SDRAM Controller Core User Guide

UG-CORE-SDRAM-v2.4
February 2023
www.efinixinc.com

Contents

Introduction

The SDRAM Controller core provides a simplified interface between the Trion® FPGA and SDRAM devices.

Use the IP Manager to select IP, customize it, and generate files. The SDRAM Controller core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix® development board.

Features

- Fully parameterized to be compatible with any SDRAM device
- Native mode or Advanced eXtensible Interface 4 (AXI4) mode user interface
- Half-rate or full-rate
- Memory burst length of 1
- Bank interleaving
- Random access within the same row
- Column access strobe (CAS) latency of 2 or 3
- Verilog HDL RTL and simulation testbench
- Includes an example design targeting the Trion® T20 BGA256 Development Board

FPGA Support

The SDRAM Controller core supports all Trion® FPGAs.

Resource Utilization and Performance

Note: The resources and performance values provided are just guidance and change depending on the device resource utilization, design congestion, and user design.

Trion Resource Utilization and Performance

FPGA	Logic Utilization (LUTs)		Memory Blocks		f_{MAX} (MHz)(1)		Efinity® Version
	Native	AXI4	Native	AXI4	Native	AXI4	
T8 QFP144 C4	649	678	0	2	116.5	106.4	2019.3
T20 BGA256 C4					121.3	113.18	
T85 BGA484 C4					113.4	114.50	
T120 BGA484 C4					113.4	114.50	

(1) Using default parameter settings.
(2) Using Verilog HDL.

Functional Description

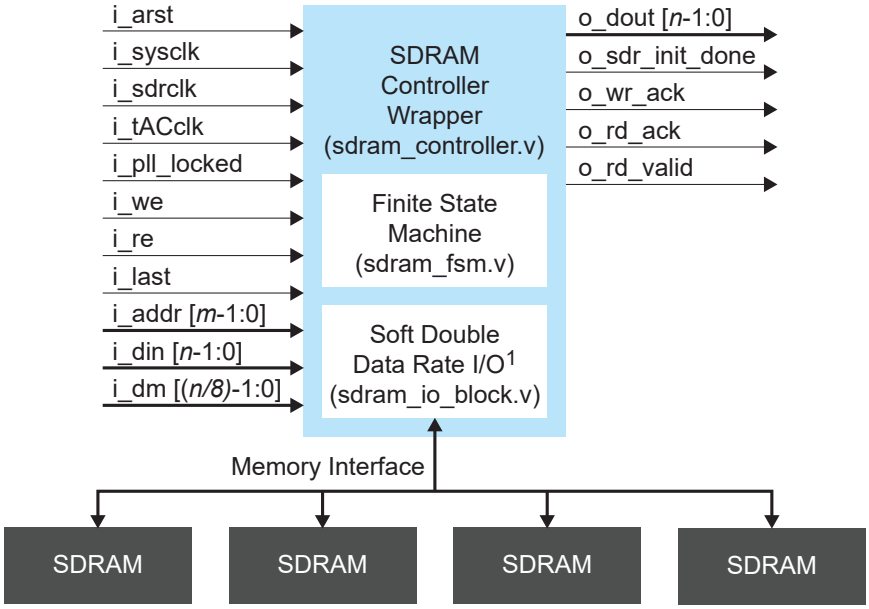
The SDRAM Controller consists of an AXI4 to native interface translator, an SDRAM controller wrapper, a finite state machine (FSM), and a soft double data-rate block.

The SDRAM Controller core supports the following user interfaces:

- Native interface
- AXI4 interface

Tip: For better performance and shorter development time, Efinix recommends that you instantiate the SDRAM Controller core with the native interface.

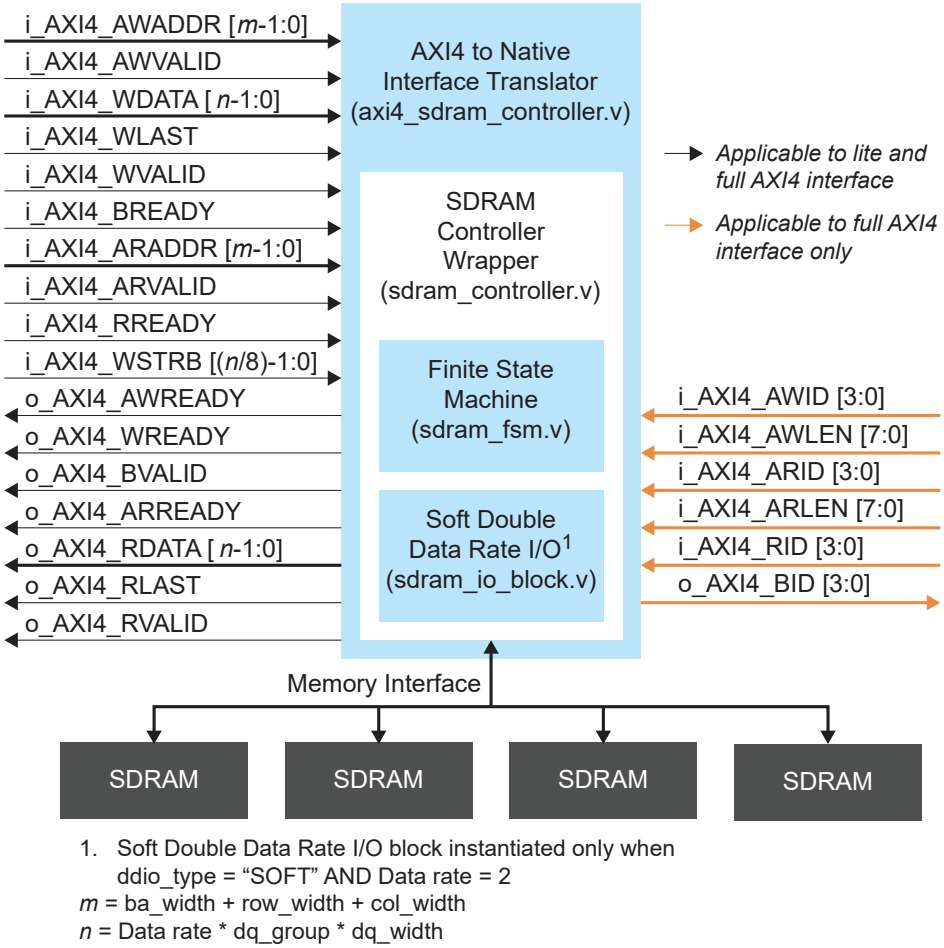
Figure 1: SDRAM Controller Native Interface System Block Diagram



1. Soft Double Data Rate I/O block instantiated only when ddio_type = “SOFT” AND Data rate = 2

m = ba_width + row_width + col_width
n = Data rate * dq_group * dq_width

Figure 2: SDRAM Controller AXI4 Interface System Block Diagram



1. Soft Double Data Rate I/O block instantiated only when ddio_type = “SOFT” AND Data rate = 2

m = ba_width + row_width + col_width

n = Data rate * dq_group * dq_width

Data Rate

The SDRAM Controller data rate is calculated using the following formula:

Data rate = fck_mhz / fsys_mhz

Ports

Table 1: SDRAM Controller Core Native Interface Ports

m = ba_width + row_width + col_width

n = Data rate * dq_group * dq_width

Port	Direction	Description
i_arst	Input	Positive controller reset.
i_sysclk	Input	Controller clock.
i_sdrclk	Input	Maps the t_{SU} and t_{WD} clocks to the following frequency: • i_sysclk–for full-rate (Data rate=1) • i_sysclk x2–for half-rate (Data rate=2)
i_tACclk	Input	Maps the t_{AC} clocks to the following frequency: • i_sysclk–for full-rate (Data rate=1) • i_sysclk x2–for half-rate (Data rate=2) 200 MHz is sufficiently slow to share both i_sdrclk and i_tACclk with the same clock without getting errors in readback. Use the phase shift to adjust the access time if an error occurs.
i_pll_locked	Input	Assert when the PLL of i_sdrclk is locked. The clock is stable, and the SDRAM controller starts the SDRAM initialization sequence.
i_we	Input	Write enable. Write enable can be kept high throughout the required burst length to perform a burst write. Write enable can only be de-asserted if i_last is asserted and o_wr_ack is sampled as high.
i_re	Input	Read enable. Read enable can be kept high throughout the required burst length to perform a burst read. Read enable can only be de-asserted if i_last is asserted and o_rd_ack is sampled as high.
i_last	Input	Set to high to indicate the last transfer to terminate a burst write or read.
i_addr [*m*-1:0]	Input	SDRAM physical address mapped in the following physical address: **MSB** … **LSB** Bank | Row | Column • Full-rate–all the column bits are valid. • Half-rate–the column LSB is ignored. The address must be in the increment of two.
i_din [*n*-1:0]	Input	Data to be written to SDRAM.
i_dm [*n*/8-1:0]	Input	Data mask per data byte.
o_dout [*n*-1:0]	Output	Data read from SDRAM. o_dout shares the same data address mapping as i_din.
o_sdr_init_done	Output	Indicates that the SDRAM initialization sequence is done and the SDRAM is ready.
o_wr_ack	Output	Write acknowledge. Handshaking with write enable.
o_rd_ack	Output	Read acknowledge. Handshaking with read enable.
o_rd_valid	Output	Read valid. The data on o_dout is valid to be read.

Table 2: SDRAM Controller Core AXI4 Interface Ports

m = ba_width + row_width + col_width

n = Data rate * dq_group * dq_width

Port	Direction	Description
o_AXI4_AWREADY	Output	The controller is ready to accept an address and associated control signals.
i_AXI4_AWADDR [*m*-1:0]	Input	Address of the first transfer in a write burst transaction mapped in the following physical address: **MSB** … **LSB** Bank | Row | Column Full-rate: All column bits are valid. Half-rate: The column LSB is ignored. The address must be in the increment of two.
i_AXI4_AWVALID	Input	Master signaling valid write address and control information.
o_AXI4_WREADY	Output	The controller can accept the write data.
i_AXI4_WDATA [*n*-1:0]	Input	Write data.
i_AXI4_WLAST	Input	The last transfer in a write burst.
i_AXI4_WVALID	Input	Valid write data and strobes are available.
o_AXI4_BVALID	Output	Controller signaling a valid write response.
i_AXI4_BREADY	Input	Master can accept a write response
o_AXI4_ARREADY	Output	Controller ready to accept an address and associated control signals.
i_AXI4_ARADDR [*m*-1:0]	Input	The address of the first transfer in a read burst transaction mapped in the following physical address: **MSB** … **LSB** Bank | Row | Column Full-rate: All column bits are valid. Half-rate: The column LSB is ignored. The address must be in the increment of two.
i_AXI4_ARVALID	Input	Master signaling valid read address and control information.
i_AXI4_RREADY	Input	Master can accept the read data.
o_AXI4_RDATA [*n*-1:0]	Output	Read data.
o_AXI4_RLAST	Output	The last transfer in read burst.
o_AXI4_RVALID	Output	Valid read data and strobes are available.
i_AXI4_WSTRB [(*n*/8)-1:0]	Input	Write strobe signal per data byte.
i_AXI4_AWID [3:0][(3)]	Input	Write address ID.
i_AXI4_AWLEN [7:0][(3)]	Input	Write burst length. The AXI write burst cannot crosses memory column address boundary. For example, if the column width = 8, 8'hFF, the last write burst need to happen at column address 8'hFF.

(3) Applicable to AXI4 full mode only.

Port	Direction	Description
o_AXI4_BID [3:0](3)	Output	Respond ID.
i_AXI4_ARID [3:0](3)	Input	Read address ID.
i_AXI4_ARLEN [7:0](3)	Input	Read burst length.
i_AXI4_RID [3:0](3)	Input	Read ID.

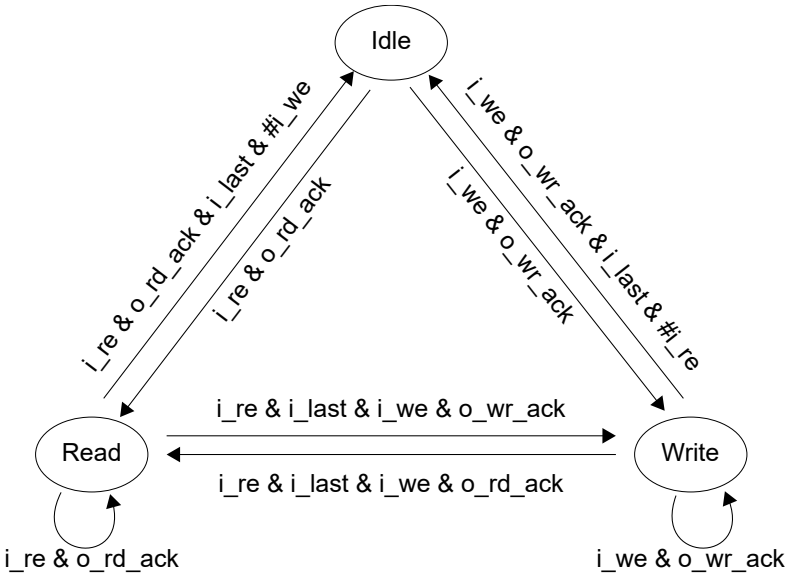
Native Interface

Native Interface Handshaking

The `i_we`, `i_re`, `o_wr_ack` and `o_rd_ack` ports provide the handshaking mechanism.

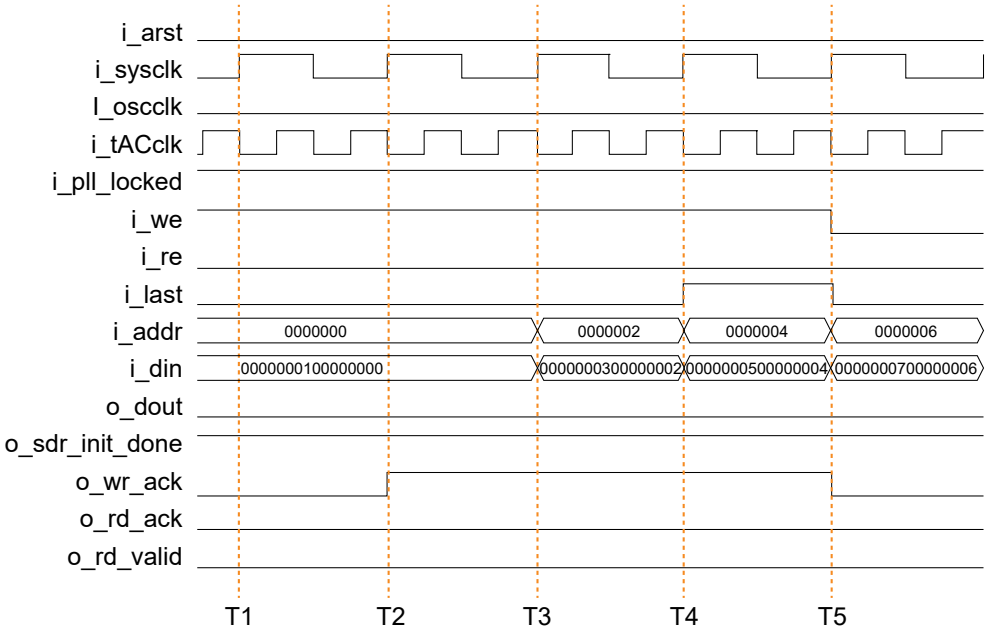
- The SDRAM controller keeps outputting data with `o_valid` asserted without any triggers.
- It is assumed that the master module always has buffers for handling both read and write handshaking.

Figure 3: SDRAM Controller Native Interface Handshaking



Native Write Timing

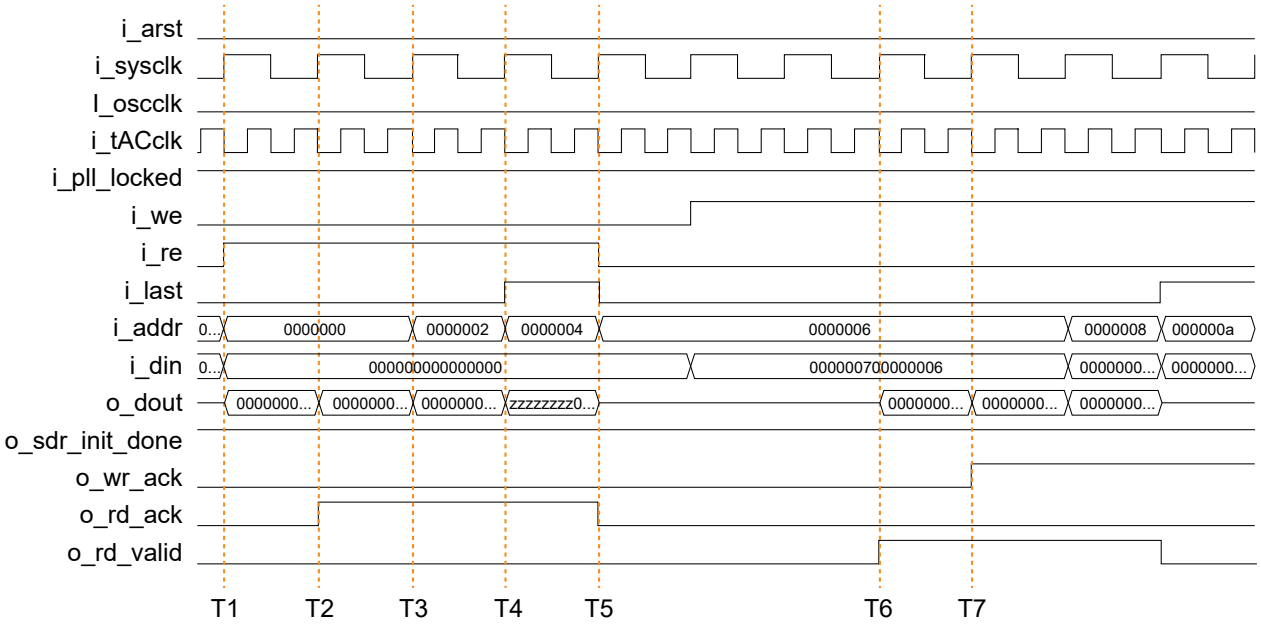
Figure 4: Native Write Timing Waveform



1. Master asserts write enable signal with both valid address and data. Master waits for the write acknowledgement from the SDRAM controller.
2. SDRAM controller acknowledges to the write enable signal.
3. Master updates the next address and data. Write enable signal remains high for a burst write.
4. Master terminates the burst write by asserting last signal.
5. Master de-asserts write enable and last signals.

Native Read Timing

Figure 5: Native Read Timing Waveform



1. Master asserts read enable signal with valid address and wait for the read acknowledgement from the SDRAM controller.
2. SDRAM controller acknowledges to the read enable
3. Master updates the next address. Read enable signal remains high for a burst read.
4. Master terminates the burst read by asserting the last signal.
5. Master de-asserts read enable and last signals.
6. SDRAM controller asserts read valid signal to indicate that the data is ready for master to read.
7. SDRAM controller continue to assert read valid signal until all read are sent out.

AXI4 Interface

AXI4 Interface Handshaking

An SDRAM controller with the AXI4 interface allows you to integrate your SDRAM controller with a system on a chip (SoC) design easily. All the channel handshaking dependencies are aligned with ARM IHI 0022E.

Memory Interface

Table 3: SDRAM Controller Memory Interface Ports

Refer to **Memory Interface Connection Settings** on page 11 for more information about connecting the memory interface ports.

Port	Direction	Width	Description
o_sdr_CKE	Output	Data rate	SDRAM CKE pin.
o_sdr_n_CS	Output	Data rate	SDRAM #CS pin.
o_sdr_n_RAS	Output	Data rate	SDRAM #RAS pin.
o_sdr_n_CAS	Output	Data rate	SDRAM #CAS pin.
o_sdr_n_WE	Output	Data rate	SDRAM #WE pin.
o_sdr_BA	Output	Data rate * ba_width	SDRAM BA pins.
o_sdr_ADDR	Output	Data rate * row_width	SDRAM ADDR pins.
o_sdr_DATA	Output	Data rate * dq_group * dq_width	SDRAM DQ pins from FPGA to SDRAM.
o_sdr_DATA_oe	Output	Data rate * dq_group * dq_width	SDRAM DQ output enable for I/O buffer.
i_sdr_DATA	Input	Data rate * dq_group * dq_width	SDRAM DQ pins from SDRAM to FPGA.
o_sdr_DQM	Output	Data rate * dq_group	SDRAM DQM pins.

Memory Interface Connection Settings

If your design uses the HARD ddio_type, set each of the memory interface ports in **Interface Designer** as follows:

- Select **resync** for **Double Data I/O Option**
- Connect the LSB half of the signal to **Pin Name (HI)**
- Connect the MSB half of the signal to **Pin Name (LO)**

For design using SOFT ddio_type, the MSB half of the signal is invalid.

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinix® 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 Efinix 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 Efinix IP cores include an example design or a testbench.

Generating a 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 an IP 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 IP core's user guide or on-line help.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinix® development board and/or testbench. For SoCs, you can also optionally generate embedded software example code. 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.

Note: Refer to the IP Manager chapter of the Efinity® Software User Guide for more information about the Efinity® IP Manager.

Customizing the SDRAM 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 4: SDRAM Controller Core Parameters

Parameters	Options	Description
Memory Clock Frequency	133, 166, 200	Define the SDRAM clock frequency, i_sdrclk in MHz. Default = 200
I/O Round Trip Time	>= 0	Define the I/O round trip time in controller clock cycles. Calibrate this parameter manually once per system. Refer to **Calibrate SDRAM Controller** on page 18 for steps to perform manual calibration. Default = 2
CAS Latency	2, 3	Define the CAS latency in SDRAM clock cycles. Default = 3
DDIO Register Implementation	SOFT, HARD	Select the double data rate I/O register implementation between: • Soft DDIO–implement with flip-flops • Hard DDIO–implement with I/O register Default = SOFT
Data Bus Grouping	1, 2, 4	Define the data bus grouping per device. Example: • A x32 width from four x8 devices dq_width = 8 dq_group = 4 • A x16 width from two x8 devices dq_width = 8 dq_group = 2 Default = 4
Row Addressing Width	12, 13	Define the row addressing width. Default = 13
Column Addressing Width	8, 9, 10	Define the column addressing width. Default = 10
tPWRUP (ns)	100000, 200000	Define the power-up sequence in ns. Default = 200000
Minimum tRAS (ns)	37, 40, 42, 44	Define the minimum time for ACTIVE-to-PRECHARGE command in ns. This value must be higher than the minimum t_{RAS} value stated in your SDRAM specification sheet. Default = 44

Parameters	Options	Description
Maximum tRAS (ns)	100000, 120000	Define the maximum time for ACTIVE-to-PRECHARGE command in ns. This value must be lower than the maximum t_{RAS} value stated in your SDRAM specification sheet. Default = 120000
tRC (ns)	55, 60, 63, 66	Define the ACTIVE-to-ACTIVE command period in ns. This value must be higher than the minimum t_{RC} value stated in your SDRAM specification sheet. Default = 66
tRCD (ns)	15, 18, 20	Define the ACTIVE-to-READ or WRITE delay in ns. This value must be higher than the minimum t_{RCD} value stated in your SDRAM specification sheet. Default = 20
tREF (ns)	25000000 - 64000000	Define the refresh period in ns. This value must be lower than the maximum t_{REF} value stated in your SDRAM specification sheet. Default = 64000000
tRFC (ns)	55, 66	Define the AUTO REFRESH period in ns. This value must be higher than the minimum t_{RFC} value stated in your SDRAM specification sheet. Default = 66
tRP (ns)	15, 18, 20	Define the PRECHARGE command period in ns. This value must be higher than the minimum t_{RP} value stated in your SDRAM specification sheet. Default = 20
Controller Interface	AXI4, Native	Defines the user interface. Default = AXI4
Data Rate	Half rate, Full rate	Defines the fsys_mhz. Data rate = fck_mhz / fsys_mhz. Default = Half rate

The SDRAM Controller core has pre-defined parameters that are not included in the IP Configuration window. These parameters cannot be changed and listed below as reference.

Table 5: SDRAM Controller Pre-Defined Parameters

Parameters	Options	Description
Burst length	1	Burst length.
Memory Data Width	8	Data input/output bus width per device. Used with the dq_group parameter.
Bank Addressing Width	2	Bank addressing width.
tWR	2	Define the WRITE recovery time in ns. This value must be higher than the minimum t_{WR} value stated in your SDRAM specification sheet.
tMRD	2	Define the LOAD MODE REGISTER command to ACTIVE or REFRESH command in SDRAM clock cycles. This value must comply with the t_{WR} value stated in your SDRAM specification sheet.

SDRAM 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.

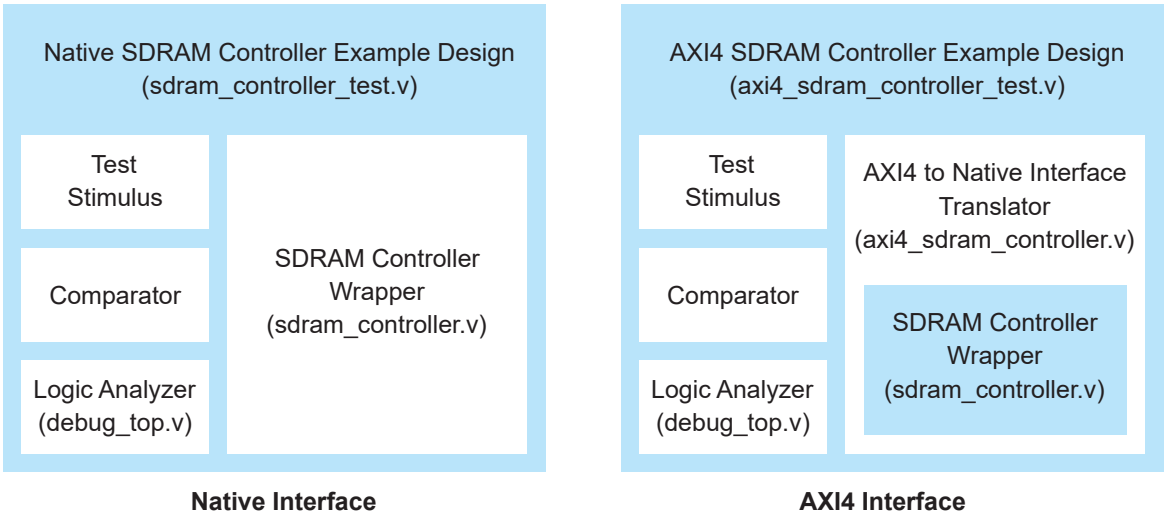
Important: Efinix tested the example design generated with the default parameter options only.

The example design targets the Trion® T20 BGA256 Development Kit. This design continuously writes to the memory with counting up data pattern. After 255 bursts of write data are written to the memory through the AXI4 interface, the design reads back all the data with a burst read operation. The comparator in the design compares the read back data with the expected write data. You can observe the following board LED behaviours:

- LED D4 and D5 blink when there is an error.
- LED D3 blinks and LED D8, D9, D10 light-up when there is no error.

To reset, press push button SW4.

Figure 6: Example Design Block Diagram



- *Test Stimulus*—Perform write burst and read burst operations.
- *Comparator*—Compares the written and read back data from the memory.
- *Logic Analyzer*—Debug core used to perform manual calibration shown in Manual Calibration on page 17.

Table 6: Example Design Implementation

FPGA	Interface	LUTs	Memory Blocks	f_{MAX} (MHz)[4]	Efinity® Version
T20 BGA256 C4	Native	5,578	64	118	2019.3
	AXI4	5,700	65	123	

(4) Using default parameter settings.
(5) Using Verilog HDL.

SDRAM 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.

Efinix 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 in your computer to use this script.

The testbench generates a clock signal, reset signal and memory model instantiation (**generic_sdr.v**). By default, the testbench runs with the AXI4 interface example design. If you want to run the native interface example design, change the module instantiation inside the testbench file.

The system displays the following error message if an error occurred:

```
ERROR: [8] READ DATA AAh is not the same as EXPECTED DATA BBh
```

Note: If you want to use your own testbench file, add the following line in your testbench file:

```
`define RTL_SIM
```

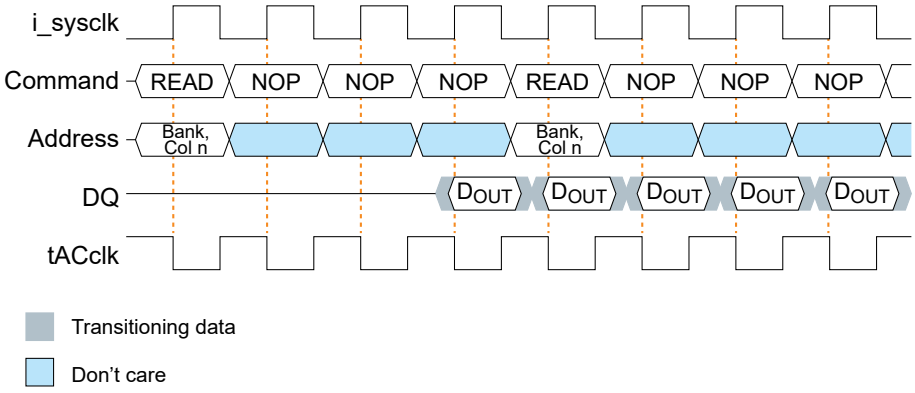
Manual Calibration

Due to the t_{CO} of the SDRAM and the printed circuit board (PCB) trace delay, you must perform a manual calibration once per system for the SDRAM controller to work properly.

t_{ACclk} Phase Shift Alignment

Align the t_{ACclk} phase shift to the center of access window of read data sample time.

Figure 7: t_{ACclk} Read Memory Operation

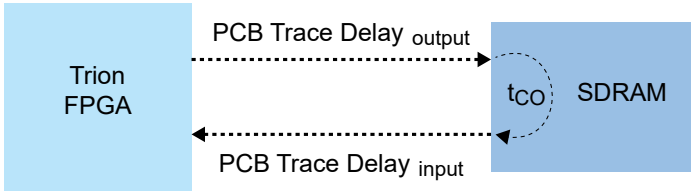


t_{IORT} Parameter Alignment

Adjust the t_{IORT} to align the t_{CO} of SDRAM and PCB trace delay. The t_{IORT} value is determined by the following equation:

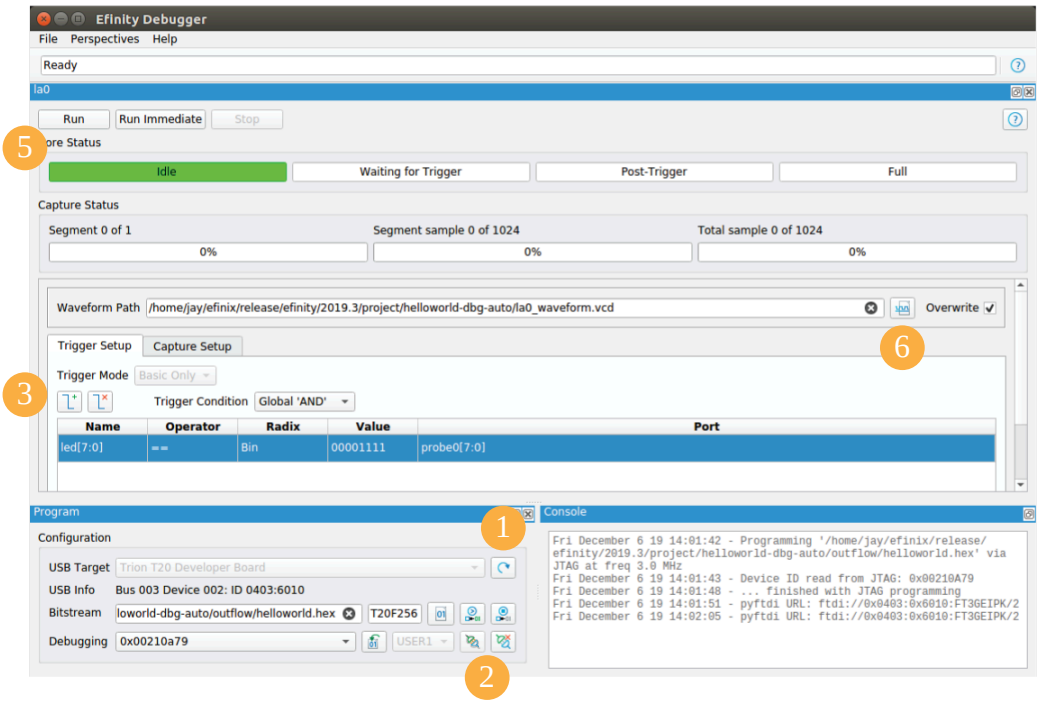
t_{IORT} = (PCB trace delay $_{output}$ + t_{CO} + PCB trace delay $_{input}$) / i_sysclk period

Figure 8: t_{IORT} Dependence



Calibrate SDRAM Controller

Figure 9: Debug Perspective GUI - Logic Analyzer



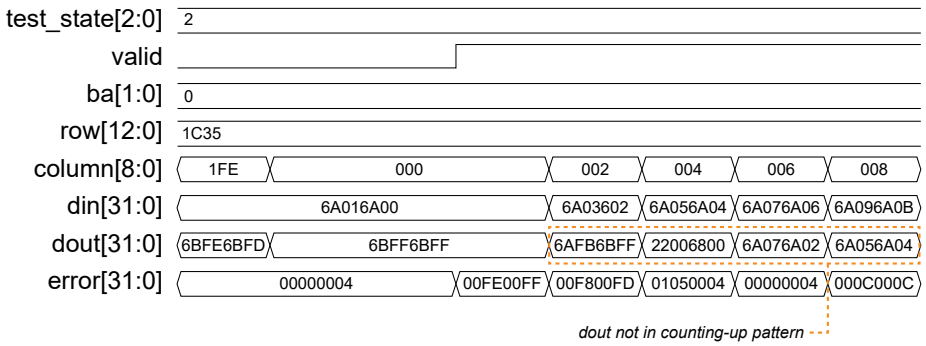
Select Bitstream	Connect Debugger	Disconnect Debugger
Start Programming	Add Net	Remove Net
Stop Programming	Select Waveform File	

Perform the following steps in the **Efinity® Debugger** to calibrate the SDRAM manually:

1. Select the bitstream and program your Trion® FPGA.
2. Click Connect Debugger.
3. In the **Trigger Setup**, click Add Net.
4. Set **Valid** to rising edge trigger.
5. Click **Run** and the waveform is stored in a **.vcd** file.
6. Click Select Waveform File to open the waveform file.

> **Note:** You can view the waveform with GTKWave software. Download and install the GTKWave software from gtkwave.sourceforge.net. *Windows:* You may need to add the path to GTKWave (`$GTKWave_folder$\bin\`) to your System Variables path for the software to launch correctly.

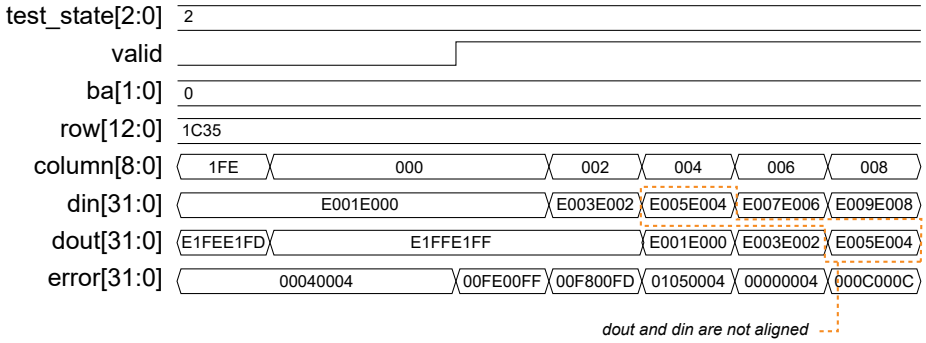
7. Verify if the `dout` signal is in a counting-up pattern.



8. You must adjust the t_{ACclk} phase shift so the `dout` signal is in a counting-up pattern. Skip this step if the `dout` signal is in a counting-up pattern. Try the available **Phase Shift (Degree)** values in the **Interface Designer** until `dout` signal is in a counting-up pattern.

   You can use the Shmoo plot technique to find optimal t_{ACclk} signal phase shift value.

9. Verify that the `din` and `dout` signals are aligned.



10. You must adjust the t_{IORT} parameter to align the `din` and `dout` signals. Skip this step if the `din` and `dout` signals are aligned. Change the t_{IORT} parameter value between 0 to 7.

    You can use the Shmoo plot technique to find optimal t_{IORT} parameter value.

Figure 10: Correct Calibration Waveform

Revision History

Table 7: Revision History

Date	Version	Description
February 2023	2.4	Added note about the resource and performance values in the resource and utilization table are for guidance only.
August 2022	2.3	Added `i_AXI4_WSTRB` and `i_dm` ports. Corrected `i_addr`, `i_din`, and `o_dout` port widths.
October 2021	2.2	Added note to state that the f_{MAX} in Resource Utilization and Performance, and Example Design Implementation tables were based on default parameter settings.
June 2021	2.1	Added note about including all **.v** generated in testbench folder is required for simulation.
December 2020	2.0	Updated user guide for Efinix® IP Manager which includes added IP Manager topics, updated parameters, and user guide structure.
October 2020	1.4	Removed dummy signals from AXI4 ports. (DOC-320) Updated i_AXI4_AWADDR, i_AXI4_ARADDR and i_AXI4_AWLEN ports description. (DOC-320)
July 2020	1.3	Updated example design description. Added note to fsys_mhz and fck_mhz parameter to state that other values can be used but not tested by Efinix®.
May 2020	1.2	Removed T4 BGA81 C2 FPGA from resource utilization and performance table. Updated description for fsys_mhz and fck_mhz parameters. Updated value range for fsys_mhz, fck_mhz, tRC, tRCD, tREF, tRFC, and tRP.
April 2020	1.1	Removed 166 MHz controller clock support from features list. Added 100 MHz to fck_mhz parameter range. Removed 133 MHz and 166 MHz from fsys_mhz parameter range.
April 2020	1.0	Initial release.