DDR Hard Memory Controller-Calibration Core User Guide

UG-DDRAUTOCAL-v3.4
June 2022
www.efinixinc.com

Contents

Introduction

The DDR hard memory controller-calibration core helps you optimize timing and calibrate the Trion DDR controller using write leveling, read leveling, and gate training. The core supports an automated calibration mode as well as calibration on demand, depending on your configuration.

Use the IP Manager to select IP, customize it, and generate files. The DDR hard memory controller-calibration 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

- Automatically performs leveling calibration for the DDR DRAM interface and external memory module
- Supports write leveling, read leveling, and gate training calibration
- Supports RL8/RW4 of DDR memory
- Verilog RTL and simulation testbench
- Includes an example design targeting the Trion® T120 BGA324 Development Board

FPGA Support

The DDR hard memory controller-calibration core supports Trion T20, T35, T55, T85, and T120 FPGAs.

Note: Refer to the **Trion DDR DRAM Block User Guide** for the list of supported DDR DRAM modules and the supported DDR PHY frequencies.

Resource Utilization and Performance

Trion Resource Utilization and Performance

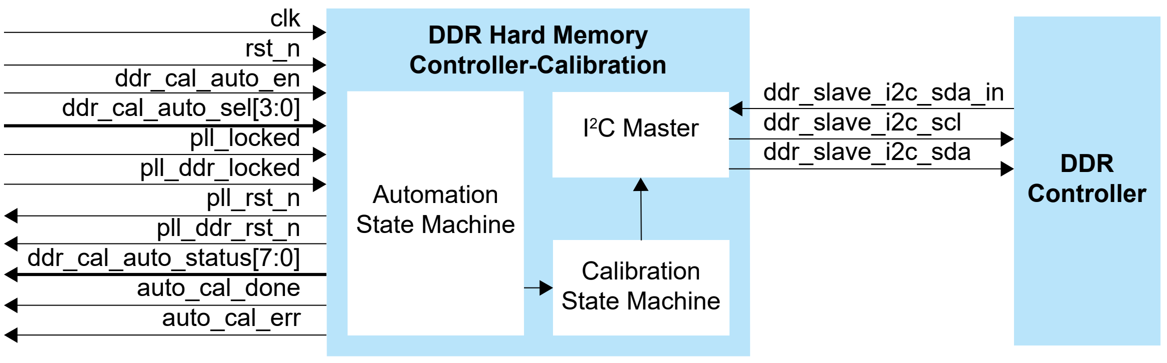
FPGA	Logic Utilizations (LUTs)	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)[1]	Efinity® Version[2]
T20 BGA256 C4	1,210	515	12	0	141	2021.1
T120 BGA324 C4	1,210	515	12	0	139	2021.1

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

Functional Description

The core consists of an automation state machine, a calibration state machine, and an I^{2}C master.

Figure 1: DDR Hard Memory Controller-Calibration Core Block Diagram



Ports

Table 1: DDR Hard Memory Controller-Calibration Core Ports

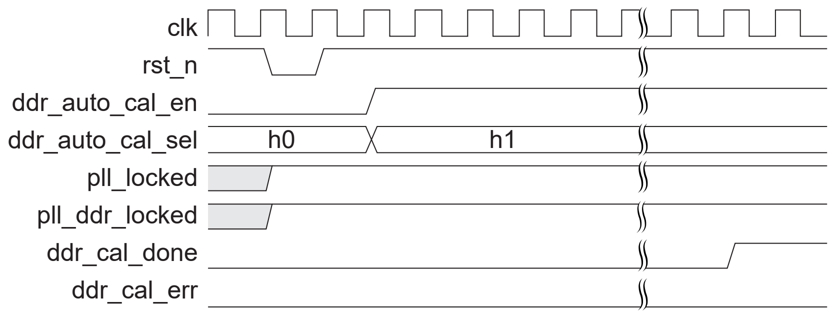
Port	Direction	Description
rst_n	Input	Reset. Asynchronous, active-low reset signal that initializes all internal pointers and output registers.
clk	Input	System clock. This clock drives all signals in the core. Efinix recommends using a clock in the range of 2 to 8 MHz, which generates an I^{2}C clock of 100 to 400 KHz.
ddr_cal_auto_en	Input	Enable signal to start the automation state machine.
ddr_cal_auto_sel[3:0]	Input	Select signal to configure the calibration mode you want to run. Bit 0: Read leveling Bit 1: Gate training Bit 2: Write leveling Bit 3: Reserved Disable the write leveling (Bit 2 = 0) if you do not use fly-by topology in your board design.
pll_locked	Input	PLL lock indicator from the generated system clock. Tie this signal to 1 (high) if the system clock is not from a PLL.
pll_ddr_locked	Input	PLL lock indicator from the DDR generated clock.
pll_rst_n	Output	PLL active-low reset to the generated system clock. Leave this port unconnected if the system clock is not from a PLL.
pll_ddr_rst_n	Output	PLL active-low reset to the generated DDR clock.

Port	Direction	Description
ddr_cal_auto_status[7:0]	Output	Optional. Status signal for debugging. Bit 0: Indicates read leveling calibration is running. Bit 1: Indicate gate training calibration is running. Bit 2: Indicate write leveling calibration is running. Bit 3: Reserved. Bit 4: Base sequence active signal from the calibration state machine. Bit 5: Main loop active signal from the calibration state machine. Bit 6: Sub-loop 1 active signal from the calibration state machine. Bit 7: Sub-loop 2 active signal from the calibration state machine.
auto_cal_done	Output	Optional. Indicates that the automation state machine successfully completed calibration.
auto_cal_err	Output	Optional. Indicates that the automation state machine started with an invalid ddr_cal_auto_sel setting.
ddr_slave_i2c_scl	Output	I^{2}C clock to the DDR controller.
ddr_slave_i2c_sda	Output	I^{2}C data to the DDR controller.
ddr_slave_i2c_sda_in	Input	I^{2}C enable to the DDR controller.

Automation State Machine

This state machine controls the calibration process. When you assert `ddr_cal_auto_en`, the state machine starts. The core uses the value in `ddr_cal_auto_sel` to choose which calibration mode to run. For example, if `ddr_cal_auto_sel` is set to `4'b0101`, the state machine performs write leveling calibration and then read leveling calibration. The core gives highest priority to write leveling, followed by gate leveling, and lastly read leveling.

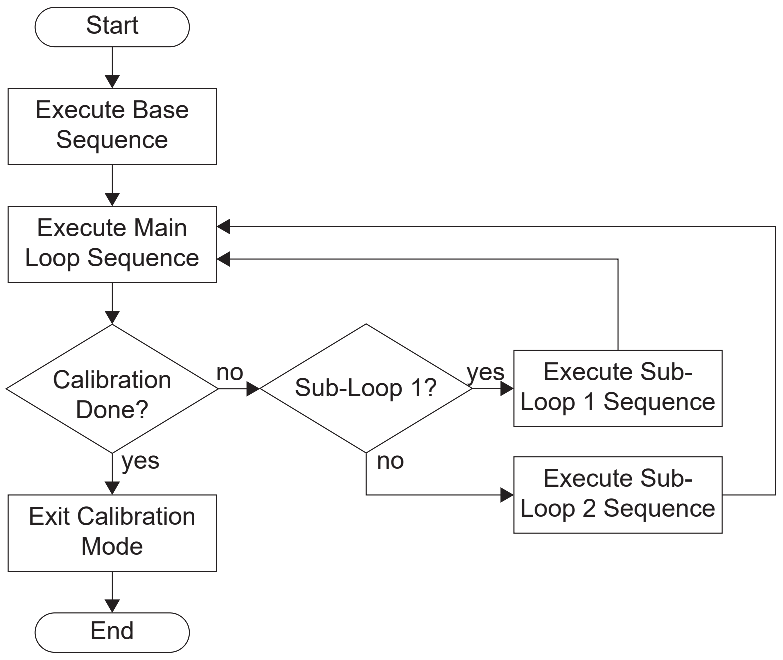
Figure 2: DDR Hard Memory Controller-Calibration Operation



Calibration State Machine

Each calibration mode executes different a programming sequence or setting. The calibration state machine ensures that the core retrieves the correct instructions from the ROM and send them to the I^{2}C master according to calibration mode you choose. Instructions continue executing in a loop until the DDR controller indicates that calibration is complete. Therefore, the calibration state machine consists of one base sequence state and 3 loop states, providing flexibility to change the programming execution according to the needs of different calibration modes.

Figure 3: Simplified Calibration State Machine Flow



I^{2}C Master

A simple I^{2}C master provides functions including read, write, polling, and masked write with returned data. The I^{2}C slave address is set to `8'h82`, and only communicates with the DDR controller. The I^{2}C master receives instructions from the calibration state machine and performs I^{2}C functions by extracting the most significant 4 bits of the total 108 instruction bits.

Table 2: I^{2}C Instruction Description

107:102	101:95	94:64	63:32	31:0
Function	Number of instructions	APB mask	APB data	APB address

Table 3: I^{2}C Functions

Bit	107	106	105	104	103	102
Function	Write	Read	Poll	Read masked write	Reserved	Save returned data

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 DDR Hard Memory Controller-Calibration

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: DDR Hard Memory Controller-Calibration Core Parameters

Parameter	Options	Description
DDR Memory Type	LPDDR3, DDR3/ DDR3L, LPDDR2	Defines DDR memory module type. Default: LPDDR3
DDR Memory Width	32-bits, 16-bits	Defines DDR memory width. Default: 32-bits

DDR Hard Memory Controller-Calibration 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® T120 BGA324 Development Board. It implements a memory checker that writes data into the DDR DRAM module on the board and then performs a read operation. The design compares the returned data to the sent data and displays the results using the board's LEDs. You can turn calibration on or off to see how the DDR DRAM module works with and without calibration. The design uses an 8 MHz clock (`clk`).

Figure 4: DDR Hard Memory Controller-Calibration Core Example Design

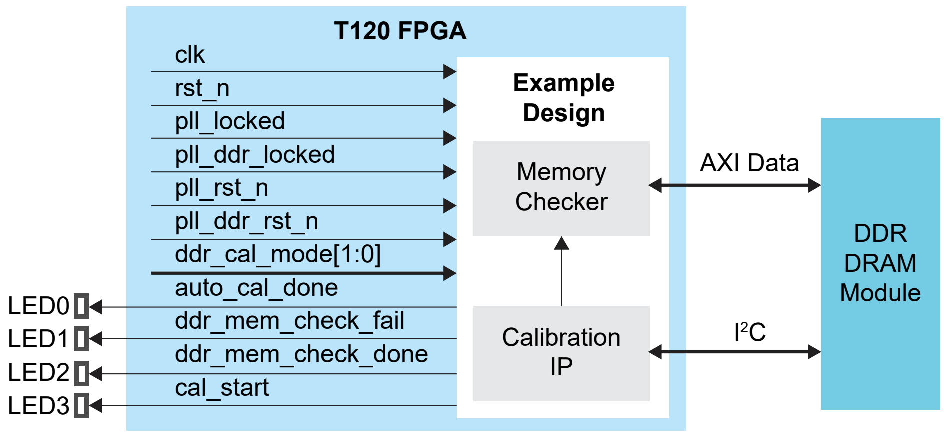


Table 5: Example Design Implementation

FPGA	Logic Utilizations (LUTs)	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)[3]	Efinity® Version[4]
T120 BGA324 C4	1,540	759	12	0	130	2021.1

Control the Calibration Mode with DIP Switches

The design lets you switch between write levelling, gate training, and read leveling calibration operations using DIP switches on SW5.

Table 6: DIP Switch Settings

Calibration Operation	SW5 DIP1	SW5 DIP2
All calibration operations	0	0
Gate training	0	1
Read leveling	1	0
Write leveling	1	1

Bypass Calibration Mode

You can bypass calibration and only run the memory checker. This mode is helpful to cross-check the result with and without calibration. To bypass calibration, set the `CALIBRATION_BYPASS` parameter to `1` in the **memory_checker_top.v** file.

```
module ddr_mem_checker_top #(
    parameter       DDR_X16                 = 1,
    parameter       DDR_TYPE                = 2'b00,
    parameter       ROM_DATA_DEPTH0         = 41,
    parameter       ROM_DATA_DEPTH1         = 29,
    parameter       ROM_DATA_DEPTH2         = 2,
    parameter       ROM_DATA_DEPTH3         = 2,
    parameter       ROM_INIT_FILE0          = "rom0_lpddr3_x16.hex",
    parameter       ROM_INIT_FILE1          = "rom1_lpddr3_x16.hex",
    parameter       ROM_INIT_FILE2          = "rom2_lpddr3_x16.hex",
    parameter       ROM_INIT_FILE3          = "rom3_lpddr3_x16.hex",
    parameter       CALIBRATION_BYPASS      = 1,    // Bypass calibration
    parameter       EFX_SIM                 = 0
)
```

[3] Using default parameter settings.
[4] Using Verilog HDL.

Using the Design

As the example design runs, the user LEDs on the Trion® T120 BGA324 Development Board show the calibration results.

- Three seconds after you reset the board, LED3 turns on to indicate the design is operating. If bypass mode is turned off, calibration begins. If bypass mode is turned on, the design skips calibration.
- When calibration completes (or in bypass mode), LED0 turns on to indicate the memory checker is operating.
- If the memory checker successfully writes and then reads the memory, it turns on LED2.
- LED1 turns on when the the write and read data do not match.

First, run the design with bypass mode turned off (default). LED3 turns on, then LED0, and then LED2.

Next, set the `CALIBRATION_BYPASS` parameter to 1, recompile, and download the new bitstream to the board. When you run the design, LED3 turns on, then LED0. The design sees the write and read data mismatch, so LED0 turns off and LED1 turns on.

Using the Core

To use the DDR hard memory controller-calibration core, instantiate it in your design. Decide the calibration operations you want to perform, and then program the corresponding bits in the `ddr_cal_auto_sel[3:0]` register. For example:

Operations	Register Setting
Write leveling only[5]	4'b0100
Write leveling, then gate training, and then read leveling	4'b0111
Write leveling, then read leveling	4'b0101

[5] Supported in LPDDR3, DDR3 and DDR3L only.

Revision History

Table 7: Revision History

Date	Version	Description
June 2022	3.4	Added description about disabling write leveling if not using fly-by topology. (DOC-838)
November 2021	3.3	Added description about the core only supports RL8/RW4 of DDR memory.
October 2021	3.2	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	3.1	Added note about including all **.v** generated in testbench folder is required for simulation.
December 2020	3.0	Updated core name to DDR Hard Memory Controller-Calibration. Updated user guide for Efinix® IP Manager which includes added IP Manager topics, updated parameters, and user guide structure.
Spetember 2020	2.0	Updated for core version 2.0 Removed supported DDR DRAM modules table and replaced it with a link to Trion® DDR DRAM Block User Guide. Updated LUTs and fMAX in the resource utilization and performance table. Updated LUTs and fMAX in the example design implementation table. Updated ROM_DATA_DEPTH0, ROM_DATA_DEPTH1, ROM_DATA_DEPTH2, and ROM_DATA_DEPTH3 parameters.
August 2020	1.3	Removed Simulated the Testbench topic. Added note to write leveling only operation to support LPDDR3 and DDR3 only.
May 2020	1.2	Updated ROM_DATA_DEPTH2 parameter description.
April 2020	1.1	Added recommendations for the clk signal speed.
April 2020	1.0	Initial release.