SYNOPSYS®

DesignWare® Cores DDR/LPDDR SINIT/CINIT

User Guide

Version: 4.10a
March 2024

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.

www.synopsys.com

Contents

Revision History

Version	Date	Description
4.10a	March 2024	**Added:** ■ “DDR Subsystem Rank-to-Rank Space Timing” on page 37 **Updated:** ■ Table 2-1 on page 9 ■ “Set DDR Controller Core and PHY Definitions” on page 12 ■ “Configuration” on page 16 ■ Table 3-4 on page 18 ■ “Create Standalone Application Using the Provided Defconfigs” on page 19 ■ “Create Standalone Application From Scratch (Advanced Approach)” on page 20 ■ “Example Defconfig Configurations” on page 24
4.00a	December 2022	Updated version only (no functional changes)
3.80a	September 2022	**Added:** New BSP custom functions (io_switch_clk, smbus_write) **Updated:** Hardware configuration folder kconfig options menu Standalone compilation instructions
3.70a	April 2022	Updated version only (no functional changes)
3.60a	April 2022	**Added:** Validated configurations **Updated:** GCC supported version Compilation flow Improved document organization Corrected DDR4 SPD configuration

Version	Date	Description
3.50a	January 2022	**Updated:** Known issues
3.40b	December 2021	**Added:** Dynamic compilation option BSP required adaptations CINIT API SINIT flow **Updated:** Python version
3.40a	November 2021	**Updated:** DDR4/DDR5 compilation options to reduce code size API for SDRAM bus width configuration Improved code structure Reduced data structures size
3.30a	August 2021	**Added:** Namespace to macros kconfig python library as a requirement **Updated:** Code structure Corrected code warnings for MIPS toolchain Dropped dependency on bison, yacc, lex, and kconfig tool
3.20a	May 2021	Updated version only (no functional changes)
3.00a	May 2021	**Added:** New compilation flow
2.00a	January 2021	Initial Draft

1 Overview

This chapter provides a high-level description of the DesignWare® DDR/LPDDR SINIT/CINIT. It contains the following topics:

1.1 About the DesignWare DDR/LPDDR SINIT/CINIT

The DesignWare DDR/LPDDR SINIT/CINIT, is a software that generates an initialization sequence for the DDR/LPDDR controller based on the selected configuration.

1.2 Reference Documents

For more information about the DesignWare DDR/ LPDDR SINIT/CINIT, refer to the following documentation:

- DesignWare Cores DDR5/4 Memory Controller Databook
- DesignWare Cores DDR5/4 Memory Controller Reference Manual
- DesignWare Cores LPDDR5/4/4X Memory Controller Databook
- DesignWare Cores LPDDR5/4/4X Memory Controller Reference Manual

2 Software Package Contents

This chapter describes the DesignWare® DDR/LPDDR SINIT/CINIT folder structure.

Table 2-1 Folder Structure of the SINIT/CINIT

Folder	Description
Makefile	Top-level Makefile
apps/standalone	Standalone applications
bsp	Board support package code
bsp/platform/example	Prototypes for the platform specific functions that need to be adapted to the target platform
bsp/platform/local_x86	Implementation of the platform specific functions for a debug x86 system
bsp/platform/simulation	Implementation of the platform specific functions for the testbench CPU_DPI simulation
bsp/platform/uvm_tb	Implementation of the platform specific functions for the testbench UVM simulation
hw_config/include	CC constants and phy VDefines files
library/configs	Default configurations for each build solution
modules	Modules source code
modules/core	Main source code of the cinit
modules/physetup	Connection to the phy init code
modules/sequences	Implementation of the Memory sub system sequences
modules/verification	Code specific for the verification environment
tools/spd_generator	Tool to create an SPD from a configuration file
tools/c_sv_converter	Tool to convert between C and System Verilog header files
tools/log_util	Tool to process cinit log files
tools/sconfig	SConfig SINIT/CINIT configuration tool
tools/trace_decoder	Tool to process log trace files

3 CINIT

SINIT/CINIT is a software library that simplifies the DDR/LPDDR controller configuration.

The DDR and LPDDR controllers are highly configurable and the process of configuring them correctly is a complex activity. This process is abstracted on the levels of library and its configuration tool.

The configuration tool simplifies passing the required settings and memory configuration.

The hardware settings restricts the available options.

SINIT (Subsystem Initialization) sequences folder is included and contains subsystem sample sequences that can be taken as a reference for Memory Subsystem initialization.

Note In the controller you need to program the microcode sequence (DDR5 only). This is not done by CINIT standalone as this is not done at time 0 and needs to be embedded. The code for generating the microcode is included and located at modules/core/src/dwc_ddrctl_cinit_prgm_ucode.c

3.1 Set DDR Controller Core and PHY Definitions

You need the Controller and PHY configurations to build the SINIT/CINIT code.

The simple way is to copy the code from the simulation paths, after extracting the release package and running the coreConsultant configuration batch.

Go to the lib/<product_code>/sim/test_dwc_ddrctl_cinit_cpu_dpi_<selected_config>_defconfig folder and use the sw_utilities as a starting working base.

If no sw_utilities folder is present, run the build_cinit.sh there.

Important files in the folder:

- cinit/hw_config/include/DWC_ddrctl_cc_constants.h
- cinit/hw_config/include/dwc_ddrphy_VDEFINES.h
- cinit/library/MemoryMapXml.csv
- cinit/library/configs/testbench_defconfig
- phy/software/protocol/*.o (precompiled phy files)

3.2 BSP Required Adaptations

To support several architectures/platforms/compilers/Oss/ the I/O access and system calls are abstracted in the bps folder.

It includes support for two verification methods and one to run as a standalone application:

CPU_DPI mode: Controller bring-up is completely managed by the SINIT/CINIT library.

UVM_TB mode: The library is only allowed to read the controller register and program the uCode in DDR5. At the end, it creates a sequence file with the registers that need to be programed.

STANDALONE mode: The controller registers are simulated and create a sequence file with the registers that need to be programed.

For other platforms, you must implement the BSP depending on the specific platform needs and characteristics.

3.2.1 Architecture

Create the file *bsp/arch/<your_arch>.mk* and use it to add the specific compilation flags for your CPU.

Use *x86.mk* and *x86_64.mk* as an example.

Pass the architecture on the make command using

```
CINIT_ARCH=<your_arch>
```

3.2.2 Compiler

Create the file *bsp/compiler/< your_compiler>.mk* and use it to define the applications that are responsible for the compilation of the code.

Use *gcc.mk* as an example.

Pass the compiler on the make command using

```
CINIT_COMPILER=<your_compiler>
```

3.2.3 Operative System

Create the folder structure *bsp/os/<your_os>/include* and create the *bsp/os/<your_os>/include/dwc_cinit_os_bsp.h*. Use it to add the standard libs specific to your OS.

Use *bsp/os/linux/include/dwc_cinit_os_bsp.h* as an example.

Pass the architecture on the make command using

```
CINIT_OS=<your_os>
```

3.2.4 Platform

Create the folder structure *bsp/platform/<your_platform>/* and use it to implement the platform specific required functions defined in “Error! Reference source not found.”.

Use *bsp/platform/local_x86 and bsp/platform/*simulation as an example.

Pass your platform on the make command using

```
CINIT_PLATFORM=<your_platform>
```

3.2.5 Hardware Abstraction Layer

The *bsp/include* folder contains predefined functions that must be adapted to the final platform.

The following table describes the functions.

Table 3-1 Platform Specific Functions

Pre-defined Functions	Description
void dwc_ddrctl_cinit_custom_io_write32 (uint32_t address, uint32_t data)	This API must be rewritten to the customer target platform. It allows writing to the memory controller register (address) with a certain value (data).
uint32_t dwc_ddrctl_cinit_custom_io_read32 (uint32_t address)	This API must be rewritten to the customer target platform. It allows reading from the memory controller register (address) and obtain a certain value.
void dwc_ddrctl_cinit_custom_io_power(bool enable)	This API must be rewritten to the customer target platform. It allows controlling the memory controller power signal.
void dwc_ddrctl_cinit_custom_io_presetn(bool enable)	This API must be rewritten to the customer target platform. It allows controlling the memory controller peripheral reset signal (inverted logic).
void dwc_ddrctl_cinit_custom_io_aresetn(bool enable)	This API must be rewritten to the customer target platform. It allows controlling the memory controller AXI reset signal (inverted logic).
void dwc_ddrctl_cinit_custom_io_ddrc_rstn(bool enable)	This API must be rewritten to the customer target platform. It allows controlling the memory controller reset signal (inverted logic).
void dwc_ddrctl_cinit_custom_io_wait_pclk(uint32_t cycles)	This API must be rewritten to the customer target platform. Waits for a certain memory controller peripheral cycle.
void dwc_ddrctl_cinit_custom_io_wait_ddrc_clk(uint32_t cycles)	This API must be rewritten to the customer target platform. Waits for a certain memory controller clock cycle.
void dwc_ddrctl_cinit_custom_io_set_pclk(bool enable)	This API must be rewritten to the customer target platform. It allows controlling the memory controller peripheral clock signal.
void dwc_ddrctl_cinit_custom_io_set_axi_clk(bool enable)	This API must be rewritten to the customer target platform. It allows controlling the memory controller AXI clock signal.
void dwc_ddrctl_cinit_custom_io_set_ddrc_clk(bool enable)	This API must be rewritten to the customer target platform. It allows controlling the memory controller clock signal.
void dwc_ddrctl_cinit_custom_io_usleep(uint32_t time)	This API must be rewritten to the customer target platform. Waits for a certain microseconds (time).

Pre-defined Functions	Description
void dwc_ddrctl_cinit_custom_io_nsleep(uint32_t time)	This API must be rewritten to the customer target platform. Waits for a certain nanoseconds (time).
bool dwc_ddrctl_cinit_custom_io_i2c_config(void)	This API must be rewritten to the customer target platform. It allows to configure/initialize the I2C controller.
bool dwc_ddrctl_cinit_custom_io_i2c_enable(void)	This API must be rewritten to the customer target platform. It allows enabling the I2C controller.
bool dwc_ddrctl_cinit_custom_io_i2c_disable(void)	This API must be rewritten to the customer target platform. It allows disabling the I2C controller.
bool dwc_ddrctl_cinit_custom_io_i2c_read(uint16_t address, uint8_t *data, uint16_t bytes)	This API must be rewritten to the customer target platform. It allows reading data through the I2C controller of a slave device.
void dwc_ddrctl_cinit_custom_io_switch_clk(uint32_t target_freq_num)	This API must be rewritten to the customer target platform. It changes the clock frequency to match the controller configuration.
bool dwc_ddrctl_cinit_custom_io_smbus_write(uint16_t ch, uint32_t msg, uint32_t smbus_info)	This API must be rewritten to the customer target platform. It writes on the SMBus.
void dwc_ddrctl_cinit_exit(uint32_t error_code);	API is called in case of a fatal error.

3.3 Configuration

3.3.1 Defconfig Files

The SINIT/CINIT software is configured using defconfig files.

They are simple text files with the non-default option set. Sconfig tool processes the defconfig files against a configuration options data base, that generates a C header file (.autoconf.h) to be used on CINIT compilation.

The format of this file is simple list of parameters that are assigned to values. Lines that start with '#' are considered as comments. Depending on the parameter the values can be set with strings, numbers, or hexadecimals.

The following is an example of the defconfig format:

```
#General configurations
SNPS_DEFCONFIG="<name_of_cofiguration>"
NUM_PSTATES=1
MEM_CTL_BASE_ADDRESS=0x00000000
DWC_DDRCTL_CINIT_SDRAM_PROTOCOL_DDR4=y
```

3.3.2 Sconfig Tool

The Python SConfig tool validates defconfig files, writes C headers and generates documentation specific to the HW configuration users.

Table 3-2 Operating Modes

Operating mode	Description
gen_config_header	Generate C configuration header
list_available	List available configuration options
gen_html_doc	Generate HTML documentation
print_reg_map	Show all valid registers and default values
gen_c_reg_map	Generate Register Map in C code
gen_config_header	Generate C configuration header

Table 3-3 Arguments

Arguments	Description
"-c", "--cc_constants"	C headers that contain the controller and phy HW configurations Required for all operating modes
"-m", "--memmap_csv"	csv file containing the controller memory map Required for all operating modes
"-d", "--defconfig"	Input defconfig file
"-o", "--output"	Output file used in multiple modes

Arguments	Description
"-r", "--report"	Path for report file when generating c header Default is output file with "_report" as a suffix
"-v", "--version"	Print version
"--show_all"	Show all parameters in documentation

3.3.2.1 Generate C Configuration Header

The "gen_config_header" mode receives a defconfig file and validates the configuration. Incorrect or deprecated parameters are reported. The C configuration header is written to the path specified with the output argument.

A report is generated in the format of a valid defconfig containing each parameter set by you, including parameters set to default. The tool provides comments with descriptions and help information for each parameter and its value. By default the report is generated with the output path and the "_report" suffix, can be overwritten with the report argument.

3.3.2.2 List of Available Configuration Options

The "list_available" mode writes a file containing all supported parameters for a particular HW configuration. The file has the same format as the report feature. For each parameter the tool provides comments with descriptions and information to how configure each option. The file path is set with the output argument.

3.3.2.3 Generate HTML Documentation

The "gen_html_doc" mode generates an html with all supported parameters for easy reading. Output directory is set with the output argument.

The generated documentation only shows parameters that are possibly supported by the HW described in the CC constants file. The "--show_all" arguments forces all parameters and possible values to be shown.

3.3.2.4 Show All Valid Registers and Default Values

The "print_reg_map" mode writes to the shell all the registers and default values for a particular memory map and HW configuration.

3.3.2.5 Generate Register Map in C Code

The "gen_c_reg_map" mode writes C headers used in CINIT containing the memory map information. Output directory is set with the output argument.

3.4 SINIT/CINIT Compilation

3.4.1 Available Make Commands

Use the makefile at sw_utilities/cinit as the main one.

Table 3-4 describes the available commands.

Table 3-4 Makefile Options

cinit/library Make Commands	Descriptions
make clean	Cleans all the builds objects and auxiliary files.
make clean-all	Performs the same operations as clean and removes auto-generated header files.
make -C apps/standalone	Builds and runs CINIT standalone.
make	Builds CINIT library.

3.4.2 Compilation Defines

The code uses several compilation flags to configure its behavior, they are passed directly by the make files.

List of defines used for compilation:

- USE_KCONFIG_DEFINITIONS: Uses the kconfig setting to setup the controller.
- DWC_DDRCTL_CINIT_CPU_DPI_MODE: Enables writing directly on the controller I/O calls.
- USE_VERIFICATION_DEFINITIONS: Uses this definition in the simulation.
- PHYINIT: Enables phy calls.
- STD_ALONE: Removes extern simulation calls.
- CINIT_TRACE: Enables SNPS_TRACE messages.
- CINIT_ENABLE_REG_FORCE_WRITE: When this option is enabled, all controller registers are written regardless of whether their values need an update from the default value or not. Using this option increases controller initialization time.

Protocol options:

- CINIT_DDR4: Enables DDR4 code.
- CINIT_DDR5: Enables DDR5 code.
- CINIT_LPDDR4: Enables LPDDR4, LPDDR4x code.
- CINIT_LPDDR5: Enables LPDDR5, LPDDR5x code.


Note You must not use the DDR and LPDDR options at the same time.

3.4.3 Create Standalone Application Using the Provided Defconfigs

1. Clean any pre-existing build (optional, just to start from a known and clean step):
   ```
   % make clean
   ```
2. Load hw config:
   ```
   % make load-hw-config PROTOCOL=<ddr4|ddr5|lpddr4|lpddr4x|lpddr5| lpddr5x>
   ```
3. Copy defconfig file:
   ```
   % cp <user_defconfig> testbench_defconfig
   ```
4. Load defconfig:
   ```
   % make load-defconfig
   ```
5. Compile and run the CINIT standalone:
   ```
   % make -C apps/standalone PROTOCOL=<ddr4|ddr5|lpddr4|lpddr4x|lpddr5| lpddr5x>
   ```
6. Execute the CINIT standalone:
   ```
   % apps/standalone/bin/cinit
   ```
7. The controller settings and log files are created on the CINIT workspace with the following names:
   - dwc_ddrctl_cinit.seq
   - dwc_ddrctl_cinit.log

3.4.4 Create Library to Integrate With External Applications

Note This method is recommended to integrate with external applications; other processes are not supported.

1. Follow the "Create Standalone Application Using the Provided Defconfigs" on page 19 steps from 1 to 5.
2. Create on the *bsp/platform* your specific platform definitions and explain in Chapter 3.2.4, "Platform".
3. Create the library:
   ```
   % make CINIT_PLATFORM=<platform> CINIT_ARCH=<arch>  CINIT_COMPILER=<compiler>
   PROTOCOL=<ddr4|ddr5|lpddr4|lpddr4x|lpddr5|lpddr5x> CINIT_OS=<os>
   ```

Note You must add the settings on the bsp folder to your system and then execute the command above with the CINIT_PLATFORM, CINIT_ARCH, CINIT_COMPILER and CINIT_OS parameters.

4. The generated dynamic library is at ../lib and includes the CINIT core, Sequences and Phyinit.
   - Ex: ../lib/ libcinit_phyinit_ddr4.so or libcinit_phyinit_ddr5.so
5. Integrate it to your application.

3.4.5 Create Standalone Application From Scratch (Advanced Approach)

Follow these steps:

1. Clean any pre-existing build (optional, just to start from a known and clean step):

   ```
   % make clean
   ```

2. Create a new defconfig file under the configs folder, for demonstration purpose the new defconfig is called "testbench_defconfig", and defines the values to be overridden.

   For more information on defconfig content, consult "Example Defconfig Configurations" on page 24:

   ```
   % touch configs/testbench_defconfig
   % vi configs/testbench_defconfig
   ```

3. Load a pre-existent defconfig available, for demonstration purpose the defconfig is called "testbench_defconfig":

   ```
   % make testbench_defconfig
   ```

4. Compile and run the CINIT standalone:

   ```
   % make -C apps/standalone PROTOCOL=<ddr4|ddr5|lpddr4|lpddr4x|lpddr5|lpddr5x>
   ```

5. After the successful execution of CINIT standalone, the controller settings and log files are created on the CINIT workspace with the names:

   - dwc_ddrctl_cinit.seq
   - dwc_ddrctl_cinit.log

3.4.6 Initialization Sequence and Log Files

The files are generated on the CINIT workspace:

- *dwc_ddrctl_cinit.seq* file contains the Synopsys DDR memory controller initialization.
- *dwc_ddrctl_cinit.log* file reflects the CINIT flow and debugging the steps.

Figure 3-1 Small Snippet of dwc_ddrctl_cinit.log

```
…
[LOG   ] [                                        dwc_cinit_dump_mmap.h] [ dwc_ddrctl_cinit_dump_init_mmap] [ 9645]
[REGB_FREQ3_CH1.PERFHPR1.hpr_max_starve = 1]
[LOG   ] [                                        dwc_cinit_dump_mmap.h] [ dwc_ddrctl_cinit_dump_init_mmap] [ 9646]
[REGB_FREQ3_CH1.PERFHPR1.hpr_xact_run_length = 15]
[LOG   ] [                                        dwc_cinit_dump_mmap.h] [ dwc_ddrctl_cinit_dump_init_mmap] [ 9647]
[REGB_FREQ3_CH1.PERFLPR1.lpr_max_starve =127]
[LOG   ] [                                        dwc_cinit_dump_mmap.h] [ dwc_ddrctl_cinit_dump_init_mmap] [ 9648]
[REGB_FREQ3_CH1.PERFLPR1.lpr_xact_run_length = 15]
…
```

Figure 3-2 Small Snippet of dwc_ddrctl_cinit.seq

```
…
dwc_ddrctl_wr(0x001005a8, 0x0070007a);
dwc_ddrctl_wr(0x00100a80, 0x000018e0);
dwc_ddrctl_wr(0x00100d08, 0x00001709);
dwc_ddrctl_wr(0x00100c80, 0x0f000001);
…
```

3.5 CINIT APIs

3.5.1 CINIT Begin

This is the first function to be called for controller initialization, it sets up the data structures.

void dwc_ddrctl_cinit_begin(SubsysHdlr_t *phdlr)

3.5.2 SDRAM Bus Width (Optional)

This function overrides the configured SDRAM BUS width.

void dwc_ddrctl_cinit_set_sdram_bus_width(SubsysHdlr_t *phdlr, dwc_sdram_bus_width_t bus_width)

3.5.3 CINIT Main

Executes the controller initialization based on the configuration.

void dwc_ddrctl_cinit_main(SubsysHdlr_t *phdlr)

3.5.4 CINIT End

This is the termination function that you must call at the end to close opened files.

void dwc_ddrctl_cinit_end(void)

3.6 SINIT Flow Used in External Applications

1. Prepare the CINIT setup and data structures: dwc_ddrctl_cinit_begin()
2. Update the SDRAM Bus Width if needed (optional): dwc_ddrctl_cinit_set_sdram_bus_width()
3. Execute the DDR controller initialization: dwc_ddrctl_cinit_main()
4. Execute the SINIT sequences required for your case

Figure 3-3 Example

```
if (dwc_ddrctl_cinit_seq_initialization() == true){
            SNPS_LOG("Initialization completed successfully",NULL);
            dwc_ddrctl_cinit_io_wait_ddrc_clk(512);
            dwc_ddrctl_cinit_seq_wait_ctrlr_idle(5 * DWC_DDRCTL_MAX_APB_POLLING);
}
```

Note

dwc_ddrctl_cinit_seq_initialization does the following:

- Performs power on reset and enables clocks.
- Initializes the controller register map.
- Releases core_rst_n to the controller.
- De-asserts aresetn.
- Performs PHY initialization sequence.
- Initializes PHY to mission mode by performing DFI initialization sequence per DFI specification.
- Polls for normal operating mode.
- Memory sub-system is now ready, enables HIF interface.
- Enables all application ports.

dwc_ddrctl_cinit_io_wait_ddrc_clk calls the following:

- dwc_ddrctl_cinit_custom_io_wait_ddrc_clk

dwc_ddrctl_cinit_seq_wait_ctrlr_idle does the following:

- Waits until the controller is idle.

5. Exit and cleanup CINIT: dwc_ddrctl_cinit_end()

3.7 Tools Version Requirement

To compile CINIT the following tools are required:

Table 3-5 Tools for Compiling CINIT

Tool	Version
make	3.82
GCC	9.2.0
python	3.8.1

3.8 Example Defconfig Configurations

This chapter shows some configuration examples as your guidance to do through the CINIT configuration tool or by overriding directly on the defconfig file.

3.8.1 Defconfig Speed Grade: DDR4 4GB 3200MTs 1x72 x8 UDIMM

```
#General configurations
SNPS_DEFCONFIG="DDR4_4GB_3200MTs_1x72_x8_UDIMM"
NUM_PSTATES=1


#DDRC channel #0
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_WR_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_RD_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_SCHED0_LPR_NUM_ENTRIES=16
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_PARITY_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_MSTR0_ACTIVE_RANKS=1
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CHCTL_DUAL_CHANNEL_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_HWLPCTL_HW_LP_EXIT_IDLE_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_DFIUPDTMG1_DFI_T_CTRLUPD_INTERVAL_MAX_X1024
=2
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_DFIUPDTMG1_DFI_T_CTRLUPD_INTERVAL_MIN_X1024
=1
# Initialization Speedups
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_INITTMG0_PRE_CKE_X1024=2
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_INITTMG0_POST_CKE_X1024=2


#Frequency #0
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_DRAMSET1TMG9_DDR4_WR_PREAMBLE_AUX=n
DWC_DDRCTL_CINIT_DDR4_PSTATE0_MR4_RD_PREAMBLE=0
NUM_DCH=1
#Memory Configuration
DWC_DDRCTL_CINIT_SDRAM_PROTOCOL_DDR4=y
DWC_DDRCTL_CINIT_SDRAM_CAPACITY_MBIT_0_4_GB=y
DWC_DDRCTL_CINIT_DDR4_SG_3200W=y
DWC_DDRCTL_CINIT_NUM_RANKS_1_RANK=y
#Misses the primary bus configuration
DWC_DDRCTL_CINIT_SDRAM_WIDTH_BITS_0_8_BITS=y
DWC_DDRCTL_CINIT_MODULE_TYPE_UDIMM=y
```

```
# Skip PHY training
CONFIG_DWC_SKIP_TRAINING=y
```

3.8.2 Defconfig JEDEC Timings: DDR4 16GB 3200MTs 1x72 x16 UDIMM

```
#General configurations
SNPS_DEFCONFIG="DDR4_16GB_1600_3200MTs_1x72_x8_UDIMM"
NUM_PSTATES=1

#DDRC channel #0
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_WR_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_RD_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_SCHED0_LPR_NUM_ENTRIES=16
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_PARITY_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_MSTR0_ACTIVE_RANKS=15
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DFIUPD0_DIS_AUTO_CTRLUPD_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DFIUPD0_DIS_AUTO_CTRLUPD_SRX_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_RFSHCTL0_DIS_AUTO_REFRESH_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_ZQCTL0_DIS_AUTO_ZQ_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_ZQCTL2_DIS_SRX_ZQCL_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DBICTL_RD_DBI_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DBICTL_WR_DBI_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_SCHED0_DIS_SPECULATIVE_ACT_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CHCTL_DUAL_CHANNEL_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_HWLPCTL_HW_LP_EXIT_IDLE_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_DFIUPDTMG1_DFI_T_CTRLUPD_INTERVAL_MAX_X1024
=2
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_DFIUPDTMG1_DFI_T_CTRLUPD_INTERVAL_MIN_X1024
=1
# Initialization Speedups
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_INITTMG0_PRE_CKE_X1024=2
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_INITTMG0_POST_CKE_X1024=2

#Frequency #0
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_DRAMSET1TMG9_DDR4_WR_PREAMBLE_AUX=n
DWC_DDRCTL_CINIT_DDR4_PSTATE0_MR4_RD_PREAMBLE=0
NUM_DCH=1
```

```
#Memory Configuration
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC=y
DWC_DDRCTL_CINIT_SDRAM_PROTOCOL_DDR4=y
DWC_DDRCTL_CINIT_SDRAM_CAPACITY_MBIT_0_16_GB=y
DWC_DDRCTL_CINIT_SDRAM_WIDTH_BITS_0_8_BITS=y
DWC_DDRCTL_CINIT_NUM_RANKS_4_RANK=y

DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_CAS_LATENCY_SUPPORT_RANGE_CL_16=y
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_CAS_LATENCY_SUPPORT_RANGE_CL_20=y

DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TCK_MIN=1250
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TCK_MAX=1250
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRCD_MIN=12500
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRP_MIN=12500
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRAS_MIN=32000
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRC_MIN=44500
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRFC1_MIN=550000
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRFC2_MIN=350000
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRFC4_MIN=260000
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRRD_S_MIN=5500
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TRRD_L_MIN=6500
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TCCD_L_MIN=5000
DWC_DDRCTL_CINIT_SPD_STATIC_JEDEC_DDR4_TFAW_MIN=30000

# Skip PHY training
CONFIG_DWC_SKIP_TRAINING=y
```

3.8.3 Defconfig SPD Memory: DDR4 16GB 3200MTs 1x72 x16 UDIMM

In the dynamic mode, the SPD binary must be created and located in the correct place. The SPD binary for this example can be created according to the “Example of Generating a Binary SPD File for DDR4” on page 31.

```
#General configurations
SNPS_DEFCONFIG="DDR4_16GB_3200MTs_1x72_x16_UDIMM"
NUM_PSTATES=1
```

```
#DDRC channel #0
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_WR_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_RD_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_SCHED0_LPR_NUM_ENTRIES=16
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_PARITY_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_MSTR0_ACTIVE_RANKS=15
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DFIUPD0_DIS_AUTO_CTRLUPD_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DFIUPD0_DIS_AUTO_CTRLUPD_SRX_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_RFSHCTL0_DIS_AUTO_REFRESH_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_ZQCTL0_DIS_AUTO_ZQ_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_ZQCTL2_DIS_SRX_ZQCL_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_OCPARCFG0_OC_PARITY_TYPE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DBICTL_RD_DBI_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DBICTL_WR_DBI_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_SCHED0_DIS_SPECULATIVE_ACT_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CHCTL_DUAL_CHANNEL_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_HWLPCTL_HW_LP_EXIT_IDLE_EN_AUX=n

# Initialization Speedups
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_INITTMG0_PRE_CKE_X1024=2
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_INITTMG0_POST_CKE_X1024=2

#Frequency #0
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_DRAMSET1TMG9_DDR4_WR_PREAMBLE_AUX=n
DWC_DDRCTL_CINIT_DDR4_PSTATE0_MR4_RD_PREAMBLE=0
NUM_DCH=1

# Memory Configuration
DWC_DDRCTL_CINIT_SPD_DYNAMIC_JEDEC=y

# Skip PHY training
CONFIG_DWC_SKIP_TRAINING=y
```

3.8.4 Defconfig Speed Grade: DDR5 16GB 4800MTs 2x40 x16 UDIMM

```
#General configurations
SNPS_DEFCONFIG="DDR5_16GB_4800MTs_2x40_x16_UDIMM"
```

```
NUM_PSTATES=1
CONFIG_NUM_DCH_2=y

#DDRC channel #0
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_WR_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_RD_CRC_ENABLE_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_SCHED0_LPR_NUM_ENTRIES=16
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CRCPARCTL1_PARITY_ENABLE_AUX=n
# When ECC enabled in UDIMM for DDR5 then Multi beat must be selected
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_MSTR0_ACTIVE_RANKS=15
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_MSTR0_DATA_BUS_WIDTH=1
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DFIUPD0_DIS_AUTO_CTRLUPD_SRX_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_RFSHCTL0_DIS_AUTO_REFRESH_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_ZQCTL2_DIS_SRX_ZQCL_AUX=n
# When Multi beat ECC is selected then Data mask must be disabled
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_DBICTL_DM_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_CHCTL_DUAL_CHANNEL_EN_AUX=y
DWC_DDRCTL_CINIT_REGB_FREQ0_CH0_TMGCFG_FREQUENCY_RATIO_AUX=y
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_PASCTL2_T_PPD_CNT_EN_AUX=y
DWC_DDRCTL_CINIT_REGB_DDRC_CH0_HWLPCTL_HW_LP_EN_AUX=n
DWC_DDRCTL_CINIT_REGB_DDRC_CH1_HWLPCTL_HW_LP_EN_AUX=n
# Enable DDR5 DQSOSC in both channels
DWC_DDRCTL_CINIT_TCR_DQSOSC_EN_0_0_ENABLE=y
DWC_DDRCTL_CINIT_TCR_DQSOSC_EN_0_1_ENABLE=y
DWC_DDRCTL_CINIT_TCR_DQSOSC_EN_1_0_ENABLE=y
DWC_DDRCTL_CINIT_TCR_DQSOSC_EN_1_1_ENABLE=y

# Enable all rank ZQCAL
DWC_DDRCTL_CINIT_ALL_RANK_ZQCAL_EN_0_ENABLE=y
DWC_DDRCTL_CINIT_ALL_RANK_ZQCAL_EN_1_ENABLE=y
DWC_DDRCTL_CINIT_CTLUPD_EN_0_ENABLE=y
DWC_DDRCTL_CINIT_CTLUPD_EN_1_ENABLE=y

#DDRC channel #1
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR8_WR_POSTAMBLE=1
```

```
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR8_RD_POSTAMBLE=0
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR8_RD_PREAMBLE=3
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR8_WR_PREAMBLE=1

# When Multi beat ECC is selected then Data mask must be disabled
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR5_DM_ENABLE=0
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR34_RTT_PARK=3
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR34_RTT_WR=1
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR35_RTT_NOM_WR=3
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR35_RTT_NOM_RD=3
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR37_ODTLON_WR_OFFSET=4
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR37_ODTLOFF_WR_OFFSET=5
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR38_ODTLON_WR_NT_OFFSET=4
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR38_ODTLOFF_WR_NT_OFFSET=5
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR39_ODTLON_RD_NT_OFFSET=4
DWC_DDRCTL_CINIT_DDR5_PSTATE0_MR39_ODTLOFF_RD_NT_OFFSET=5

#Memory Configuration
DWC_DDRCTL_CINIT_SDRAM_PROTOCOL_DDR5=y
DWC_DDRCTL_CINIT_SDRAM_CAPACITY_MBIT_0_16_GB=y
DWC_DDRCTL_CINIT_DDR5_SG_0_4800AN=y
DWC_DDRCTL_CINIT_NUM_RANKS_4_RANK=y

#Misses the primary bus configuration
DWC_DDRCTL_CINIT_SDRAM_WIDTH_BITS_0_16_BITS=y
DWC_DDRCTL_CINIT_SDRAM_WIDTH_BITS_1_16_BITS=y

DWC_DDRCTL_CINIT_MODULE_TYPE_UDIMM=y
# Skip PHY training
CONFIG_DWC_SKIP_TRAINING=y
```

3.8.5 SPD Emulation Through a Binary File

For the testing purposes, on testbench is emulated the SPD data read through a read of a binary file. In some cases, it can be useful to implement the same approach, while it cannot be tested on a real system. In that case, you can re-implement the same approach by following what has been implemented on *bsp/platform/simulation/src/dwc_ddrctl_cinit_custom_io.c* and applying on *bsp/platform/local_x86/src/dwc_ddrctl_cinit_custom_io.c.*

To use the binary SPD file emulation you have to select and use the option "Dynamic by reading SPD memory".

A tool for generating an SPD binary file to emulate a real SPD is available and located on the path:

<*CINIT workspace*>*/tools/spd_generator/*

To compile it execute make in that folder.

```
% make
```

Command syntax:

spd_generator -p <*protocol*> *-i* <*configuration parameters file*> *-o* <*binary SPD file*>

Where <*protocol*> parameter can be:

- 4 for DDR4 protocol
- 5 for DDR5 protocol

The configuration parameter file structure differs according to the protocol selected, consult the “Example of Generating a Binary SPD File for DDR4” on page 31 and “Example of Generating a Binary SPD File for DDR5” on page 32 for the specific parameters allowed.

3.8.6 Example of Generating a Binary SPD File for DDR4

This example describes how to create a DDR4, 16GB 3200MTs 1x72 x16 UDIMM to be used in "Defconfig SPD Memory: DDR4 16GB 3200MTs 1x72 x16 UDIMM" on page 26.

Create a text file with the name DDR4_16GB_3200MTs_1x72_x16_UDIMM.ini and the following content.

Figure 3-4 Content of the DDR4_16GB_3200MTs_1x72_x16_UDIMM.ini File

```
type 2
capacity 6
device_ranks 3
col_address 0
row_address 0
banks 0
bank_group 0
device_width 2
bus_width 0
ecc 1
package_type 0
die_count 0
signal_loading 0
CL_supported 8704
tCKAVGmin 625
tCKAVGmax 625
tAAmin 13750
tRCDmin 12500
tRPmin 12500
tRASmin 32000
tRCmin 44500
tRFC1min 550000
tRFC2min 350000
tRFC4min 260000
tRRD_S_min 5500
tRRD_L_min 6500
tWRmin 15000
tWTR_S_min 2500
tWTR_L_min 7500
tCCD_L_min 5000
tFAWmin 30000
```

1. Execute the SPD generator tool (spd_generator) to create the binary SPD:

   ```
   % spd_generator -p 4 -i DDR4_16GB_3200MTs_1x72_x16_UDIMM.ini -o SPD.bin
   ```

2. If the command execution is successful, the output is:

Figure 3-5 Binary SPD Output File

```
Fundamental Memory Class = DDR4 SDRAM
Base Module Type = UDIMM
Module Capacity = 16 Gb
Number of DIMM Ranks = 4
Number of Column Addresses = 9 bits
Number of Row Addresses = 12 bits
Number of Bank Addresses = 2 bits (4 banks)
Bank Group Addressing = 0 bit (no groups)
DRAM Device Width = 16 bits
Primary Memory Bus Width = 8 bits
Memory Bus Width Extension = 8 bits
DRAM Device Package = Standard Monolithic
DRAM Device Die Count = Single die
Signal Loading = Not specified
CAS Latencies Supported: 20T 16T
Minimum Clock Cycle Time (tCK min): Maximum        625 ps      tCKAVG_min_mtb = 5   tCKAVG_min_ftb = 0
Clock Cycle Time (tCK max):                        625 ps      tCKAVG_max_mtb = 5   tCKAVG_max_ftb = 0
CAS# Latency Time (tAA min):                       13750 ps    tAA_min_mtb = 110    tAA_min_ftb = 0
RAS# to CAS# Delay Time (tRCD min):                12500 ps    tRCD_min_mtb = 100   tRCD_min_ftb = 0
Row Precharge Delay Time (tRP min):                12500 ps    tRP_min_mtb = 100    tRP_min_ftb = 0
Active to Precharge Delay Time (tRAS min):         32000 ps    tRAS_min_mtb = 256
Act to Act/Refresh Delay Time (tRC min):           44500 ps    tRC_min_mtb = 356
Normal Refresh Recovery Delay Time (tRFC1 min):    550000 ps   tRFC1_min_mtb = 4400
2x mode Refresh Recovery Delay Time (tRFC2 min):   350000 ps   tRFC2_min_mtb = 2800
4x mode Refresh Recovery Delay Time (tRFC4 min):   260000 ps   tRFC4_min_mtb = 2080
Short Row Active to Row Active Delay (tRRD_S min): 5500 ps     tRRD_S_min_mtb = 44  tRRD_S_min_ftb = 0
Long Row Active to Row Active Delay (tRRD_L min):  6500 ps     tRRD_L_min_mtb = 52  tRRD_L_min_ftb = 0
Write Recovery Time (tWR min):                     15000 ps    tWR_min_mtb = 120
Short Write to Read Command Delay (tWTR_S min):    2500 ps     tWTR_S_min_mtb = 20
Long Write to Read Command Delay (tWTR_L min):     7500 ps     tWTR_L_min_mtb = 60
Long CAS to CAS Delay Time (tCCD_L min):           5000 ps     tCCD_L_min_mtb = 40  tCCD_L_min_ftb = 0
Four Active Windows Delay (tFAW min):              30000 ps    tFAW_min_mtb = 240
DDR4 Binary SPD dump file = 512 bytes
```

3.8.7 Example of Generating a Binary SPD File for DDR5

This example describes how to create a DDR5, 16GB 4800MTs 2x40 x8 RDIMM.

Create a text file with the name DDR5_16GB_4800MTs_2x40_x8_RDIMM.ini and the following content.

Figure 3-6 Content of the DDR5_16GB_4800MTs_2x40_x8_RDIMM.ini File

```
type 1
first_capacity 4
second_capacity 4
first_banks 1
second_banks 0
first_col_address 0
second_col_address 0
first_row_address 0
second_row_address 0
first_banks 2
first_bank_group 3
second_bank_group 3
first_device_width 1
second_device_width 1
CL_supported 2048
tCKAVGmin 416
tCKAVGmax 416
tAAmin 13750
tRCDmin 17472
tRPmin 17472
tRASmin 32000
tRCmin 49472
tWRmin 29952
tRFC1_slr_min 295
tRFC2_slr_min 160
tRFCsb_slr_min 13
tRFC1_dlr_min 99
tRFC2_dlr_min 99
tRFCsb_dlr_min 99
number_package_ranks_channel 2
primary_bus_width 2
bus_width_extension 1
```

1. Execute the SPD generator tool (spd_generator) to create the binary SPD:

   ```
   % spd_generator -p 5 -i DDR5_16GB_4800MTs_2x40_x8_RDIMM.ini -o SPD.bin
   ```

2. If the command execution is successful, the output is:

Figure 3-7 Binary SPD Output File

```
Fundamental Memory Class = DDR5 SDRAM
Base Module Type = RDIMM
First Module Capacity = 16 Gb
Second Module Capacity = 16 Gb
First Number of Bank Addresses = 2 banks
Second Number of Bank Addresses = 1 banks
First Number of Column Addresses = 10 bits
Second Number of Column Addresses = 10 bits
First Number of Row Addresses = 16 bits
Second Number of Row Addresses = 16 bits
First Number of Bank Addresses = 3 banks
First Bank Group Addressing = 8 bank group
Second Bank Group Addressing = 8 bank group
First DRAM Device Width = x8
Second DRAM Device Width = x8
CAS Latencies Supported: 26T
Minimum Clock Cycle Time (tCK min):                          416 ps
Maximum Clock Cycle Time (tCK max):                          416 ps
CAS# Latency Time (tAA min):                                 13750 ps
RAS# to CAS# Delay Time (tRCD min):                          17472 ps
Row Precharge Delay Time (tRP min):                          17472 ps
Active to Precharge Delay Time (tRAS min):                   32000 ps
Act to Act/Refresh Delay Time (tRC min):                     49472 ps
Write Recovery Time (tWR min):                               29952 ps
SDRAM Minimum Refresh Recovery Delay Time (tRFC1min, tRFC1_slr min):    295 ns
SDRAM Minimum Refresh Recovery Delay Time (tRFC2min, tRFC2_slr min):    160 ns
SDRAM Minimum Refresh Recovery Delay Time (tRFCsbmin, tRFCsb_slr min):  13 ns
SDRAM Minimum Refresh Recovery Delay Time, 3DS Different Logical Rank (tRFC1_dlr min):  99 ns
SDRAM Minimum Refresh Recovery Delay Time, 3DS Different Logical Rank (tRFC2_dlr min):  99 ns
SDRAM Minimum Refresh Recovery Delay Time, 3DS Different Logical Rank (tRFCsb_dlr min): 99 ns
Number of Package Ranks per channel = 3 Package rank
Primary bus width per channel = 32 bits
Bus width extension per channel = 4 bits (no extension)
ddr5 Binary SPD dump file = 1024 bytes
```

3.9 SINIT Validated Configurations on DDR5/4 Controller

PHY Type	Memory Type	DIMM Types	Ranks	PHY Train Modes Tested				
				Skip	Dev Init	Full Train	Full Train (board delays)	Notes
DDR5/4 PHY	DDR4	RDIMM	4	Yes	Yes	No	Yes	
DDR5/4 PHY	DDR4	UDIMM	1	Yes	Yes	Yes	Yes	
DDR5/4 PHY	DDR4	UDIMM	4	Yes	Yes	Yes	Yes	
DDR5/4 PHY	DDR5	None	4	Yes	Yes	No	No	
DDR5/4 PHY	DDR5	RDIMM	4	Yes	Yes	Yes	No	
DDR5/4 PHY	DDR5	UDIMM	1	Yes	Yes	Yes	Yes	
DDR5/4 PHY	DDR5	UDIMM	4	Yes	Yes	Yes	No	2 DIMMs
DDR5/4 PHY	DDR5	LRDIMM	1	Yes	Yes	No	No	
DDR5/4 PHY	DDR5 3DS	None	2	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR4	None	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR4	RDIMM	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR4	UDIMM	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR4	LRDIMM	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR5	None	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR5	RDIMM	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR5	UDIMM	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR5	LRDIMM	1	Yes	Yes	No	No	
DDR5/4 PHY V2	DDR5	RDIMM	4	Yes	Yes	No	No	

Note Contact Synopsys for the latest DDR5/4 controller supported configurations

3.10 SINIT Validated Configurations on LPDDR5/4 Controller

PHY Type	Memory Type	DIMM Types	Ranks	PHY Train Modes Tested			
				Skip	Dev Init	Full Train	Notes
LDDR5/4/4X	LPDDR4	RDIMM	2	Yes	Yes	No	

PHY Type	Memory Type	DIMM Types	Ranks	PHY Train Modes Tested			
				Skip	Dev Init	Full Train	Notes
DDR5/4/4X	LPDDR5	UDIMM	2	Yes	Yes	No	

3.11 Known Issues

- SINIT is not supported in CoreAssembler in LPDDR5/4/4X, and is not fully supported in DDR5/4.
- SAR is not supported in SINIT.
- Heterogeneous ranks are not fully supported.

4 DDR Subsystem Rank-to-Rank Space Timing

DFI rank-to-rank space timing must be determined by the actual board delay (DQ/DQS Bus Turnarounds). DDRPHY firmware training results can provide critical delay difference (CDD) information to help to calculate the minimum required timing spacing (that is, memory controller registers).

This chapter describes how to update the spacing (including single rank as well as multi-ranks for read-to-write, write-to-read, and rank-to-rank) by utilizing the firmware results after training. It contains the following sections:

- "Timing Spacing" on page 38 describes how the board delay impacts the DFI timing spacing.
- "Controller Registers Configuration" on page 41 describes the affected DDRCTL registers by board delay and how to configure these registers to meet the timing.
- "Read Training Results" on page 45 provides an example to get the firmware returned CDD values.

DDR Subsystem Rank-to-Rank Space Timing applies to the following products:

- DesignWare Cores DDR5/4 PHY
- DesignWare Cores Enhanced Universal DDR Memory Controller (DDRCTL)

4.1 Timing Spacing

This section describes how the board delay impacts the DFI timing spacing. For more details, refer to the section “Rank-to-Rank Spacing” in *DDRPHY PUB Databook*.

Note MEMCLK (as well as tCK) specifies the period of DRAM CK.

4.1.1 tphy_wrcsgap

Minimum possible Rank-to-Rank for Write-Write (W-W) transaction are given with the tphy_wrcsgap parameter.

```
tCCDmin (W_rank[i], W_rank[j]) = 4 + MAX[0, CDD_WW_R[i]_R[j]]
+PreamblePostambleAdjust (DDR4)
tCCDmin (W_rank[i], W_rank[j]) = 5 + MAX[0, CDD_WW_R[i]_R[j]]
+PreamblePostambleAdjust +D5PositionTxPhaseUpdateAdjust (DDR5)
```

Parameter	Units	Description
CDD_WW_R[i]_R[j]	MEMCLKs	PHY FW returned values, indicates write to write critical delay difference from rank i to rank j. The range of i/j is: ■ DDR4/5: i/j = [0..3], and i≠j
tCCDmin (W_rank[i], W_rank[j])	MEMCLKs	Indicates the number of clocks of gap in data responses when performing consecutive writes to different ranks.
PreamblePostambleAdjust	MEMCLKs	DDR4 calculation is based on 1tck write preamble. When 2tck write preamble is enabled, transaction spacing must be increased by 1tck. DDR5 calculation is based on 2tck write preamble. When 3tck or 4tck write preamble is enabled, transaction spacing must be increased by 1tck or 2 tck accordingly. And it is based on 0.5tck write postamble. When 1.5tck write postamble is enabled, transaction spacing must be increased by 1tck.
D5PositionTxPhaseUpdateAdjust	MEMCLKs	Value of PositionTxPhaseUpdate field of CSR DqsPreambleControl_i. If and only if 2tck write preamble is used and either EnTxDqPreamblePattern = 0xF or EnTxDmPreamblePattern = 0xF is used, transaction spacing must be increased by 1 tck.

4.1.2 tphy_rdcsgap

Minimum possible Rank-to-Rank for Read-Read (R-R) transaction are given with the tphy_rdcsgap parameter.

```
tCCDmin (R_rank[i], R_rank[j]) = 4 + MAX[0, CDD_RR_R[i]_R[j]]
+PreamblePostambleAdjust (DDR4)
tCCDmin (R_rank[i], R_rank[j]) = 5 + MAX[0, CDD_RR_R[i]_R[j]]
+PreamblePostambleAdjust +D5PositionRxPhaseUpdateAdjust (DDR5)
```

Parameter	Units	Description
CDD_RR_R[i]_R[j]	MEMCLKs	PHY FW returned values, indicates read to read critical delay difference from rank i to rank j. The range of i/j is: ■ DDR4/5: i/j = [0..3], and i≠j
tCCDmin (R_rank[i], R_rank[j])	MEMCLKs	Indicates the number of clocks of gap in data responses when performing consecutive reads to different ranks.
PreamblePostambleAdjust	MEMCLKs	This calculation is based on 1 tck read preamble. When 2 tck or 3 tck or 4 tck read preamble is enabled, transaction spacing must be increased by 1tck or 2tck or 3tck accordingly and based on 0.5 tck read postamble. When 1.5 tck read postamble is enabled, transaction spacing must be increased by 1 tck.
D5PositionRxPhaseUpdateAdjust	MEMCLKs	Value of PositionRxPhaseUpdate field of CSR DqsPreambleControl.

4.1.3 rd2wr

Read DQS postamble received by the PHY must complete before the PHY attempts to transmit the Write DQS preamble. The value of the PHY FW returned values CDD_RW_R[i]_R[j] can be used to adjust scheduling of reads to writes.

```
tCCDmin (R_rank[i], W_rank[j]) = System_R2W + 5 + max(0,CDD_RW_R[i]_R[j])
(+5MEMCLK is not needed for same rank)
```

Parameter	Units	Description
System_R2W	MEMCLKs	Specifies the minimum time from read command to write command. ■ DDR4: RL + BL/2 + 1 + WR_PREAMBLE - WL (same rank) ■ DDR4: RL + BL/2 + 2 - WL (different physical rank) ■ DDR5: CL - CWL + RBL/2 + 2tCK - (Read DQS offset) + (tRPST - 0.5tCK) + tWPRE (same rank) ■ DDR5: RBL/2 + CL - CWL + RD_POSTAMBLE + WR_PREAMBLE (different physical rank) For more information, refer to DRAMSET1TMG2.rd2wr in *DDR5/4 Memory Controller Reference Manual*.
CDD_RW_R[i]_R[j]	MEMCLKs	PHY FW returned values, indicates read to write critical delay difference from rank i to rank j. The range of i/j is: ■ DDR4/5: i/j = [0..3] For example, for single rank i=j=0.
tCCDmin (R_rank[i], W_rank[j])	MEMCLKs	Indicates the number of clocks of gap in data responses when performing read to write.

4.1.4 wr2rd

Write DQS transmitted by PHY must be observed to be completed at all Ranks before any Rank attempts to transmit the Read DQS. The value of the PHY FW returned values CDD_WR_R[i]_R[j] can be used to adjust scheduling of reads to writes.

```
tCCDmin (W_rank[i], R_rank[j]) = System_W2R  + 5 + max(0,CDD_WR_R[i]_R[j])
(+5MEMCLK is not needed for same rank)
```

Parameter	Units	Description
System_W2R	MEMCLKs	Specifies the minimum time from write command to read command. ■ DDR4: CWL + PL + BL/2 + tWTR_L (same bank group) ■ DDR4: CWL + PL + BL/2 + tWTR_S (different bank group) ■ DDR5: CWL + BL/2 + tWTR_L (same bank group) ■ DDR5: CWL + BL/2 + tWTR_S (different bank group) For more information, refer to DRAMSET1TMG2.wr2rd and DRAMSET1TMG9.wr2rd_s in *DDR5/4 Memory Controller Reference Manual*.
System_W2R_DR	MEMCLKs	Specifies minimum time from write command to read command on different physical ranks. ■ DDR4: WL + BL/2 - RL + WR_POSTAMBLE + RD_PRE-AMBLE For more information, refer to RANKTMG1.wr2rd_dr (DDR4) in *DDR5/4 Memory Controller Reference Manual*.
CDD_WR_R[i]_R[j]	MEMCLKs	PHY FW returned values, indicates write to read critical delay difference from rank i to rank j. The range of i/j is: ■ DDR4/5: i/j = [0..3] For example, for single rank i=j=0.
tCCDmin (W_rank[i], R_rank[j])	MEMCLKs	Indicates the number of clocks of gap in data responses when performing write to read.

4.2 Controller Registers Configuration

This section describes which DDRCTL registers are affected and how to configure these registers to meet the timing spacing.

DDR4

The following controller registers need to be updated based on the formula in the section "Timing Spacing" on page 38 (refer to *DDR5/4 Memory Controller Reference Manual* for detailed description and setting constraints).

`RANKTMG0.diff_rank_wr_gap`

Only present for multi-rank configurations. Indicates the number of clocks of gap in data responses when performing consecutive writes to different ranks. This is used to switch the delays in the PHY to match the rank requirements. This value must consider both PHY requirement and ODT requirement.

`RANKTMG0.diff_rank_rd_gap`

Only present for multi-rank configurations. Indicates the number of clocks of gap in data responses when performing consecutive reads to different ranks. This is used to switch the delays in the PHY to match the rank requirements. This value must consider both PHY requirement and ODT requirement.

`RANKTMG1.rd2wr_dr`

Minimum time from read command to write command for different rank. Includes time for bus turnaround, recovery times, and all per-bank, per-rank, and global constraints. The value must be larger than or equal to the value of DRAMSET1TMG2.rd2wr.

`DRAMSET1TMG2.rd2wr`

Minimum time from read command to write command. Includes time for bus turnaround and all per-bank, per-rank, and global constraints.

`DRAMSET1TMG2.wr2rd`

In DDR4, minimum time from write command to read command for same bank group. In others, minimum time from write command to read command. Includes time for bus turnaround, recovery times, and all per-bank, per-rank, and global constraints.

`RANKTMG1.wr2rd_dr`

Minimum time from write command to read command on different physical ranks. This must include time for bus turnaround between ranks and all PHY and system requirements.

`DRAMSET1TMG9.wr2rd_s`

Minimum time from write command to read command for different bank group. Includes time for bus turnaround, recovery times, and all per-bank, per-rank, and global constraints. Present only in designs configured to support DDR4.

`DFITMG1. dfi_t_wrdata_delay`

Specifies the number of DFI clock cycles between when the dfi_wrdata_en signal is asserted and when the corresponding write data transfer is completed on the DRAM bus.

When TMGCFG.frequency_ratio is set to 0 (1:2 Mode), divided the value by 2 and round it up to the next integer value. When TMGCFG.frequency_ratio is set to 1 (1:4 Mode), divided the value by 4 and round it up to the next integer value. Unit: DFI clock cycles.

When enable board delay during simulation or run test in silicon, above registers must be updated according to actual board delay, the values have been captured in the message block of firmware.

DDR5

`RANK_SWITCH_TIMING_CONTROL(0-5).t_wr2wr_gap_rYrX`

Indicates the number of clocks of gap in data responses when performing consecutive writes from RankY to RankX. This is used to switch the delays in the PHY to match the rank requirements. This value must consider PHY requirement.

`RANK_SWITCH_TIMING_CONTROL (0-5).t_rd2rd_gap_rYrX`

Indicates the number of clocks of gap in data responses when performing consecutive reads from RankY to RankX. This is used to switch the delays in the PHY to match the rank requirements. This value must consider PHY requirement.

`RANK_SWITCH_TIMING_CONTROL (0-5).t_wr2rd_gap_rYrX`

Indicates the number of clocks of gap in data responses when performing write to read switch from RankY to RankX. This is used to switch the delays in the PHY to match the rank requirements. This value must consider PHY requirement.

`RANK_SWITCH_TIMING_CONTROL (0-5).t_rd2wr_gap_rYrX`

Indicates the number of clocks of gap in data responses when performing read to write switch from RankY to RankX. This is used to switch the delays in the PHY to match the rank requirements. This value must consider PHY requirement.

`DRAMSET1TMG9.wr2rd_s`

Minimum time from write command to read command for different bank group. Includes time for bus turnaround, recovery times, and all per-bank, per-rank, and global constraints.

`DRAMSET1TMG2.wr2rd`

Minimum time from write command to read command for same bank group. Includes time for bus turnaround, recovery times, and all per-bank, per-rank, and global constraints.

`DRAMSET1TMG2.rd2wr`

Minimum time from read command to write command. Includes time for bus turnaround and all per-bank, per-rank, and global constraints.

`DRAMSET1TMG22.rd2wr_dlr`

Read to write switch to different logical rank in same physical rank.

`DRAMSET1TMG22.wr2rd_dlr`

Write to read switch to different logical rank in same physical rank.

`DFITMG1.dfi_t_wrdata_delay`

Specifies the number of DFI clock cycles between when the dfi_wrdata_en signal is asserted and when the corresponding write data transfer is completed on the DRAM bus.

When TMGCFG.frequency_ratio is set to 0 (1:2 Mode), divided the value by 2 and round it up to the next integer value. When TMGCFG.frequency_ratio is set to 1 (1:4 Mode), divided the value by 4 and round it up to the next integer value. Unit: DFI clock cycles.

The section “DDR4” on page 43 provides example usage for DDRx based on 1:1 frequency ratio. When the controller is operating in 1:2 mode, the value must be divided by two and round up to the next integer.

The trained CDD results are per-Pstate (PHY support up to 4 Pstates, each Pstate has related CDD message blocks), however, above controller registers are not per-Pstate. Therefore, after PHY initialization, all CDD

results for each Pstate must be stored, and when frequency change, above controller registers must be updated accordingly based on Pstate-related CDD results.

4.2.1 DDR4

Check the mnPmuSramMsgBlock_ddr4.h (i,j=[0..3], indicates the timing group) for all of available CDD fields.

- `CDD_RR_[i]_[j]`
- `CDD_WW_[i]_[j]`
- `CDD_RW_[i]_[j]`
- `CDD_WR_[i]_[j]`

For the description of the CDD fields, refer to the tables in the section "Timing Spacing" on page 38.

Controller registers can be adjusted based on above CDD values.

- DDR4, only DFI0 of PHY will be used.

```
DRAMSET1TMG2.rd2wr = System_R2W + max(0, CDD_RW_R[i]_R[i]) (Same bank group in same physical rank)

RANKTMG1.rd2wr_dr = System_R2W + 5 + max(0, CDD_RW_R[i]_R[j] ) (Different physical ranks)

DRAMSET1TMG2.wr2rd = System_W2R+ max(0, CDD_WR_R[i]_R[i]) (Same bank group in same physical rank)

DRAMSET1TMG9.wr2rd_s = System_W2R + max(0, CDD_WR_R[i]_R[i]) (Different bank group in same physical rank)

RANKTMG1.wr2rd_dr = System_W2R_DR + 5 + max(0, CDD_WR_R[i]_R[j]) (Different physical ranks)

RANKTMG0.diff_rank_wr_gap = MAX((BL/2+ 4 + max(0,CDD_WW_[i]_[j])), (ODTCFG.wr_odt_hold-BL/2))= BL/2+ 4 + max(0,CDD_WW_[i]_[j]

RANKTMG0.diff_rank_rd_gap = MAX((BL/2+ 4 + max(0, CDD_RR_[i]_[j])), (ODTCFG.rd_odt_hold - BL/2)) = BL/2+4+ max(0, CDD_RR_[i]_[j])
```

Note

- The value of `ODTCFG.wr_odt_hold-BL/2` is smaller than `BL/2+4+max(0,CDD_WW_[i]_[j]`. The value of `ODTCFG.rd_odt_hold - BL/2` is smaller than `BL/2+ 4 + max(0, CDD_RR_[i]_[j])`.
- `RANKTMG0.diff_rank_wr_gap` must also +1 if `WR_PREAMBLE=2` and +1 if CRC is enabled.
- `RANKTMG0.diff_rank_rd_gap` must also +1 if `RD_PREAMBLE=2`.

4.2.2 DDR5

Check the mnPmuSramMsgBlock_ddr5.h (i,j=[0..3], indicates the timing group) for all of available CDD fields.

- `CDD_ChA_RR_[i]_[j]`
- `CDD_ChB_RR_[i]_[j]`

- CDD_ChA_WW_[i]_[j]
- CDD_ChB_WW_[i]_[j]
- CDD_ChA_RW_[i]_[j]
- CDD_ChB_RW_[i]_[j]
- CDD_ChA_WR_[i]_[j]
- CDD_ChB_WR_[i]_[j]

For the description of the CDD fields, refer to the tables in the section "Timing Spacing" on page 38.

In DDR5/4 controller:

```
t_rd2wr_dpr=RBL/2=8
```

```
t_wr2rd_dpr=WBL/2=8
```

Controller registers can be adjusted based on above CDD values.

```
RANK_SWITCH_TIMING_CONTROL(0-5).t_wr2wr_gap_rXrY= 5 + MAX[0, CDD_WW_R[i]_R[j]] +
(Twpre -2) + (Twpst-0.5)
```

```
RANK_SWITCH_TIMING_CONTROL (0-5).t_rd2rd_gap_rXrY= 5 + MAX[0, CDD_RR_R[i]_R[j]] +
(Trpre -1) + (Trpst-0.5)
```

```
RANK_SWITCH_TIMING_CONTROL (0-5).t_wr2rd_gap_rXrY= (CWL-CL) + 5 +
max(0,CDD_WR_R[i]_R[j]) + (Twpst -0.5) + (Trpre-2) (different physical ranks)
```

```
RANK_SWITCH_TIMING_CONTROL (0-5).t_rd2wr_gap_rXrY= (CL-CWL) + 5 +
max(0,CDD_RW_R[i]_R[j])+ (Trpst-0.5) + (Twpre -2) (different physical ranks)
```

```
DRAMSET1TMG2.rd2wr = System_R2W + max(0, CDD_RW_R[i]_R[i] ) + (Twpre -2) +
(Trpst-0.5) (same physical rank)
```

```
DRAMSET1TMG9.wr2rd_s = System_W2R + max(0, CDD_WR_R[i]_R[i]) + (Twpst-0.5) + (Trpre
-2) (same physical rank)
```

```
DRAMSET1TMG2.wr2rd = System_W2R + max(0, CDD_WR_R[i]_R[i]) + (Twpst-0.5) + (Trpre
-2) (same physical rank)
```

```
DRAMSET1TMG22.rd2wr_dlr = System_R2W + max(0, CDD_RW_R[i]_R[i] ) + (Twpre -2) +
(Trpst-0.5) (same physical rank)
```

```
DRAMSET1TMG22.wr2rd_dlr = System_W2R + max(0, CDD_WR_R[i]_R[i]) + (Twpst-0.5) +
(Trpre -2) (same physical rank)
```

Note Above equations include preamble and postamble, do not consider them for System_* calculation.

4.3 Read Training Results

The section describes how PhyInit tracks register address and values for the purpose of generating the retention exit sequence.

To read training results, user must implement `dwc_ddrphy_phyinit_userCustom_H_readMsgBlock();` see function details for requirements.

The following is an example pseudo code for implementing this function:

```
if (Train2D)
 {
   _read_2d_message_block_outputs_
 }
 else
 {
   _read_1d_message_block_outputs_
 }
```

Read the Firmware Message Block via APB read commands to the DMEM address to obtain training results.

The default behavior of this function is to print comments related to this process.

A function call of the same name is printed in the output text file. You must implement "_read_1d_message_block_outputs_" to read CDD message block and update the memory controller registers based on the results.

Trained CDD results address can be found from message block header file.

For example, mnPmuSramMsgBlock_lpddr4.h

int8_t CDD_ChA_RR_1_0;	// Byte offset 0x26, CSR Addr 0x54013, Direction=Out
	// This is a signed integer value.
	// Read to read critical delay difference from cs 1 to cs 0 on Channel A
	// See PUB Databook for details on use of CDD values.
int8_t CDD_ChA_RR_0_1;	// Byte offset 0x27, CSR Addr 0x54013, Direction=Out
	// This is a signed integer value.
	// Read to read critical delay difference from cs 0 to cs 1 on Channel A
	// See PUB Databook for details on use of CDD values.

The read data is 16bits, consists of 2 CDD fields, the lower 8 bits is CDD_ChA_RR_1_0, while the higher 8 bits is CDD_ChA_RR_0_1.

Note

The trained CDD are per-Pstate.

The value of CDD for different Pstate can be different, so when training multiple Pstates, you must store the CDD values when each Pstate training is done, and controller needs to be programmed based on related PHY Pstate when frequency change.

The max CDD must be computed over trained ranks only. That is, when it is a single rank configuration, full 1D training, then only CDD_RW_0_0 is of interest. There is no need to extract the max over CDD_RW_i_j for all (i, j) combinations, this may report false maximum.