.2
	Native	3,290	2,019	27	2022.2

(1) Using Verilog HDL.

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes is available in the Efinity Downloads page under each Efinity software release version.

Note: You must be logged in to the Support Portal to view the IP Core Release Notes.

Functional Description

The DDR3 Soft Controller core consists of the following blocks:

- Instruction RAM—Instruction memory for the main controller behavior
- Main Controller—Main state machine of the DDR3 controller
- Arbiter—Handles the write and read prioritization
- Buffer—64 depth buffer for write and read data
- Input Auto-Calibration—For DQ input clock (tac_clock) calibration

Figure 1: DDR3 Soft Controller Native Block Diagram

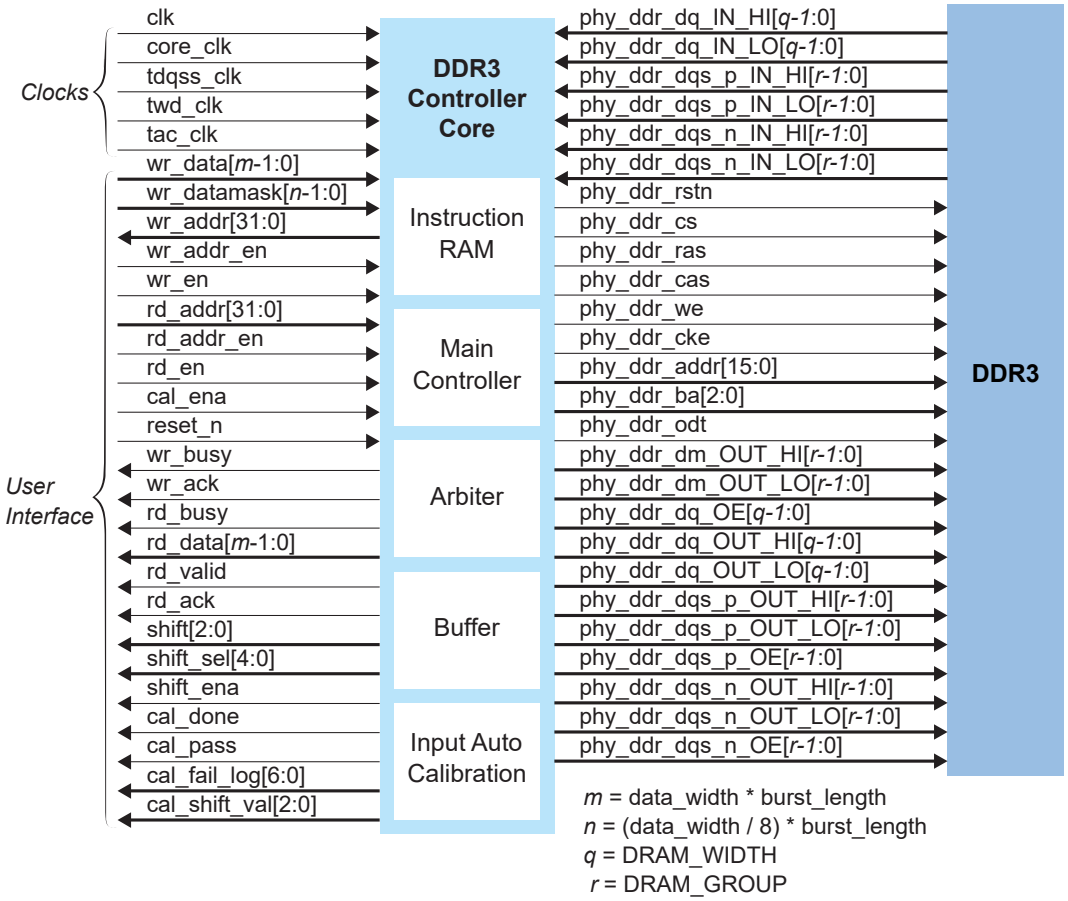
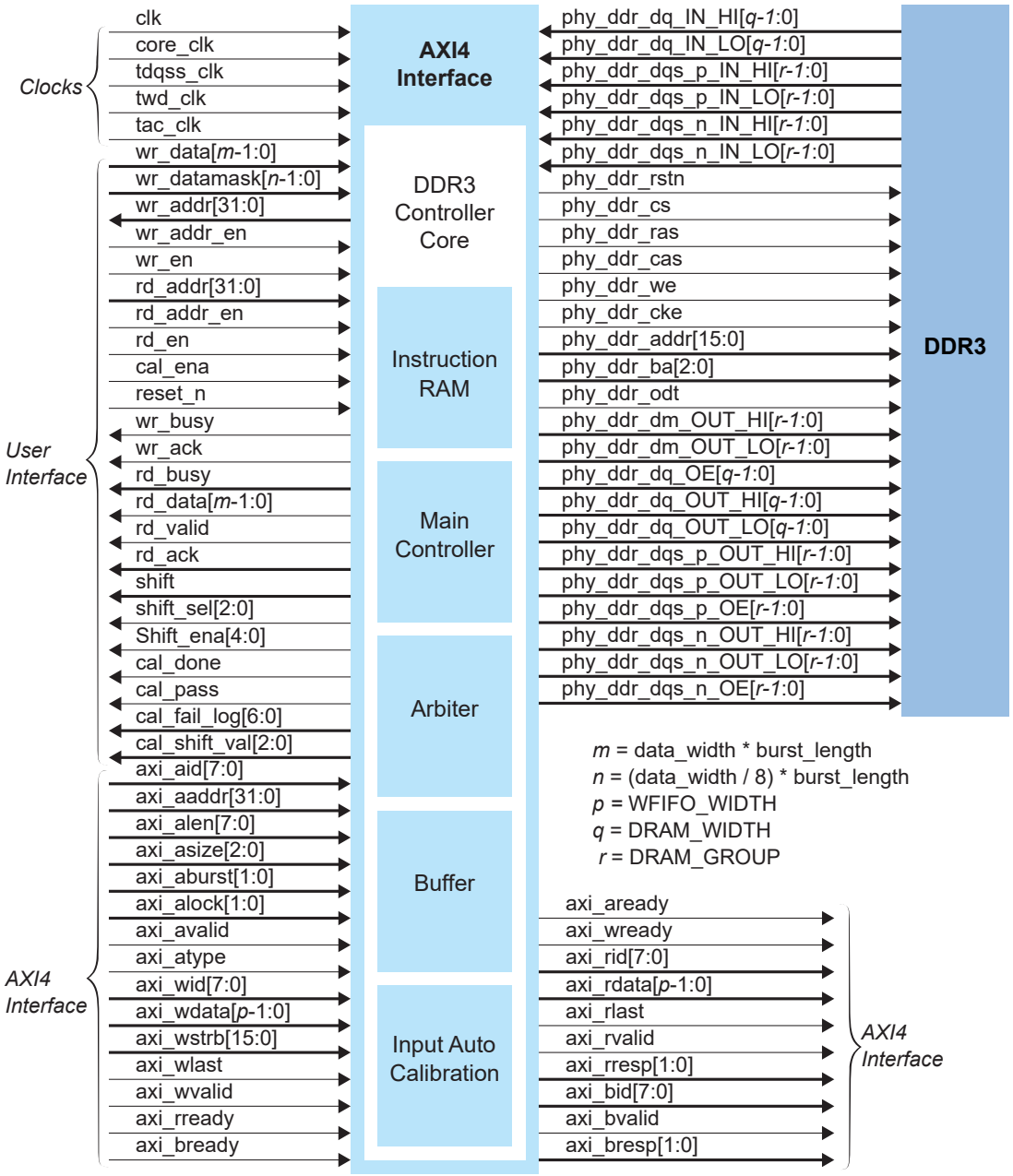


Figure 2: DDR3 Soft Controller with AXI4 Interface Block Diagram



Ports

Table 3: Clocks

Port	Direction	Description
clk	Input	Controller clock for user interface.
core_clk	Input	Main controller (state machine) clock frequency.
tdqss_clk	Input	Output clock for DDR's address, control and DQS signals. Same as PHY clock frequency.
twd_clk	Input	DQ/DM output write clock. Same as PHY clock frequency.
tac_clk	Input	DQ/DQS input read clock. Same as PHY clock frequency.

Table 4: User Interface

m = DRAM_WIDTH*BL

n = GROUP_WIDTH/8*BL

Port	Direction	Description
wr_data[m-1:0]	Input	Data to be written to DRAM.
wr_datamask[n-1:0]	Input	Datamask to be written to DRAM.
wr_addr[31:0]	Output	Write address (byte) to DRAM in [ROW, BANK, COL].
wr_addr_en	Input	Enable to write wr_addr.
wr_en	Input	Write data with wr_data and wr_datamask.
rd_addr[31:0]	Input	Read address (byte) to DRAM in [ROW, BANK, COL].
rd_addr_en	Input	Enable to read rd_addr.
rd_en	Input	Ready for read.
reset_n	Input	Active low reset for the controller.
wr_busy	Output	Busy status of the write buffer is full. The core cannot perform a write operation when this signal is asserted.
wr_ack	Output	Write acknowledge. Handshaking with write enable(wr_en).
rd_busy	Output	Busy status of the read buffer is full. The core cannot perform a read operation when this signal is asserted.
rd_data[m-1:0]	Output	Data read from DRAM. o_rd_data shares the same data address mapping as wr_data.
rd_valid	Output	Read valid. The data on o_rd_valid is valid to be read.
rd_ack	Output	Read acknowledge. Handshaking with read address enable (rd_addr_en).
shift[2:0]	Output	PLL auto-calibration value.[2]
shift_sel[4:0]	Output	Selects the clock for PLL auto-calibration for PLL (tac_clk only).[2]
shift_ena	Output	PLL auto-calibration for PLL enabled.[2]
cal_ena	Input	Assert this signal to enable the auto-calibration function.
cal_done	Output	Auto-calibration function done.[2]
cal_pass	Output	Logic high indicates the auto-calibration function passed.
cal_fail_log[6:0]	Output	Logic high indicates the calibration result of phase shift failed. Bit[6]: Phase shift value 6 PASS:0 FAIL:1 Bit[5]: Phase shift value 5 PASS:0 FAIL:1 Bit[4]: Phase shift value 4 PASS:0 FAIL:1 Bit[3]: Phase shift value 3 PASS:0 FAIL:1 Bit[2]: Phase shift value 2 PASS:0 FAIL:1 Bit[1]: Phase shift value 1 PASS:0 FAIL:1 Bit[0]: Phase shift value 0 PASS:0 FAIL:1
cal_shift_val[2:0]	Output	Current phase shift value after calibration.

[2] Connect this signal to PLL in the Interface Designer if you are using the auto-calibration function.

Table 5: AXI4 Interface

p = WFIFO_WIDTH

Port	Direction	Description
axi_aid[7:0]	Input	Read or write address channel transaction ID
axi_aaddr[31:0]	Input	Read or write address channel address.
axi_alen[7:0]	Input	Read or write address channel burst length.
axi_asize[2:0]	Input	Read or write address channel transfer size.
axi_aburst[1:0]	Input	Read or write address channel burst type.
axi_alock[1:0]	Input	Read or write address channel locked transaction.
axi_avalid	Input	Read or write address channel valid
axi_atype	Input	0: Change all axi_ax signals for read address channel 1: Change all axi_ax signals for write address channel
axi_wid[7:0]	Input	Write channel transaction ID
axi_wdata[p-1:0]	Input	Write channel data
axi_wstrb[15:0]	Input	Write channel strobe (Single bit represents data byte)
axi_wlast	Input	Write channel last data
axi_wvalid	Input	Write channel valid
axi_rready	Input	Read data channel ready
axi_bready	Input	Write response channel ready
axi_aready	Output	Read or write address channel ready
axi_wready	Output	Write channel ready
axi_rid[7:0]	Output	Read data channel transaction ID
axi_rdata[p-1:0]	Output	Read data channel data
axi_rlast	Output	Read data channel last data
axi_rvalid	Output	Read data channel valid
axi_rresp[1:0]	Output	Read data channel response
axi_bid[7:0]	Output	Write response channel transaction ID
axi_bvalid	Output	Write response channel valid
axi_bresp[1:0]	Output	Write response channel response

Table 6: DDR Interface

q = DRAM_WIDTH

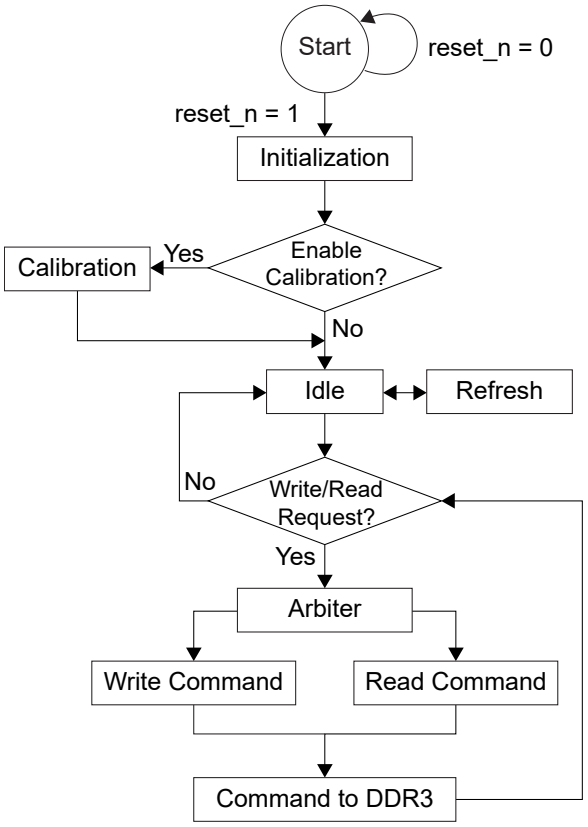
r = DRAM_GROUP

Port	Direct	Clock Domain	Description
phy_ddr_dq_IN_HI[q-1:0]	Input	tac_clk	DDR3 DQ input for DDIO HI.
phy_ddr_dq_IN_LO[q-1:0]	Input	tac_clk	DDR3 DQ input for DDIO LO.
phy_ddr_dqs_n_IN_HI[r-1:0]	Input	tac_clk	DDR3 DQS# input for DDIO HI.
phy_ddr_dqs_n_IN_LO[r-1:0]	Input	tac_clk	DDR3 DQS# input for DDIO LO.
phy_ddr_dqs_p_IN_HI[r-1:0]	Input	tac_clk	DDR3 DQS input for DDIO HI.
phy_ddr_dqs_p_IN_LO[r-1:0]	Input	tac_clk	DDR3 DQS input for DDIO LO.
phy_ddr_addr[15:0]	Output	tdqss_clk	DDR3 A[15:0].
phy_ddr_ba[2:0]	Output	tdqss_clk	DDR3 BA[2:0].
phy_ddr_cas	Output	tdqss_clk	DDR3 CAS#.
phy_ddr_cke	Output	tdqss_clk	DDR3 CKE.
phy_ddr_cs	Output	tdqss_clk	DDR3 CS#.
phy_ddr_dm_OUT_HI[r-1:0]	Output	twd_clk	DDR3 DM for DDIO HI.
phy_ddr_dm_OUT_LO[r-1:0]	Output	twd_clk	DDR3 DM for DDIO LO.
phy_ddr_dq_OE[q-1:0]	Output	twd_clk	DDR3 DQ output enable.
phy_ddr_dq_OUT_HI[q-1:0]	Output	twd_clk	DDR3 DQ output for DDIO HI.
phy_ddr_dq_OUT_LO[q-1:0]	Output	twd_clk	DDR3 DQ output for DDIO LO.
phy_ddr_dqs_n_OE[r-1:0]	Output	tdqss_clk	DDR3 DQS# output enable.
phy_ddr_dqs_n_OUT_HI[r-1:0]	Output	tdqss_clk	DDR3 DQS# output for DDIO HI.
phy_ddr_dqs_n_OUT_LO[r-1:0]	Output	tdqss_clk	DDR3 DQS# output for DDIO HI.
phy_ddr_dqs_p_OE[r-1:0]	Output	tdqss_clk	DDR3 DQS output enable.
phy_ddr_dqs_p_OUT_HI[r-1:0]	Output	tdqss_clk	DDR3 DQS output for DDIO HI.
phy_ddr_dqs_p_OUT_LO[r-1:0]	Output	tdqss_clk	DDR3 DQS output for DDIO HI.
phy_ddr_odt	Output	tdqss_clk	DDR3 ODT.
phy_ddr_ras	Output	tdqss_clk	DDR3 RAS#.
phy_ddr_rstn	Output	tdqss_clk	DDR3 RESET#.
phy_ddr_we	Output	tdqss_clk	DDR3 WE#.

State Machine

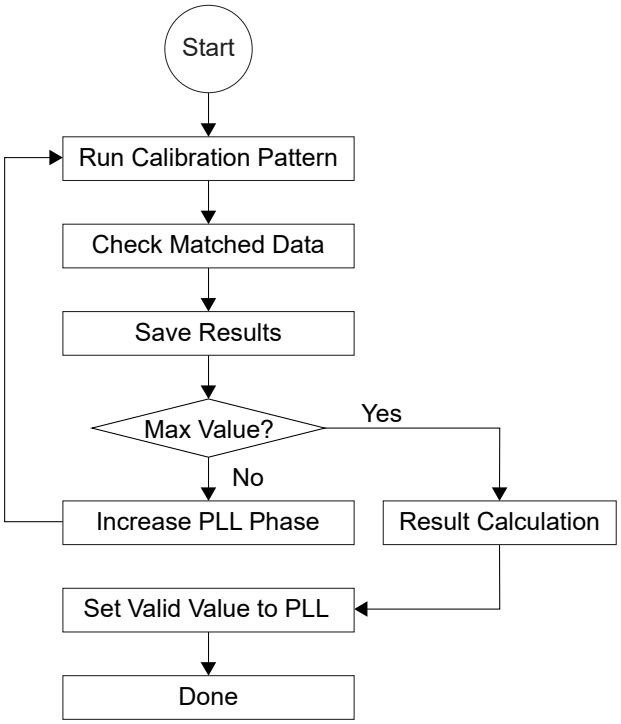
The DDR3 Soft Controller core starts initialization when you assert the `reset_n` signal. Next, the core runs the auto-calibration process if you enable the auto-calibration (`cal_ena=1`) function. See **Input Auto-Calibration** on page 11 for more information about the auto-calibration process. When the core receives a write or read request, the arbiter prioritizes either read or write based on the arbiter parameter settings you set. The core then sends the command to the DDR3 SDRAM.

Figure 3: DDR3 Soft Controller State Machine



Input Auto-Calibration

The input auto-calibration block starts the calibration by increasing the phase with `tac_clk` (read data clock) a step at a time. The auto-calibration block then writes and reads data from DDR3 starting from address `0x0000` to `0x1000` (4 KB). At the same time, the block compares the data from DDR3 matches the write data. After exhausting all possible phase increment, the auto calibration block calculates the best valid value and write out the PLL setting.



Arbiter

The DDR3 Soft Controller core includes an arbiter that manages the priority for write and read operations. The arbiter also controls the number of consecutive read or write operations before switching to another read or write operation.

The following waveforms illustrates the examples of read and write operation based on predefined arbiter parameters.

Figure 4: arbiter_init = 0, arbiter_count = 4 Example Waveform

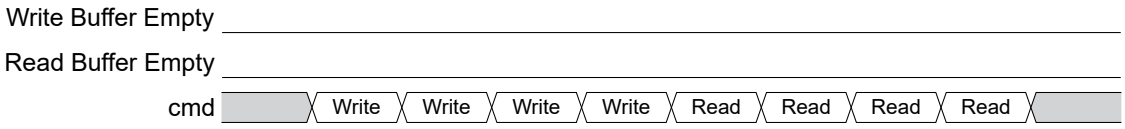


Figure 5: arbiter_init = 1, arbiter_count = 1 Example Waveform

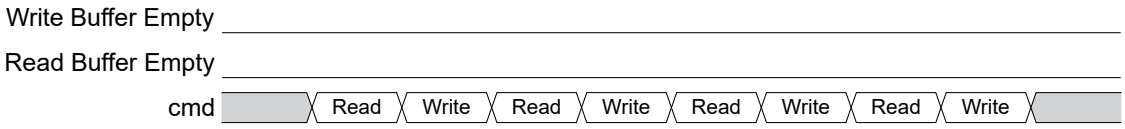


Figure 6: arbiter_init = 1, arbiter_count = 1 Example Waveform (with Write Buffer Empty)

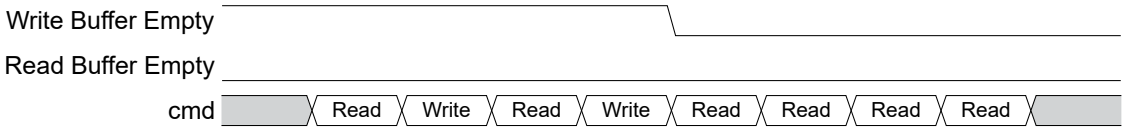
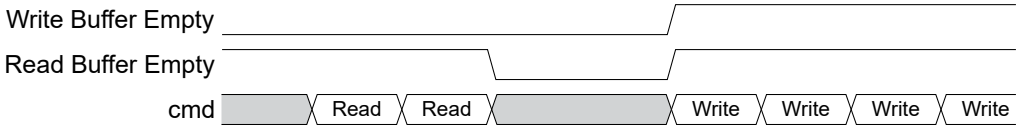


Figure 7: arbiter_init = 0, arbiter_count = 8 Example Waveform (with Write Buffer Empty and Read Buffer Empty)

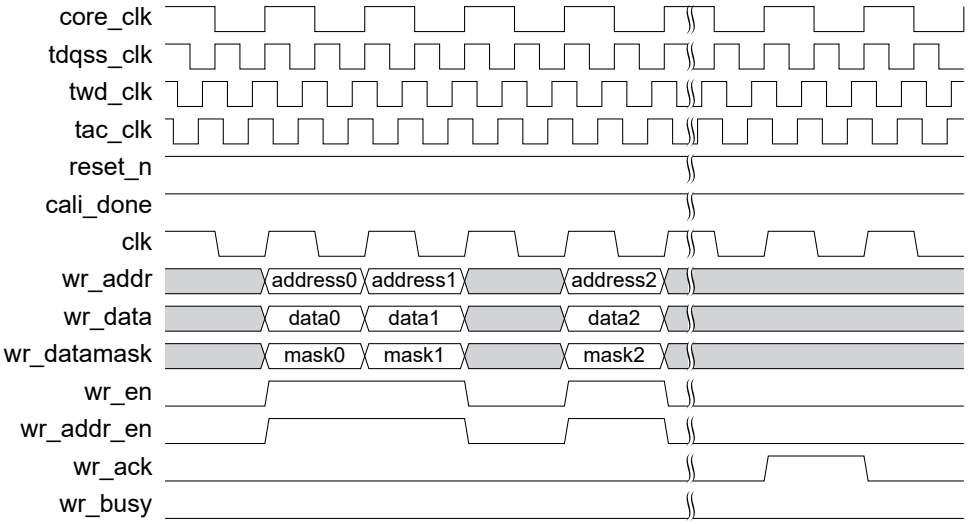


Operation Examples

The following waveforms illustrate examples of write and read operations.

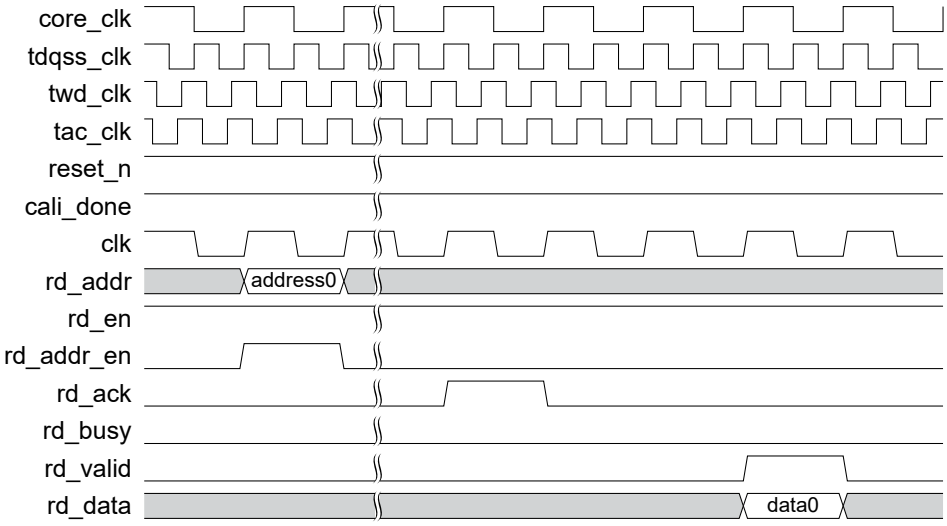
The write operation starts when you assert the `wr_en` signal high. The write operation stops when the `wr_en` signal is de-asserted. The DDR3 Soft Controller core asserts the `wr_ack` signal when the data is successfully written into the DDR3 memory.

Figure 8: Write Operation Example Waveform



The read operation starts when you assert the `rd_en` signal. The DDR3 Soft Controller core asserts the `rd_ack` to acknowledge that a read operation at the specified address is received. The read data is available at the `rd_data` port when the `rd_valid` is asserted.

Figure 9: Read Operation Example Waveform



Read Enable Pipeline

The **Read Enable Pipeline** parameter is an additional pipeline on top of the memory Read Latency parameter settings. You set this parameter to account for:

- Board trace delays
- DDIO delays
- Finding the word boundary to start de-serializing the incoming read data. For example, for a x16 DRAM, the data is de-serialized to AXI data width of 128 bits.

The DDR3 Soft Controller core includes an example design targeting the Ti180 M484 Development Board. In this example design, the Read Enable Pipeline parameter is set to 8. For your own board and design, you need to adjust the value to determine the correct settings.

The example design includes a memory checker which compares the expected (`rdata_store`) and the actual data (`rdata`). You can use this memory checker as a reference for your board. In the memory checker, the actual read data is available when `rready` and `rvalid` signals are high. The actual read data should be the same as the expected read data if the Read Enable Pipeline is set to the correct word boundary. Each step of the Read Enable Pipeline value shifts the word boundary by:

- 32 bits for DRAM x16
- 16 bits for DRAM x8

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate 易灵思® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an 易灵思 development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.

Note: Not all 易灵思 IP cores include an example design or a testbench.

Generating the DDR3 Soft Controller Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **Memory Controllers** > **DDR3 Soft Controller** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.

   Note: You cannot generate the core without a module name.
4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the Customizing the DDR3 Soft Controller section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an 易灵思® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.

   Note: You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.
9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:

- **<module name>_define.vh**—Contains the customized parameters.
- **<module name>_tmpl.v**—Verilog HDL instantiation template.
- **<module name>_tmpl.vhd**—VHDL instantiation template.
- **<module name>.v**—IP source code.
- **settings.json**—Configuration file.
- **<kit name>_devkit**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains generated RTL and testbench files.

Customizing the DDR3 Soft Controller

The core has parameters so you can customize its function. You set the parameters in the General tab of the core's IP Configuration window.

Table 7: Parameters (General Tab)

Parameter	Options	Description
Simulation	yes, no	Select to generate either the simulation testbench or example design. yes: generate testbench no: generate example design (default)
DDR3 Interface	Native, AXI4	Select the DDR3 interface type. Default: AXI4
Read Enable Pipeline	6 - 8	Set the number of additional pipelines for read enable. Read Enable Pipeline + read latency (tRL) must be less than 32. See **Read Enable Pipeline** on page 14. Default: 8
DDR3 Data Rate	800D, 800E	Select the DDR3 data rate. Default: 800D
DDR3 DQ Width	8, 16	Select the DDR DQ total width. Default: 16
DDR Frequency	100 - 400	Set DDR running speed up in MHz. Default: 400
Column	10, 11	Column address bit width for reference. Value is automatically set depending on the selected DQ width. DQ Width 8: 11 DQ Width 16: 10
Row	16	Row address bit width for reference. This value is fixed.
Bank	3	Bank address bit width for reference. This value is fixed.
Rank	1	Number of ranks for reference. This value is fixed.

Table 8: Parameters (MR0 Tab)

Parameter	Options	Description
Burst Length	8	Burst length for reference. This value is fixed.
Read Burst Type	sequential, interleaved	Set the read burst type. Default: sequential
CAS Latency (CL)	5, 6	Set the CAS latency value. Default: 5
DLL Reset	yes	DLL reset function. This value is fixed.
Write Recovery (tWR)	6	Write recovery value. This value is fixed.
Pre-charge Power-down	on	Pre-charge power-down function. This value is fixed.

Table 9: Parameters (MR1 Tab)

Parameter	Options	Description
DLL Enable	true	DLL Enable function. This value is fixed.
Output Drive Strength	rzq/6, rzq/7	Set the output drive strength. Default: rzq/6

Parameter	Options	Description
Posted CAS Additive Latency (AL)	disable, cl-1, cl-2	Set the posted CAS additive latency value. Default: disable
Write Leveling	disable	Write leveling function. This value is fixed.
Termination Data Strobe (TDQS)	disable	Termination data strobe (TDQS) function. This value is fixed.
Output Enable	enable	Output enable function. This value is fixed.
On-die Termination (ODT) Resistance	disable, rzq/2, rzq/4, rzq/6, rzq/8, rzq/12	Set the dynamic on-die termination (ODT) function. Default: rzq/6

Table 10: Parameters (MR2 Tab)

Parameter	Options	Description
CAS Write Latency (CWL)	5	CAS write latency (CWL) for reference. This value is fixed.
Auto Self Refresh	disable	Auto self refresh function. This value is fixed.
Self Refresh Temperature (SRT)	normal, extended	Set the self refresh temperature (SRT). Default: normal
Dynamic On-die Termination (ODT)	disable, rzq/2, rzq/4	Set the dynamic on-die termination (ODT) function. Default: disable

Table 11: Parameters (MR3 Tab)

Parameter	Options	Description
Multipurpose Register Read Function (MPRRF)	0	Multipurpose register (MPR) read function. This value is fixed.
Multipurpose Register	normal, enable	Set the multipurpose register function. Default: normal

Table 12: Parameters (Timing Tab)

These values are automatically set depending on the DDR3 data rate you select except for tREFI.

Parameter	Options	Description
tCL	5	CAS latency (in CK) for reference.
tCWL	5	CAS write latency (in CK) for reference.
tAL	0	Additive latency (in CK) for reference.
tCCD	4	Command to command period (in CK).
tRTP	4	Read to precharge period (in CK) for reference.
tWR	6	Write recovery.
tRCD	12.5	Activate-to-internal read/write delay (in ns) for reference.
tRAS	37.5	Activate-to-precharge command period (in ns) for reference.
tREFI	1 - 7800	Maximum average periodic refresh (in ns). Depend on the DDR3 800D and 800E specification. Default: 7800
tRC	50	Activate-to-activate/refresh command period (in ns) for reference.
tRFC	350	Refresh-to-activate/refresh command period (in ns) for reference.
tWTR	4	Delay from start of internal write transaction to internal read command (in CK) for reference.
tMRD	4	Mode register set command cycle time (in CK) for reference.
tRP	12.5	Precharge command period (in ns) for reference.
tMOD	12	Mode register set command update delay (in CK) for reference.
ODTH8	6	ODT high time with write command and BL8 (in CK) for reference.

Table 13: Parameters (Arbiter Tab)

Parameter	Options	Description
Arbiter Init	0, 1	Select to prioritize write or read operation. 0: Write 1: Read (default)
Arbiter Count	1 - 16	Set the number of read or write command before switching priority to write or read command. Default: 8

Interface Designer Settings

When creating your own DDR3 Soft Controller core design, you need to create GPIO blocks/buses and PLL output clocks in the Efinity Interface Designer with the settings shown in following tables.

Table 14: Interface Designer Settings for PLL

Option	Output Clock 0	Output Clock 1	Output Clock 2	Output Clock 3	Output Clock 4
Instance Name	tdqss_clk	core_clk	tac_clk	twd_clk	Feedback clk
Clock Frequency (MHz)	400	200	400	400	50
Phase Shift	0	0	Dynamic	90	0

Table 15: Interface Designer Settings for GPIO

Block/Bus	Settings					
	Mode	I/O standard	Register Option	Double Data I/O Option	Clock	Inverted Clock
clk_p	clkout	1.5 V Differential SSTL	-	-	tdqss_clk	Yes
cke	output	1.5 V SSTL	register	-	tdqss_clk	-
Addr[15:0]	output	1.5 V SSTL	register	-	tdqss_clk	-
Ba[2:0]	output	1.5 V SSTL	register	-	tdqss_clk	-
cs	output	1.5 V SSTL	register	-	tdqss_clk	-
ras	output	1.5 V SSTL	register	-	tdqss_clk	-
cas	output	1.5 V SSTL	register	-	tdqss_clk	-
we	output	1.5 V SSTL	register	-	tdqss_clk	-
i_dq_hi[15:0	input	1.5 V SSTL	register	resync	tac_clk	-
i_dq_lo[15:0]	input	1.5 V SSTL	register	resync	tac_clk	-
o_dq_hi[15:0]	output	1.5 V SSTL	register	resync	twd_clk	-
o_dq_lo [15:0]	output	1.5 V SSTL	register	resync	twd_clk	-
o_dq_oe[15:0]	output	1.5 V SSTL	register	-	twd_clk	-
i_dqs_hi	input	1.5 V Differential SSTL	register	resync	tac_clk	-
i_dqs_lo	input	1.5 V Differential SSTL	register	resync	tac_clk	-
o_dqs_hi	output	1.5 V Differential SSTL	register	resync	tdqss_clk	-
o_dqs_lo	output	1.5 V Differential SSTL	register	resync	tdqss_clk	-
o_dqs_oe	output	1.5 V Differential SSTL	register	-	tdqss_clk	-
o_dqs_n_oe	output	1.5 V Differential SSTL	register	-	tdqss_clk	-
o_dm_hi	output	1.5 V SSTL	register	resync	twd_clk	-
o_dm_lo	output	1.5 V SSTL	register	resync	twd_clk	-
odt	output	1.5 V SSTL	register	-	tdqss_clk	-
reset	output	1.5 V SSTL	register	-	tdqss_clk	-

DDR3 Soft Controller Hardware Requirements

You need to observe the following hardware connection guidelines when creating your own design with the DDR3 Soft Controller core:

- The 钛金系列 FPGA `VREF` pin must be connected to the DDR3 VDD/2 for the DQ input signal.
- The DQS pins must be connected to FPGA I/O pins according to their polarity.

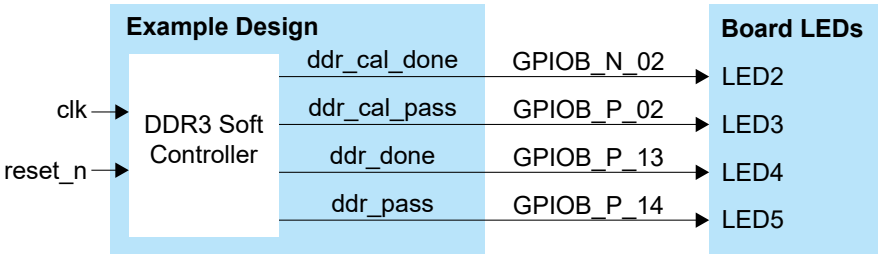
DDR3 Soft Controller Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

Important: 易灵思 tested the example design generated with the default parameter options only.

The example design targets the 钛金系列 Ti180 M484 Development Board with the FMC DDR3 and GPIO Daughter Card attached. The IP Manager generates the native or AXI4 mode example design depending on the DDR Interface parameter you select.

Figure 10: Example Design Block Diagram



The design performs calibration, write, read, and compares the write and read operations (memory checking). The design then outputs the operation results through the 钛金系列 Ti180 M484 Development Board LEDs.

Table 16: Example Design LEDs

LED	Description
LED2	DDR3 memory calibration done
LED3	DDR3 memory calibration pass
LED4	DDR3 memory checker done
LED5	DDR3 memory checker pass

Table 17: Example Design Implementation

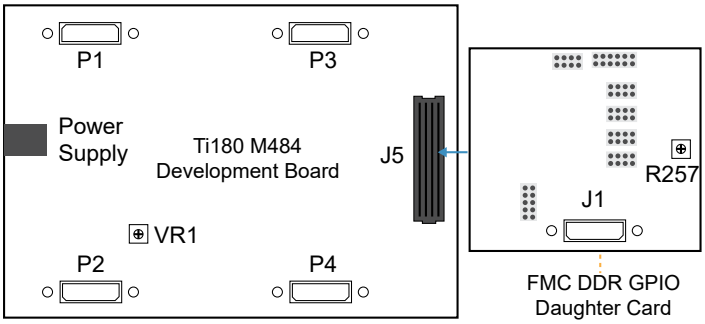
FPGA	Mode	Logic and Adders	Flip-flops	Memory Blocks	f_{MAX}	Efinity® Version[3]
Ti180 M484 C4	AXI4	2,912	2,124	27	[4]	2022.2
	Native	3,198	2,019	27		2022.2

[3] Using Verilog HDL.
[4] Clock out f_{MAX} matched the PLL settings in **Table 14: Interface Designer Settings for PLL** on page 19.

Attaching the FMC DDR3 and GPIO Daughter Card Daughter Card

The FMC DDR3 and GPIO Daughter Card attaches to the 钛金系列 Ti180 M484 Development Board. 易灵思® recommends that you also perform voltage tuning before using the FMC DDR3 and GPIO Daughter Card.

Figure 11: Attaching FMC DDR3 and GPIO Daughter Card



To connect a daughter card and tune the voltage supply:

1. Without attaching the FMC DDR3 and GPIO Daughter Card, turn on the 钛金系列 Ti180 M484 Development Board.
2. Probe the PT4 test point and use the potentiometer VR1 to tune the voltage to 1.35 V.
3. Turn off the 钛金系列 Ti180 M484 Development Board.
4. Ensure the power supply and board power switch are turned off, then connect FMC DDR3 and GPIO Daughter Card to the 钛金系列 Ti180 M484 Development Board.
5. Turn on the 钛金系列 Ti180 M484 Development Board.
6. Probe pin 1 or 2 on the J199 header and use potentiometer R257 to tune the voltage to 1.35 V.

DDR3 Soft Controller Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.

Note: You must include all **.v** files generated in the **/testbench** directory in your simulation.

易灵思 provides a simulation script for you to run the testbench quickly using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` in a terminal application. You must have Modelsim installed on your computer to use this script.

The IP Manager also generates source codes for you to simulate with different simulators.

Table 18: Testbench Files

Directory/File	Note
../Testbench/modelsim_native.do	Modelsim testbench script for native mode.
../Testbench/modelsim_AXI4.do	Modelsim testbench script for AXI4 mode.
../Testbench/modelsim	Contains the generated encrypted source code to simulate with the Modelsim simulator.
../Testbench/ncsim	Contains the generated encrypted source code to simulate with the NCSIM simulator.
../Testbench/synopsys	Contains the generated encrypted source code to simulate with the VCS simulator.

The IP Manager generates the native or AXI4 mode testbench depending on the DDR Interface parameter you select. The testbench performs a pattern-matching simulation as shown in the following diagrams.

Figure 12: Native Testbench Pattern Matching Flow Diagram

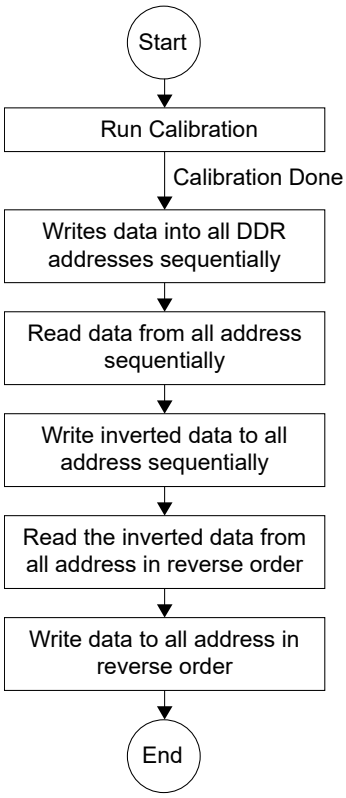
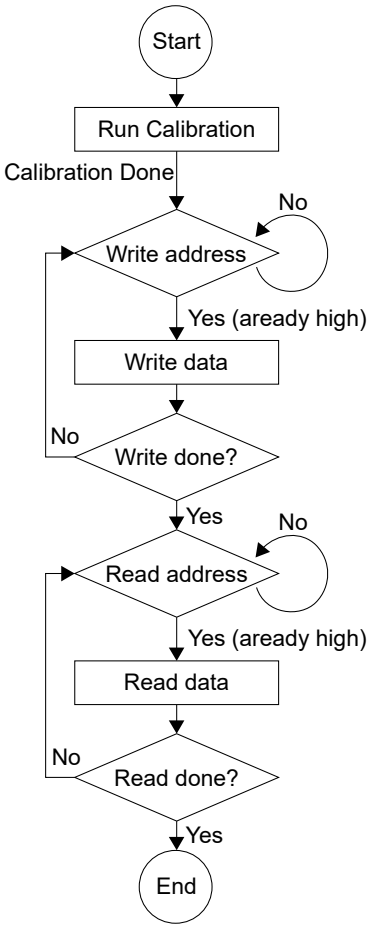


Figure 13: AXI4 Testbench Pattern Matching Flow Diagram



After running the simulation successfully, the test prints the following message:

```
efx_ddr3_soft_controller_tb.ddr3md.data_task: at time 24233850.0 ps INFO: READ @ DQS= bank = 0 row
 = 0000 col = 0000017e data = 0304
efx_ddr3_soft_controller_tb.ddr3md.data_task: at time 24235100.0 ps INFO: READ @ DQS= bank = 0 row
 = 0000 col = 0000017f data = 0102
DDR3 Controller PASS
```

Revision History

Table 19: Revision History

Date	Version	Description
October 2023	1.6	Corrected shift, shift_sel, and shift_ena signal widths in the block diagram. (DOC-1509)
June 2023	1.5	Added Device Support and release notes sections. (DOC-1234)
May 2023	1.4	Added Read Enable Pipeline section. (DOC-1241)
February 2023	1.3	Updated for Efinity IP Manager support. Added note about The resource and performance values in the resource and utilization table are for guidance only.
October 2022	1.2	Added READ_ENABLE_PIPELINE parameter.
July 2022	1.1	Added support for 钛金系列 Ti180 FPGA and AXI4 interface.
March 2022	1.0	Initial release.