cādence®

SD/eMMC DLL PHY IP

Internal Name: SD/eMMC PHY

Part Number: IP6185

NDA # _________

Integration Guide

Revision: 1.0

Confidentiality Notice

Trademarks

Restricted Permission

Disclaimer

Restricted Rights

Table of Contents

List of Figures

List of Tables

1. Introduction

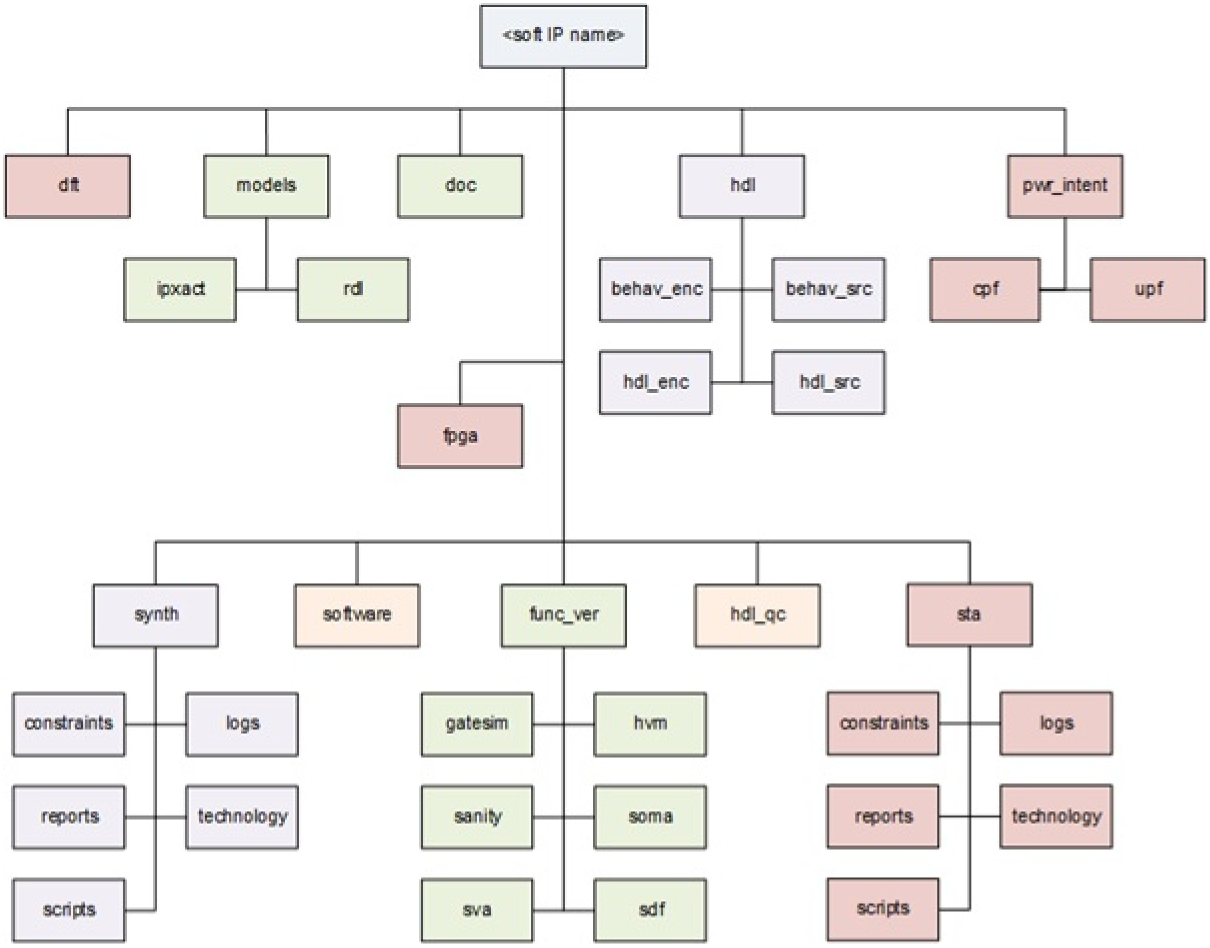
This document describe how the Combo PHY IP should be fully integrated into a SoC design. It cover following topics:

- **Package Content**
- **Verification and Test**
- **Synthesis**
- **STA analysis**

2. Package Content

The Figure 2.1, “Directory structure” illustrates the IP directory structure. The “soft IP name” for this delivery is main directory root.

Figure 2.1. Directory structure



3. Overview of the environment

The simulation environment is a part of the controller package. Check controller integration guide for more details.

4. Synthesis Instructions

4.1. Overview

Note

Before synthesis all ***HIC*** (**H**and **I**nstantiated **C**ell) modules have to be adapted to the target technology. Related files can be found in the IP_name/hdl/behav_src/hic/

A set of RC scripts have been developed to help get started with synthesis and are located within the top level of the delivered package in the IP_name/synth/scripts directory:

- run_synth.csh --> run the RC tool sourcing the rc/genus_synth.tcl script.
- setup_project.csh --> set up environment variables for PHY core level.
- project.tcl --> set up environment variables for PHY core level.
- genus/genus_synth.tcl --> The main command script that is read into RC..
- genus/dft_setup.tcl --> Sourced by genus_synth.tcl, configures RC for full scan insertion.
- genus/dft_insert_scan.tcl -> Sourced by genus_synth.tcl, inserts scan chains into synthesized netlist.
- genus/lec*.do --> LEC scripts
- genus/mmmc.tcl

Constraints files are provided in the synth/constraints directory.

4.2. Flow

Here are the various stages executed by genus_synth.tcl to perform RTL synthesis:

1. project*.tcl is read by genus_synth.tcl to set various project-specific TCL variables, which are then used by the RC script and other point scripts.

2. Setup stage

- Define global attributes.
- Read .lib library files as determined by value of TECHNOLOGY variable.
- Select VT library
- Define "don't use" cells.
- Configure for PLE mode, ensures accurate correlation between RC and EDI results.

3. Read in the RTL

4. Elaborate the design. Script supports extraction of source file list and parameters from .f file.

5. Source dft_setup.tcl

- Define test signals (mode signals, shift enables, resets).

- Define test clocks.
- Define test signal for clock gating.
- Check that all flops pass DFT rule checks.

6. Create ATPG clock constraints file:

"$DUT_PATH/cfg/${CONFIG}/dft/constraints/clock_constraints_${CONFIG}.txt" The information in the file describes the clock to clock relationships and timing relationships that must be enforced between them. Refer to “Clock Constraints File” in the Automatic Test Pattern Generation User Guide for additional information. Example format:

clk_a {posedge,5ns} {100Mhz}

clk_b {posedge,1250ps} {400Mhz};

7. Perform lint check of SDC's

8. Synthesize to generic.

9. Synthesize to mapped.

10. Source dft_insert_scan.tcl

- Ensure mapped to scan flops.
- Preserve pre-existing shift registers.
- Define scan chains.
- Connect scan chains.
- Write out netlist for Encounter Test

11. Incremental optimization.

12. Write reports.

4.3. Running synthesis

To run synthesis with Cadence RTL Compiler tool, change the working directory to: ./synth/scripts and run shell scripts run_synth.csh.

5. Static Timing Analysis

This chapter describes the standard set of scripts static timing analysis for Cadence Design IP DLL PHY. In addition to that a detailed description on timing the data paths, specialized reports being used and STA runs are also discussed.

The following topics are covered in this chapter:

- Script Overview
- Static Timing Analysis Scripts
- Special constraints
- Analysis
- Specialized Reports
- Synthesis Methodology
- Special Notes

5.1. Script Overview

Cadence has developed a standard set of scripts that may be used by the Encounter Timing System Tool.

5.1.1. Directory Structure

There are five directories in the ./sta/ directory:

- **constraints/** - Holds IP timing constraints in SDC format with user_setup file.
- **logs/** - Holds logs of the execution.
- **reports/** - Holds output report files from the execution.
- **technology/** - Holds library information and setup files. These files are shared among any modules being tested.
- **scripts/** - Holds the setup and execution scripts for the Cadence Design IP DLL PHY.
  - **tool/** - Holds tool-specific information and control files. These files are shared among any modules being tested.
  - **analysis/** - Holds analysis scripts.
  - **netlists/** - Holds the netlists file from the synthesis/PnR execution.

5.1.2. User Setup Preparation

Cadence provides user setup files that allow the user to define parameters for static timing analysis. These provide a singular location for all variables that affect all static timing scripts. The user setup file should be modified prior to executing any scripts. These files require user specific information and must be changed. The file is located in **sta/constraints/user_setup**.

The user specific information that are required is the directory structure, clock skew, memory pin information and some information from the user’s system timing budget. The information needed for the **sta/constraints/user_setup** file come from several places. Some variables convey technology library information to the scripts, such as names of the data and clock inputs to a flip-flop. In addition, some variables change the hierarchical scope of the analysis. The PNR_STAGE variable is used to define the source of the netlist. If the netlist to analysis is sourced from synthesis execution the PNR_STAGE should be set to zero and if netlist is from post route the PNR_STAGE should be set high.

Most of the variables set a maximum latency and maximum skew criteria for critical paths. The values needed are determined from system level timing requirements and understanding of the paths. These variables are described in detail throughout this chapter.

Loading of backannotated timing information (such as SDF) or parasitics (SPEF) is not provided in the Cadence STA. The user need to modify the scripts to load this type of information. Typically, these types of files are loaded after the design has been read in. Any special settings such as those for SI analysis or running at different operating conditions need to be handled by the user as well.

5.1.3. Execution

Timing Analysis is executed by running the **sta/scripts/run_phy.ets.sh** script. The Cadence Design IP DLL PHY is independently synthesized and timed, and therefore the scripts may be run before or after any PHY timing optimizations.

Note

The timing analysis scripts require some input from synthesis, and therefore synthesis scripts must be run on a module prior to checking timing.

The **sta/scripts/run_phy.ets.sh** script execute the *cdns_combo_dll_phy.ets* script.

5.2. Static Timing Analysis Scripts

The following tables list the input and output files associated with static timing for the Cadence Design IP DLL PHY.

Table 5.1. Timing Scripts (in the Distribution sta/scripts/ Directory)

File name	Description
module.file_list.ets	Holds the file list used by this module for static timing analysis.
module.ets	Top-level script for running the static timing analysis for this module. This script sources the technology setup, user setup, netlists, generic constraints, tool constraints and report generation files.
run_phy.ets.sh	Script that executes the timing analysis for all modules and compiles the log files.

Table 5.2. Timing Scripts (in the Distribution sta/technology Directory)

File name	Description
generic_io.lib	Library that contains the I/O cells used in this configuration. This PHY uses the Cadence I/O pad models.
setup_io.ets	Setup file for the I/O library being used.
setup_lib.ets	Setup file for the technology library being used.

Table 5.3. Timing Scripts (in the Distribution sta/scripts/tool Directory)

File name	Description
message_control.ets	Specifies warning messages that are irrelevant to the timing analysis operation.
segment_timing.tcl	Script used to report timing from a specific point to a specific point in the design.
sta_analysis_tools.tcl	Script used to report skew between two input lists, or the maximum skew between the inputs of one list.
sta_skew_check_dll_select.tcl	Script to find the delay and skew to the select lines of a digital DLL.

File name	Description
sta_skew_check_slice.tcl	Procedure used when timing the data slice and the Cadence Design IP DLL PHY. This procedure specifically views max delays and skew between specified groups.
sta_suhld_check_slice.tcl	Script to find the setup and hold values of the entry flip-flops that grab the data from external memory parts.

Table 5.4. Analysis Files (in the Distribution sta/scripts/analysis Directory)

File name	Description
module.analyze.ets	Generates the timing and other reports for this module. This calls procedures sourced from the tool/directory and is used to analyze the delay of each cell in the delay line by looking at the skew of data/clock/address/control, etc.
module.tool_con.ets	Tool-specific constraints definition file for this module. This identifies “don’t touch” directives and name change rules.

Table 5.5. Output Files (in the Distribution sta/logs/ Directory)

File name	Description
cdns_combo_dll_phy. master_phase_1_delay.log	Shows the delay of each element in the master phase 1 delay line.
cdns_combo_dll_phy. master_phase_2_delay.log	Shows the delay of each element in the master phase 2 delay line.
cdns_combo_dll_phy. read_delay.log	Shows the delay of each element in the read delay line.
cdns_combo_dll_phy. write_delay.log	Shows the delay of each element in the write delay line.
cdns_combo_dll_phy. element_delay.log	Shows the delay of each delay element.
cdns_combo_dll_phy.ets.log	Report file for static timing analysis for this module. This file contains information on errors that occurred in the STA run.

Table 5.6. Output Files (in the Distribution sta/reports/ Directory)

File name	Description
cdns_combo_dll_phy. master_phase_1_delay.rpt	Analysis of the delay elements of the master phase 1 delay line against the provided window.
cdns_combo_dll_phy. master_phase_2_delay.rpt	Analysis of the delay elements of the master phase 2 delay line against the provided window.
cdns_combo_dll_phy. read_delay.rpt	Analysis of the delay elements of the read delay line against the provided window.
cdns_combo_dll_phy.reports.	Output from the check timing function; reports on unconstrained ports.

File name	Description
check_constraints	
cdns_combo_dll_phy. write_delay.rpt	Analysis of the delay elements of the write delay line against the provided window.
cdns_combo_dll_phy.skew.addrcntrl.rpt	Skew check across the address and control bits.
cdns_combo_dll_phy.skew.slice_X.rpt	Slice X skew report. There is a unique file for each slice of the Cadence Design IP DLL PHY.
cdns_combo_dll_phy.suhld.slice_X.rpt	Shows the max setup and skews for the slice X entry flip-flop's setup times and the max hold and skews for the slice X entry flip-flop's hold times. This setup/hold report only applies to the flip-flop; it is not a setup/hold check from the port to this flip-flop. This report is similar to looking in the SDF file for the setup/hold values for the entry flip-flops and it can be used to calculate the read data setup/hold from the port to these flip-flops. This information is used to calculate the read data valid window.
cdns_combo_dll_phy.skew.slice.rpt	Reports on the max delay and skew of defined groups for the data slice. Skew groups are defined as follows: • wr_h_*/wr_l_* flip-flop to dataout • clk_wr to wr_h_/wr_l_/output mux • shift* flip-flop that generates OE to OE’s output • clk_phy to the DLL • read_data to the entry* flip-flop • DLL to the entry* flip-flops • read_dqs insertion delay • dqs to clk_phy domain crossing • gate timing check • read DQ/DQS path balancing
cdns_combo_dll_phy. suhld.slice.rpt	Shows the max setup and skews for the entry flip-flop's setup times and the max hold and skews for the entry flip-flop's hold times. This setup/hold report only applies to the flip-flop; it is not a setup/hold check from the port to this flip-flop. This report is similar to looking in the SDF file for the setup/hold values for the entry flip-flops and it can be used to calculate the read data setup/hold from the port to these flip-flops. This information is used to calculate the read data valid window.
module.check_timing.ets	check_timing report.
module.clock.ets	Reports that describe the clock.
module.timing.max.ets	Full worst-case timing report.

File name	Description
module.timing.max_summary.ets	Summarized worst-case timing report for the modules of the PHY. Contains only start point, end point and slack information.
module.timing.min.ets	Full best-case timing report.
module.timing.min_summary.ets	Summarized best-case timing report for the modules of the PHY. Contains only start point, end point and slack information.

The files listed in the table Tab-Delimited Output Files (in the Distribution sta/reports/ Directory) are tab-delimited to allow easy insertion into any spreadsheet program such as Microsoft Excel. To read these files into an Excel spreadsheet, follow the steps below:

1. Open Microsoft Excel.
2. Create a new worksheet, or select a cell in an existing worksheet.
3. From the Data Menu, select **Get External Data** and then **Import Text File**.
4. Locate the desired report file and then click **Import**.
5. Once the file has been imported, click **Finish** and then **OK**.

Since data may be used on a per-slice basis or across all slices, the data is presented in both forms. Therefore, there is some duplication across the report files. To see the actual paths that were used in the calculation of the **summary** reports, please refer to the *cdns_combo_dll_phy.skew.slice_X.rpt* and *cdns_combo_dll_phy.skew.slice_X.rpt* reports.

Table 5.7. Tab-Delimited Output Files (in the Distribution sta/reports/ Directory)

File name	Description
cdns_combo_dll_phy.skew_summary.addrcntrl.rpt.ets	Tab-delimited file reporting a summary of the delays and skew calculations on the address and control bus.
cdns_combo_dll_phy.skew_summary.slice_X.rpt.ets	Tab-delimited file reporting a summary of the delay and skew calculation for data slice X at the PHY level. There is a unique file for each slice of the Cadence Design IP DLL PHY.
cdns_combo_dll_phy.skew_summary_table.rpt.ets	Tab-delimited file reporting a summary of the delay and skew calculation for each data slice at the PHY level - this one file contains information for all data slices.
cdns_combo_dll_phy.suhld_summary.slice_X.rpt.ets	Tab-delimited file reporting the max setup and skews for the entry flip-flop’s setup times and the max hold and skews for the entry flip-flop's hold times. There is a unique file for each slice of the Cadence Design IP DLL PHY, timed at the PHY level.
cdns_combo_dll_phy. suhld_summary.rpt.ets	Tab-delimited file reporting the max setup and skews for the entry flip-flop's setup times and the max hold and skews for the entry flip-flop's hold times. This setup/hold report only applies to the flip-flop; it is not a setup/hold check from the port to this flip-flop. This report is similar to looking in the SDF file for the setup/hold values for the entry flip-flops and it can be used to calculate the read data setup/hold from the port to these flip-flops. This information is used to calculate the read data valid window.

5.3. Special constraints

This chapter describes the reset methodology being employed by the synthesis scripts, clock relationships in the Cadence Design IP DLL PHY, points to remember while building clock trees and criticial timing boundaries.

5.3.1. Reset Methodology

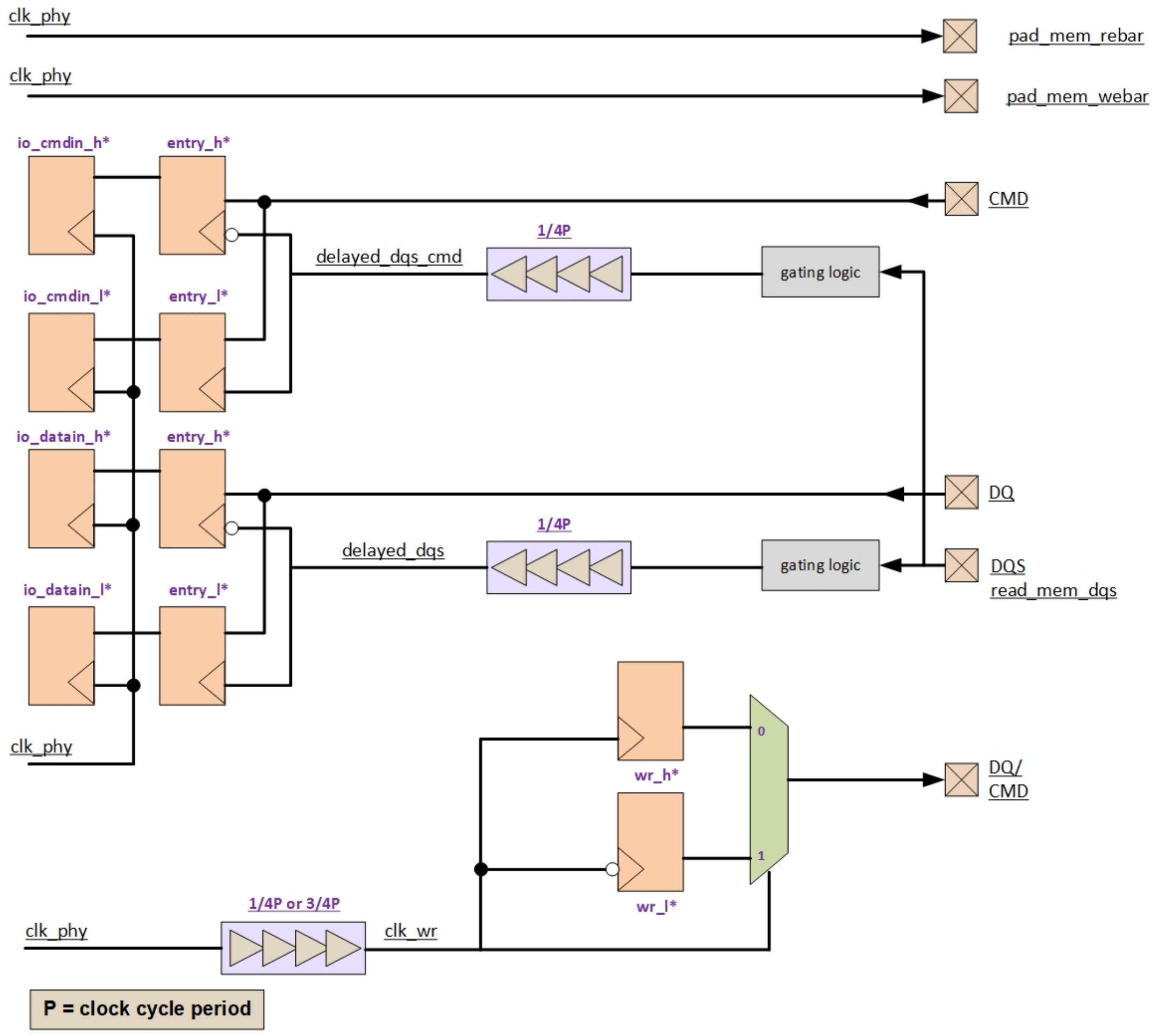
It is assumed that the reset signals are built by a clock tree insertion tool. With this assumption, the synthesis scripts treat the resets as ideal nets, and the STA scripts treat them as false paths.

The *sta/constraints/user_setup* file contains the variable SET_RESETS_FALSE. This variable defaults to a “1” which declares the resets as false paths. Reset may be asserted asynchronously but must be de-asserted synchronously. To run the scripts without the resets set to false paths, change this variable to “0” before executing the script. It is recommended that timing of the resets be checked after the reset tree has been built.

5.3.2. Clock Relationships

There are several clocks in the Cadence Design IP DLL. Many of these clocks do not interface, or they are asynchronous relative to each other. The following diagram shows a block diagram of the clocks in the PHY.

Figure 5.1. Clocks in the Cadence Design IP DLL PHY



*io_datain** and *entry** flops are part of the *read_datablk_fifo* module.

- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_*/hic_dll_entry_flop
- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/dfi_read_cmdblk/read_cmdblk_fifo/dll_entry_flop_*/hic_dll_entry_flop
- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/io_datain_*
- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/dfi_read_cmdblk/read_cmdblk_fifo/io_datain_*

*wr_l** and *wr_h** flops are part of the *io_data*cell* module.

- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/io_data*cell_*/wr_l*
- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/io_data*cell_*/wr_h*
- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/io_cmdcell/wr_l*
- Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/io_cmdcell/wr_h*

The ideal relationships depicted in the above block diagram are detailed in the following table.

Table 5.8. Functional Clock Relationships

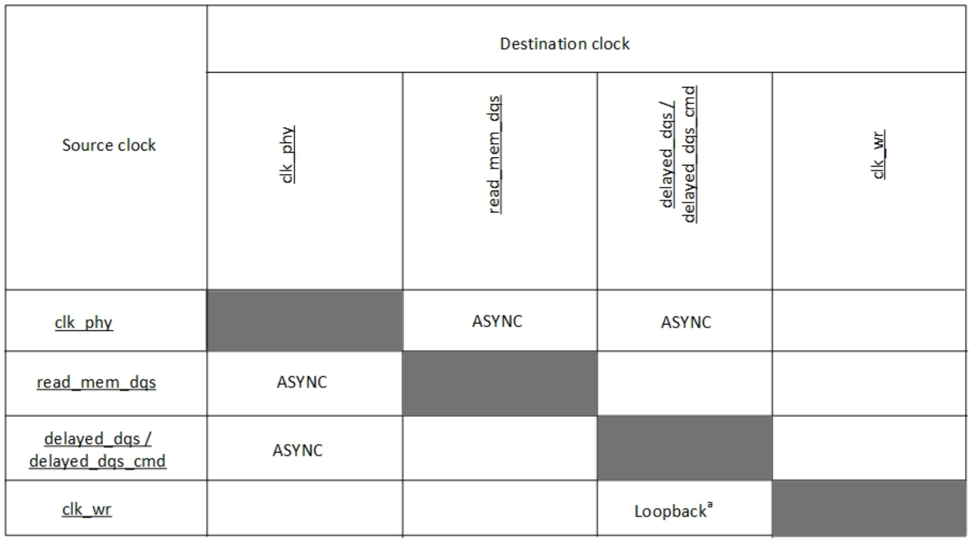
Source	Destination	Nominal shift
clk_phy	clk_wr	1/4 or 3/4 clock shift
read_mem_dqs	delayed_dqs	1/4 clock shift
read_mem_dqs	clk_phy	pure asynchronous boundary
delayed_dqs	clk_phy	pure asynchronous boundary

Note

Details about the gating logic on the read DQS path are shown in "user guide".

Many of these clocks do not communicate, or only communicate in a test mode. The table below shows the paths that are defined with the *set_clock_groups -asynchronous* constraint. This constraint eliminates timing checks between the clock domain pair.

Figure 5.2. Clock crossing boundaries



Source clock	Destination clock: clk_phy	Destination clock: read_mem_dqs	Destination clock: delayed_dqs / delayed_dqs_cmd	Destination clock: clk_wr
clk_phy	▨	ASYNC	ASYNC	
read_mem_dqs	ASYNC	▨		
delayed_dqs / delayed_dqs_cmd	ASYNC		▨	
clk_wr			Loopback[a]	▨

lpbk_dqs – is asynchronous to all other clocks

In the above table, *** a** refers to the path that only occurs during loop back, and therefore the timing is less critical on these paths. STA still checks this timing.

5.3.3. Clock tree

The following points should be considered when building the clock trees:

- When building the clk_phy clock tree, it is assumed that the DLL inputs are synchronous points with all other flip-flops on this clock tree.
- When building the clk_wr clock tree, it is assumed that the output data multiplexer select is a synchronous point on this clock tree with all other flip-flops. In addition, it is assumed that this clock tree will be minimized.

Note

DLL refers to the : Inst_flash_databahn_dll_phy/dll_phy_slice_core/data_slice_0/dll/

5.3.4. Critical Timing Boundaries

This section describes the critical timing boundaries for Cadence Design IP DLL PHY.

5.3.4.1. Timing to the wr_l and wr_h Flip-Flops

Write data is passed over the DFI bus synchronous to the core clock. This data is then passed to the flip-flops *wr_l* and *wr_h* which drive the data out onto the memory data bus. In this transition, the data is passed from the core clock domain to the *clk_wr* clock domain. This creates a tight timing path that is a 1/4-cycle clock path.

5.3.4.2. Output Enable

The control signal dfi_wrdata_en identifies that write data is being transmitted across the DFI bus. Once data is received, it must be driven out to memory. Therefore, the dfi_wrdata_en signal is fed through a delay chain to create a 1/2-clock programmable resolution signal which is used to enable/disable the output. This delay chain is created with pipeline flip-flops that are clocked on alternating clock edges to create 1/2-clock domain crossings.

The memory chain is: posedge clk_phy –> negedge clk_phy –> posedge clk_phy –> negedge clk_phy, etc.

5.3.4.3. Output Enable for data mask

The control signal xspi_dfi_wrdata_mask_en identifies that write data mask is being transmitted across the DFI bus. Once data mask is received, it must be driven out to memory. Therefore, the xspi_dfi_wrdata_mask_en signal is fed through a delay chain to create a 1/2-clock programmable resolution signal which is used to enable/disable the output. This delay chain is created with pipeline flip-flops that are clocked on alternating clock edges to create 1/2-clock domain crossings.

Note

There is a mux at the toplevel that drives the output enable (dqs_oepad). For byte masking, this is treated as a normal data out so the dqs_oe in this case is treated the same as dq_oe. The two output enables feed into the mux and get selected according to whether we are in loopback mode or normal functional mode.

The memory chain is: posedge clk_phy –> negedge clk_phy –> posedge clk_phy –> negedge clk_phy, etc.

5.3.4.4. Tsel Timing

The control signal *dfi_rddata_en* identifies that read data will soon be transmitted across the DFI bus. The *dfi_rddata_en* signal is fed through a delay chain to create a 1/2-clock programmable resolution signal to enable/disable the termination at the memory controller boundary. This delay chain is created with pipeline flip-flops that are clocked on alternating clock edges to create 1/2-clock domain crossings.

The memory chain is: posedge clk_phy –> negedge clk_phy –> posedge clk_phy –> negedge clk_phy, etc.

5.3.4.5. Timing from the “entry” Flip-Flops to the “io_datain” Flip-Flops

Since the arrival of the read DQS signal is dependent on board trace lengths and the memory timing, this signal is considered asynchronous to the core clock. Therefore, this timing path is not checked during the normal synchronous timing. To ensure that the timing boundary between the DQS and the core clock do not add considerable latency to the read timing, the timing scripts do calculate the delay from the entry flip-flops that capture the data off the memory bus to the synchronous flip-flops in the core clock domain (io_datain_h/io_datain_l). This delay is compared to a max delay value specified in the **sta/ constraints/ user_setup** file.

5.3.4.6. Loopback Mode

Loopback mode can be used for testing the read and write data paths. For more details on this feature, refer to "user guide". While it is required that timing be met in order for loopback to work, this timing is not checked with normal synchronous timing for the following reasons:

- DLL settings for the write and read paths are different for functional mode and loopback mode.
- Trying to close timing on the loopback mode while closing timing on the read and write paths can cause conflicting changes that will lengthen the time to close timing.

It is recommended that the timing of the read path and write path be closed first. Once this timing has been closed, the loopback timing should be met. To ensure that this occurs, there are custom timing checks for loopback mode similar to the ones for the read and write paths. These are explained in detail in Section 5.4.1.4, “*Loopback Mode*”.

5.3.4.7. Read and Write Data Paths

For the read and write data paths, timing checks that verify that setup and hold times are insufficient for a device bus that supports DDR data transfer. This type of check will not maximize the data capture window, which will require the clock path to be equal to or slightly longer that the data path. There are custom checks that view the DQS insertion delay relative to the data. The exact checks are described in detail in Write path, and Section 5.4.1.2, “*Read Path*”.

5.4. Analysis

It is assumed that PHY timing is accomplished by closing timing on the slice level, then closing timing on the PHY-level. This is beneficial because critical timing is on the slice-level, and once this timing is closed, the slice can be replicated. The slice timing need not be closed again at the PHY level. Since many of the timing paths are dependent on each other, following this order for timing closure can save overall time and effort:

- Balance the write data path
- Balance the read data path
- Balance the loopback timing
- Balance the memory clock timing

After passing flip-flop to flip-flop timing, the user should first work on balancing the DQS to each of the DQ lines. It is important to balance the DQS and DQ lines to ensure that the DLL can be used to sweep the entire data valid window, and then set the delay for maximum margin.

On the read side, the DQS path is typically much longer than the DQ bit paths. This is because the DQS passes through gating logic and a DLL unlike the DQ paths which go straight to capture flip-flops. To overcome this difference, each DQ bit path must be buffered. The *Rdiff* STA* check (refer to Section 5.4.1.2, "*Read Path*") indicates whether the DQS and DQ lines are balanced. A similar problem exists on the write path and the *Wdiff* STA* check (refer to Write path) indicates the quality of the buffering on this path. The balancing of the DQS to the DQ bit lines is more important than the actual delays, as long as the actual delays do not become prohibitively long on the write path such that the DQS or DQ does not turn in time for a write.

Many of the variables in the **sta/constraints/user_setup** file should be determined by the system timing budget and analysis. However, Cadence offers the following general recommendations:

- **MAX_WRITEDATA_PAD_DELAY** should be set to match the maximum pad delays. Similar values should be defined on the read side with **MAX_READDATA_PAD_DELAY** and **MAX_READDQS_PAD_DELAY**. Since pad timing is not generally fixed from the I/O provider, it can not be optimized; however, the STA reports should show the pad timing at the PHY level.
- **MAX_READDATA_SKEW** and **MAX_WRITEDATA_SKEW** reduce the data valid windows as their values are increased, so it is very important to reduce these as much as possible.
- Where minimum delays are specified, the user should define minimum values with the same number as the maximum delay. This results in the backend tool slowing down faster paths to make them match the longer paths which reduces skew. However, when performing hold time analysis and fixes, these minimum delays should be set to **0** so that the unnecessary buffers are not introduced.
- **CLK_SOURCE_LAT** is used to model external clock network delay to the PHY. The actual delay leading to the PHY is not important. Also, note that the clock signal going to external memory will be sourced from the PHY. That clock will have JEDEC requirements for tests such as duty cycle and jitter which are not checked by the Cadence scripts.

  Note

  Please ensure that the clock going to external memory meets the requirements for the memory parts being used.

- Skew values in general should be set to less than or equal to 5% of the clock period and delays should be set to less than or equal to 1/4 cycle.
- **MAX_LPBK_DQS_SKEW** specifies the maximum amount of skew on the internal loopback data path and should be set accordingly.

- Cadence's STA do not directly support running the STA scripts at a level of hierarchy above the cdns_combo_dll_phy module. However, attempting this by setting the **DATABAHN_PHY_HIERARCHY** to a non-empty string will provide a starting point for changes that the user will need to make.

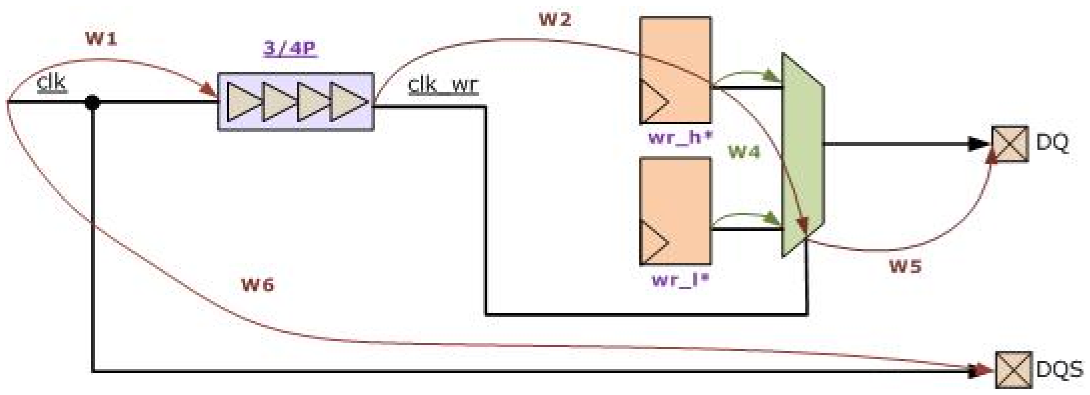
5.4.1. Timing the data paths

The read data path, write data path and loopback mode path require timing analysis. The primary concept for closing this timing is that the data path timing must be equal to the "clock" or DQS path time. It is assumed that the DLL's intrinsic delay is negligible or zero. Also, it is assumed that the DQS and data arrive simultaneously for reads, or clock (webar) and data arrive simultaneously for writes with the DLLs set to having no delay. If this is accurate, the DLL can be used to shift the DQS to the center of the data eye.

5.4.1.1. Write Path

The term *Wdata* is used for the delay from the source clock to the output data pin. The following diagram shows the write path in PHY.

Figure 5.3. Write Path in the Cadence Design IP DLL PHY



The relationships depicted in the above block diagram are detailed in the following table. This table includes information on each of the timing arcs as well as the variable used in the **sta_skew_check_slice.tcl** script. The STA pass/fail variables are defined in the **sta/constraints/user_setup** file.

Table 5.9. Write Data Path Relationships

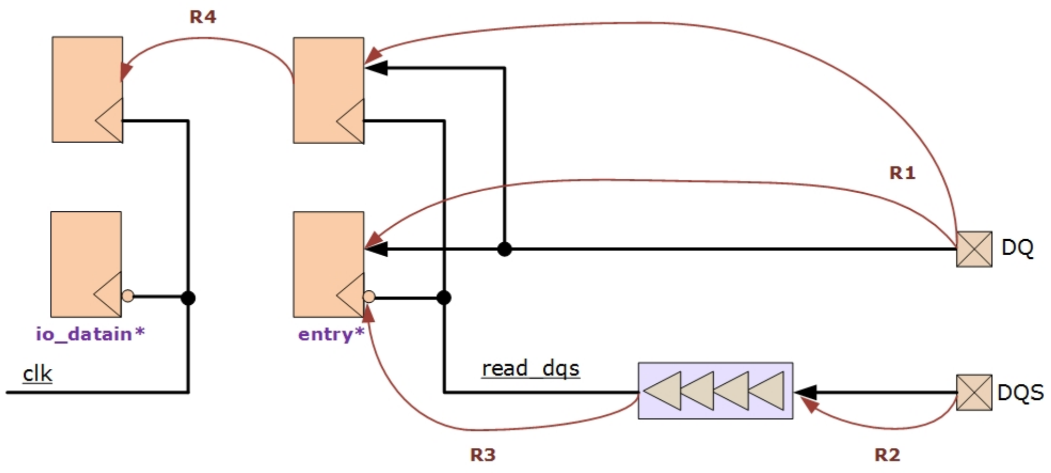
Tag	STA Variable	STA Pass/Fail Variable	Description
W1	actual_delay_clk_wr_ins_delay	MAX_CLKWRINS_DELAY	Delay of the reference clock from the clk_phy input to the clk_wr DLL
W1	actual_skew_clk_wr_ins_delay	MAX_CLKWRINS_SKEW	
W2	actual_delay_wr_mux_clock	MAX_WRITEDQS_DELAY	Delay of the clk_wr clock tree from the DLL to the multiplexer select of the output data multiplexer
W2	actual_skew_wr_mux_clock_posedge	MAX_WRITEDQS_SKEW	Skew of the clk_wr clock tree from the DLL to the multiplexer select of the output data multiplexer
W2	actual_skew_wr_mux_clock_negedge	MAX_WRITEDQS_SKEW	
W4	actual_delay_wr_reg_data_r_delay	PHY_QUARTER	Delay of the write data from the output data flip-flops to the input of the data multiplexer

Tag	STA Variable	STA Pass/Fail Variable	Description
W4	actual_delay_wr_reg_data_f_dela y	PHY_QUARTER	
W4	actual_delay_wr_reg_data_delay	PHY_QUARTER	
W5	actual_skew_wr_mux_data_rise	MAX_WRITEDATA_DELAY	Delay of the write data from the output data multiplexer select to the output port of **cdns_combo_dll_phy_data_slice.v** (at the slice level) or to the pad (at the PHY level)
W5	actual_skew_wr_mux_data_fall	MAX_WRITEDATA_SKEW	Skew of the write data from the output data multiplexer select to the output port of **cdns_combo_dll_phy_data_slice.v** (at the slice level) or to the pad (at the PHY level)
W5	actual_skew_wr_mux_data	MAX_WRITEDATA_SKEW	
W5	actual_delay_wr_mux_data	MAX_WRITEDATA_SKEW	

5.4.1.2. Read Path

The term *Rdata* is used for the read data pin to the capture flip-flop, and *Rdqs* as the delay from the read DQS pin to the clock pin of the capture flip-flop. The following figure shows the read path.

Figure 5.4. Read path



The relationships depicted in the above figure are detailed in the following table. This table includes information on each of the timing arcs as well as the variable used in the **sta_skew_check_slice.tcl** script. The STA pass/fail variables are defined in the **sta/constraints/user_setup** file.

Table 5.10. Read Data Path Relationships

Tag	STA variable	STA Pass/Fail variable	Description
R1	actual_delay_read_reg_data	MAX_READDATA_DELAY	Longest read data path from the module port (slice level) or the pad (PHY level) to all the entry flip-flops

Tag	STA variable	STA Pass/Fail variable	Description
R1	actual_skew_read_reg_data_rise	MAX_READDATA_SKEW	Skew of the read data from the module port (slice level) or the pad (PHY level) to the posedge entry flip-flops
R1	actual_skew_read_reg_data_fall	MAX_READDATA_SKEW	Skew of the read data from the module port (slice level) or the pad (PHY level) to the negedge entry flip-flops
R1	actual_skew_read_reg_data	MAX_READDATA_SKEW	Skew of the read data from the module port (slice level) or the pad (PHY level) to all the entry flip-flops
R2	actual_skew_dqs_ins_rise_delay	MAX_READQSINS_SKEW	Skew of the rising edge read DQS from the module port (slice) or pad (PHY level) to the read_dqs DLL.
R2	actual_skew_dqs_ins_fall_delay	MAX_READQSINS_SKEW	Skew of the falling edge read DQS from the module port (slice) or pad (PHY level) to the read_dqs DLL.
R2	actual_skew_dqs_ins_skew	MAX_READQSINS_SKEW	Skew of the read DQS from the module port (slice) or pad (PHY level) to the read_dqs DLL.
R2	actual_delay_dqs_ins_delay	MAX_READQSINS_DELAY	Delay of the read DQS from the module port (slice) or pad (PHY level) to the read_dqs DLL.
R3	See more description in Section 5.4.1.3, “*Read DQS*”.	MAX_READDQS_SKEW	Skew of the read DQS from the read_dqs DLL to the posedge entry flip-flops
R4	actual_delay_async_boundary	MAX_ASYNC_DELAY	Delay across the asynchronous boundary from the entry* flipflops to the io_datain* flop-flops

From the definitions in the above table, the following equations may be derived:

```
Rdata = R1
Rdqs_rise = R2_rise + R3_rise
Rdqs_fall = R2_fall + R3_rise
Rdiff_rise = Rdqs_rise - Rdata
Rdiff_fall = Rdqs_fall - Rdata
0 # (-1*Rdiff) < $RDATADQS_THRESHOLD
```

To close timing, the user must define these delays such that Rdiff_rise # 0 and Rdiff_fall # 0.

Note

The equation for Rdqs_fall uses “R3_rise” (instead of “R3_fall”) because an inverter is used on that path before reaching the timing segment for R3. The read DQS signal goes through separate DLLs to independently delay the rising and falling edges.

RDATADQS_THRESHOLD is a user-defined variable that is ideally zero. However, since a zero value is not possible, this value should be minimized. System level timing determines the maximum allowable value for RDATADQS_THRESHOLD.

The delay for the DQS must be less than or equal to the Rdata delay, or the DLL will not be able to walk the DQS across the entire THEORETICAL data eye. At the same time, the Rdata delay must not be too large or the DLL will not be able to center the DQS in the data eye.

Note

When interpreting these results, make sure there is an intrinsic delay from the DLL.

5.4.1.3. Read DQS

The script **sta_skew_check_slice.tcl** shows how the Read DQS delay skews are calculated. The entry_l and entry_h flip-flops are created with an “always@(posedge …)” reference. This should produce positive edge flip-flops, even though the entry_h flip-flop is receiving an inverted clock to effectively create a negative edge clock. The script **sta_skew_check_slice.tcl** is built using the procedure **segment_rise_to_from_timing** in **sta/scripts/tool/segment_timing.tcl** to determine the timing of a rising edge at all the entry flip-flops. If the entry_h flip-flops are built with negative edges, the script **sta_skew_check_slice.tcl** must be updated to capture the timing of a falling edge at the entry flip-flops.

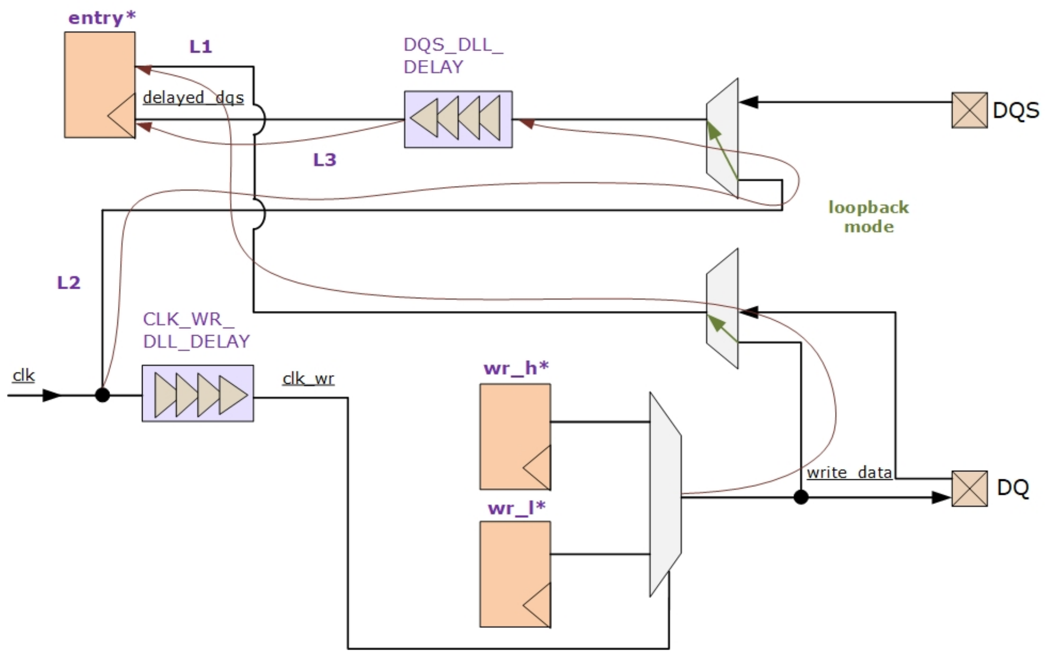
5.4.1.4. Loopback Mode

Once the read and write paths are timing closed, the loopback path should intrinsically be timing closed since it is a combination of the read and write paths. The intent of the loopback timing check is not to change the read/write timing paths that are closed. Rather, the intent is to report the loopback path delay and check the combined skew of the data bits in the read and write path.

The term **LPBKdata** is used for the loopback data pin to the capture flip-flop, and **LPBKdqs** as the delay from the read DQS pin to the clock pin of the capture flip-flop.

The following figure shows the loopback path. The clk_phy signal is looped back and multiplexed onto the read DQS clock tree before the DLL and is used to capture the data on the read side.

Figure 5.5. Loopback Path in the Cadence Design IP DLL PHY



The relationships depicted in the above block diagram are detailed in the following table. This table includes information on each of the timing arcs as well as the variable used in the sta_skew_check_slice.tcl script. The STA variables are defined in the **sta/constraints/user_setup** file.

Table 5.11. Loopback Path Relationships

Tag	STA variable	STA Pass/Fail variable	Description
L1	actual_delay_lpbk_data	MAX_LPBK_DATA_DELAY	Delay from the write_data multiplexer output to the input of the entry flip-flop.
L2	actual_skew_lpbk_dqs_ins_rise	MAX_LPBK_DQS_SKEW	Delay from the clk_phy signal to the input of the DQS_DLL_DELAY chain
L2	actual_skew_lpbk_dqs_ins_fall	MAX_LPBK_DQS_SKEW	
L2	actual_skew_lpbk_dqs	MAX_LPBK_DQS_SKEW	
L2	actual_delay_lpbk_dqs	MAX_LPBK_DQS_DELAY	
L3	R3 from the read path		Delay from the DQS_DLL_DELAY chain output to the clock input of the entry flip-flop.

From the definitions in the above table, the following equations may be derived:

```
LPBKdata = L1
LPBKdqs = L2 + L3
LPBKdiff = LPBKdqs - LPBKdata
```

Cadence recommends that the loopback data skew (MAX_LPBK_DATA_SKEW) be kept less than one-quarter cycle. The resulting data valid window will be 1/4 cycle wide or more and the DQS strobe will be able to reliably capture the loopback data.

There is no specific requirement for LPBKdiff but balancing it by reducing its value to near **0** will make it easier to find loopback settings that pass. Any buffering added to make LPBKdiff close to 0 must not cause Rdiff or Wdiff to fail. The buffers should be added only on the portion of the loopback path between the write and read datapath.

5.4.1.5. Output Enable / Termination Select

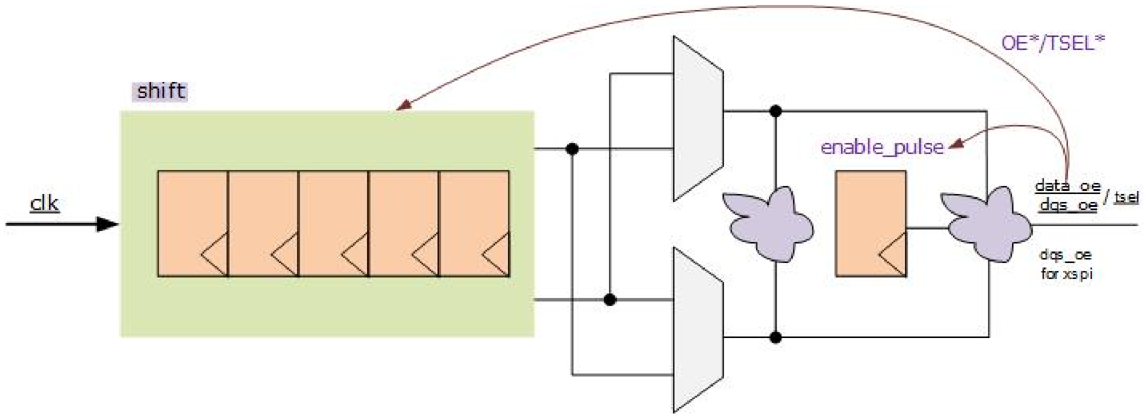
The output enable logic and the termination select logic are built in the same way. There is a control signal that enters a shift chain, and the output of each stage of the shift chain is fed to two multiplexers. The first multiplexer turns the OE/TSEL on and the second turns the OE/TSEL off. The additional flip-flop shown in Figure 5.6, “Output Enable and Termination in the Cadence Design IP DLL PHY” is a holding register used to maintain the OE/TSEL on. The multiplexer selects are programmable values which allow for adjustment in the turn on/off time.

Effectively, the turn_on, turn_off and enable_pulse signals are OR’d together. The enable_pulse term is used to bridge the on and off signals if they do not overlap. If the PHY is running with a memory burst length of 4 or greater, the on and off signals will be at least 2 clocks long. This knowledge allows the user to set a multicycle path constraint from the shift flip-flop to the enable_pulse flip-flop. However, if the memory burst length is 2, the on/off signals will be a single clock pulse, so this constraint can not be applied.

The purpose of the OE/TSEL check is to ensure that the timing from the shift flip-flops to the enable_pulse flip-flop is “tight”. An exact definition of “tight” is not critical - and good engineering judgment is needed to ensure the delays in the path are reasonable and that the buffering is not excessive. Since the multiplexer selects are programmable, any reasonable delay from the shift/enable_pulse flip-flops to the pad OE/TSEL may be set. The exact setting will be determined in the lab.

Because all of the flip-flops of the shift chain are on clock trees, it is assumed that the clock tree skew is well managed. Therefore, the skew check of this circuit checks the functional paths. There are multiple sources and in the case of DQ, multiple destinations, it is desired that the skew be minimized. The shift chain allows 1/2-clock programming resolution, and so the skew should be less than 1/2-clock.

Figure 5.6. Output Enable and Termination in the Cadence Design IP DLL PHY



The relationships depicted in the above figure are detailed in the following table. This table includes information on each of the timing arcs as well as the variable used in the **sta_skew_check_slice.tcl script**. The STA pass/fail variables are defined in the **sta/constraints/user_setup** file.

MAX_DATA_OE_SKEW and MAX_TSEL_SKEW should initially be set to 5% of the clock period but if this requirement can not be met, the budget could be relaxed and OEs from the PHY are asserted 1 cycle earlier than the data The *_DELAY values can be set to 1 cycle with the same caveat.

Table 5.12. Output Enable and Termination Relationships

Tag	STA Variable	STA Pass/Fail Variable	Description
OE_DQ	actual_skew_dq_oe_rise	MAX_DATA_OE_SKEW	Skew of the signals from the shift flip-flop to the output port data_oe (at the slice level) or the DQ pad (at the PHY level)
OE_DQ	actual_skew_dq_oe_fall	MAX_DATA_OE_SKEW	
OE_DQ	actual_skew_dq_oe	MAX_DATA_OE_SKEW	
OE_DQ	actual_skew_dq_oe	MAX_DATA_OE_DELAY	Delay of the signals from the clock pin of the shift flip-flop to the output port data_oe (at the slice level) or the DQ pad (at the PHY level)
TSEL_DQS	actual_skew_dqs_tsel_rise	MAX_TSEL_SKEW	Skew of the signals from the shift flip-flop to the output port tsel (at the slice level) or the DQS pad (at the PHY level)
TSEL_DQS	actual_skew_dqs_tsel_fall	MAX_TSEL_SKEW	
TSEL_DQS	actual_skew_dqs_tsel	MAX_TSEL_SKEW	
TSEL_DQS	actual_skew_dqs_tsel	MAX_TSEL_DELAY	Delay of the signals from the clock pin of the shift flip-flop to the output port tsel (at the slice level) or the DQS pad (at the PHY level)
TSEL_DQ	actual_skew_dq_tsel_rise	MAX_TSEL_SKEW	Skew of the signals from the shift flip-flop to the output port tsel (at
TSEL_DQ	actual_skew_dq_tsel_fall	MAX_TSEL_SKEW	
TSEL_DQ	actual_skew_dq_tsel	MAX_TSEL_SKEW	

Tag	STA Variable	STA Pass/Fail Variable	Description
			the slice level) or the DQ pad (at the PHY level)
TSEL_DQ	actual_delay_dq_tsel	MAX_TSEL_DELAY	Delay of the signals from the clock pin of the shift flip-flop to the output port tsel (at the slice level) or the DQ pad (at the PHY level)

5.4.1.6. Read DQS Gate Open

Read DQS gating is the blocking and passing of the input read DQS signal. The gate must be opened during the pre-amble of the read and closed during the post-amble. Since the DQS is asynchronous to the clk_phy signal, an STA constraint can not be written. However, the block diagram shown below depicts what must occur at the system level.

Figure 5.7. Read DQS Gate Open Timing in the Cadence Design IP DLL PHY

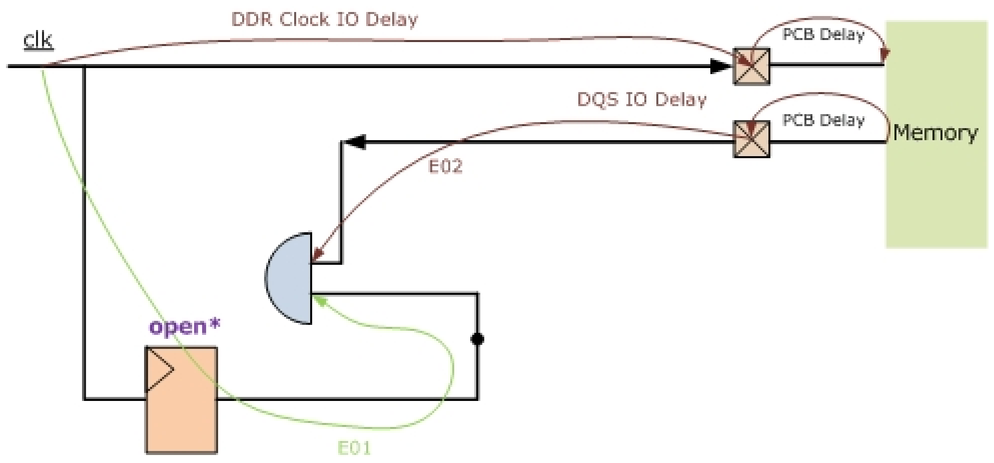


Table 5.13. Read DQS Gate Open/Close Relationships

Tag	STA Variable	STA Pass/Fail Variable	Description
E01			Delay from the gate flip-flops to the DQS
E02	actual_delay_dqs2nand	MAX_DQS2NAND_DELAY	DQS insertion delay from the DQS I/O buffer to the gating cell

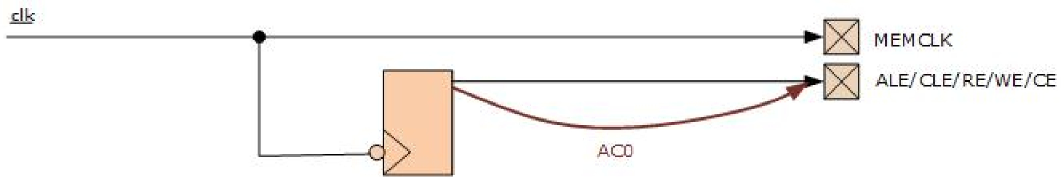
The gate signal travels straight to the gating cell. The open time is control by the gate_cfg field in the phy_gate_lpbk_ctrl_reg. The gate is closing when dfi_rddata_en is low.

There are no hard checks for the delays E01, E02i. Therefore, these paths need to be minimized. The STA scripts contain a check for these specific paths. To help verify that these paths have been minimized, the STA scripts provide a check using the user-defined variables shown in the following table. Also note that the E02 path will be defined during the read data path timing closure, so this value should not be changed once the read data path is timing closed.

5.4.1.7. Address/Control Timing

The timing on the address and control paths are not as critical as the data path. However, to ensure the largest margin, the delay from the output flip-flop to the pin is checked. This delay and the skew should be minimized.

Figure 5.8. Address/Control Block Diagram



There is an output delay applied to the address and control paths relative to the signal pad_mem_clk. This constraint forces STA to calculate the output latency of the memory clock and use this latency to calculate the setup and hold of the address/control to the memory clock at the pads of this design. It is expected that the system level timing will check this setup and hold at the memory device.

The relationships depicted in the above figure are detailed in the following table.

Table 5.14. Address/Control Relationships

Tag	STA Variable	STA Pass/Fail Variable	Description
AC0	actual_skew_addrcntrl_out_rise	MAX_ADDRCNTRL_SKEW	Skew between the address/ control outputs from the output flip-flop to the pad
AC0	actual_skew_addrcntrl_out_fall	MAX_ADDRCNTRL_SKEW	
AC0	actual_skew_addrcntrl	MAX_ADDRCNTRL_SKEW	
AC0	actual_delay_addrcntrl	MAX_ADDRCNTRL_DELAY	Delay from the address/control output flip-flop to the pad

5.4.1.8. Digital DLL checks

After building the DLL, the timing is verified with a special routine named **check_delay_line** that checks the delay of each delay cell. This routine resides in the file **sta_analysis_tools.tcl** and calculates the delay of the delay line for n elements (delay([n]), then calculates the delay of the delay line for n + 1 elements (delay[n+1]). The delay of a single element [n+1] is then calculated by subtracting the two calculations (delay[n+1] - delay[n]). This difference is compared to the user specified parameter ELEMENT_SKEW to ensure that no single step has a delay that is unreasonable.

Since there are multiple delay lines in the system, this timing check is performed at the delay line level and again for each delay line at the digital_dll level.

5.5. Specialized Reports

Cadence provides specialized reports for the synthesis and STA runs in reporting data as efficiently as possible.

5.5.1. Standard Reports

The following reports provide an overview of the STA:

- module.clock.ets

  These reports contain the results of standard STA tool reports for report clock, report_clock -skew and report clock -group.

- module.timing.max.ets

  This report contains the results of the standard STA tool report for worst-case timing with a full path.

- module.timing.max_summary.ets

  This report contains the results of the standard STA tool report for worst-case timing with a slack less than 0. This report includes only the beginning and ending points.

- module.timing.min.ets

  This report contains the results of the standard STA tool report for best-case timing with a full path.

- module.timing.min_summary.ets

  This report contains the results of the standard STA tool report for worst-case timing with a slack less than 0. This report includes only the beginning and ending points.

5.5.2. DLL Reports

This section lists all the DLL reports.

5.5.2.1. Digital Delay Line Delay Reports

The following reports describe the delay through the digital delay line with a select setting of [n] and [n+1]. The difference between the [n] and [n+1] results determines the delay of a single element [n+1]. This delay is compared to the input variables ACC and ELEMENT_SKEW. ACC is the expected average delay of each delay element, and ELEMENT_SKEW is the amount of skew allowed off of this average delay. This report is generated by the procedure **check_delay_line** in the file **sta_analysis_tools.tcl**.

- cdns_combo_dll_phy_digital_dll.master_phase_1_delay.rpt
- cdns_combo_dll_phy_digital_dll.master_phase_2_delay.rpt
- cdns_combo_dll_phy_digital_dll.read_delay.rpt
- cdns_combo_dll_phy_digital_dll.write_delay.rpt
- cdns_combo_dll_phy_dll_delay_line.element_delay.rpt

5.5.2.2. Digital Delay Line Skew Report

The following report describes the delay to digital DLL:

- cdns_combo_dll_phy_digital_dll.skew.rpt

  This report checks the delay from the select lines to the digital DLL to ensure that the skew is minimized. The file **sta_skew_check_dll_select.tcl** contains the procedure **sta_skew_check_dll_select** used to generate this report.

5.5.3. Custom Reports

This section lists all the custom reports.

Many of the timing checks required for timing closure can not be performed with standard constraints and timing reports. To ensure that timing is correct, custom timing reports are generated. The following subsections describe these report files.

5.5.3.1. Skew Reports

The following reports provide an overview of the skew:

- cdns_combo_dll_phy_data_slice.skew.slice.rpt

  This report is run on the slice level and calculates the required delays and skews for each slice. The report uses the input variables defined in the analyze section of the timing analysis spreadsheet. In the report, the full data path is shown and an error message is reported if the calculated delay/skew does not meet the input variable specification.

- cdns_combo_dll_phy_data_slice.suhld.slice.rpt

  Since the read data paths are analyzed by comparing the data insertion delay to the DQS insertion delay, setup and hold checks are not performed. If a setup and hold check is desired, this report shows the setup and hold value for every input entry flip-flop. It assigns variables to contain the worst-case setup and hold values and other variables with the skew for the setup and hold times. For example, if the setup values of the flip-flops are 10, 30 and 50ps, the setup variable will be assigned 50ps and the skew will be 40ps.

- cdns_combo_dll_phy.skew.slice_X.rpt

  This report is run on the PHY level for each slice and calculates the needed delays and skews for each slice. It uses the input variables defined in the analyze section of the timing analysis spreadsheet. In the report, the full data path is shown and an error message is reported if the calculated delay/skew does not meet the input variable specification.

5.5.3.2. Setup and Hold Report

The following report provides details on the setup and hold times for a single slice:

- cdns_combo_dll_phy.suhld.slice_X.rpt

  Since the read data paths are analyzed by comparing the data insertion delay to the DQS insertion delay, setup and hold checks are not performed. If a setup and hold check is desired, this report shows the setup and hold value for every input entry flip-flop of the slice. This report is not a setup/hold check from the port; rather, it details the times for this flip-flop. The results are similar to setup/hold information from an SDF file or a datasheet.

  This report assigns variables to contain the worst-case setup and hold values and other variables with the skew for the setup and hold times. For example, if the setup values of the flip-flops are 10, 30 and 50ps, the setup variable will be assigned 50ps and the skew will be 40ps.

  These setup/hold vales can be used if the user wishes to manually calculate the setup/hold times to these flip-flops from the port as part of a read data valid window calculation.

5.5.3.3. Address and Control Reports

The following report provides details on the address/control path skew:

- cdns_combo_dll_phy.skew.addrcntrl.rpt

  This report calculates the output delay and skew between the address and control signals.

5.5.3.4. Table reports

The tabular reports are tab-delimited summary reports. These are tab-delimited to allow simple import into spreadsheet format. The data included in these files is also reported in more verbose forms as listed in previous sections.

- cdns_combo_dll_phy_data_slice.skew_summary.rpt.ets

  This report is a tab-delimited summary of the report **cdns_combo_dll_phy_data_slice.skew.slice.rpt**. It shows the input threshold variable name, its value, the internal variable name used to hold the data and its value. This report is based on the slice level.

- cdns_combo_dll_phy_data_slice.suhld_summary.rpt.ets

  This report is a tab-delimited summary of the report **cdns_combo_dll_phy_data_slice.suhld.slice.rpt**. It shows the input threshold variable name, its value, the internal variable name used to hold the data and its value. This report is based on the slice level.

- cdns_combo_dll_phy.skew_summary.slice_X.rpt.ets

  This report is a tab-delimited summary of the report **cdns_combo_dll_phy.skew.slice_X.rpt**. One report is generated for each slice from the PHY level. It shows the input threshold variable name, its value, the internal variable name used to hold the data and its value. This report is based on the slice level.

- cdns_combo_dll_phy.skew_summary_table.rpt.ets

  This report is similar to **cdns_combo_dll_phy.skew_summary.slice_X.rpt**, except it contains the data for all the slices from the PHY level. It provides access to the delays and skews of all the slices at one time.

- cdns_combo_dll_phy.suhld_summary.slice_X.rpt.ets

  This report is a tab-delimited summary of the report **cdns_combo_dll_phy.suhld.slice_X.rpt**. It shows the input threshold variable name, its value, the internal variable name used to hold the data and its value. This report is based on the slice level.

  These setup/hold vales can be used if the user wishes to hand calculate the setup/hold times to these flip-flops from the port.

- cdns_combo_dll_phy.skew_summary.addrcntrl.rpt.ets

  This report is a tab-delimited summary of the **cdns_combo_dll_phy.skew.addrcntrl.rpt**.

5.5.3.5. Slice Summary

This section lists an example of Slice Summary and Slice Summary Table.

5.5.3.5.1. Slice Summary Example

The following table is an example of the **cdns_combo_dll_phy.skew_summary.slice_X.rpt.ets** and the **cdns_combo_dll_phy_data_slice.skew_summary.rpt.ets**. This table shows the variable from the user input file, the value of the user input variable, the name of the STA variable used to hold the measured value and the value of this variable.

Table 5.15. Example Slice Summary for Report Slice 0

Variable	Value	STA Variable	Value
MAX_WRITEDATA_SKEW	0.1	actual_skew_wr_mux_data_rise	0
MAX_WRITEDATA_SKEW	0.1	actual_skew_wr_mux_data_fall	0
MAX_WRITEDATA_SKEW	0.1	actual_skew_wr_mux_data	0.830992
MAX_WRITEDATA_DELAY	1.5	actual_delay_wr_mux_data	0.507344

5.5.3.5.2. Slice Summary Table Example

At the PHY level, the timing of each slice is checked. The information for each slice is summarized in the file **cdns_combo_dll_phy.skew_summary_table.rpt.ets.** The following table shows an example of this report.

The format of this table is similar to that of the individual slice reports. It shows the variable from the user input file, the value of the user input variable, the name of the STA variable used to hold the measured value and the value of this variable for each slice.

Table 5.16. Example Slice Summary

Variable	Value	STA Variable	STA Values			
			Slice 0	Slice 1	Slice 2	Slice 3

Variable	Value	STA Variable	STA Values			
MAX_WRITEDATA_SKEW	0.1	global_actual_skew_wr_mux_data_rise	0	0	0	0
MAX_WRITEDATA_SKEW	0.1	global_actual_skew_wr_mux_data_fall	0	0	0	0
MAX_WRITEDATA_SKEW	0.1	global_actual_skew_wr_mux_data	0.0303	0.0303	0.0303	0.0303
MAX_WRITEDATA_DELAY	3	global_actual_delay_wr_mux_data	1.722	1.722	1.722	1.722

5.5.4. Additional Checks

Once the timing scripts have been run, additional reviews of the timing should be done. Since all of the data is provided in a tab-delimited file, the results may be imported into a spreadsheet and checked.

5.5.5. Write Data Setup and Hold

The write data setup and hold STA scripts check the write data by verifying delay and skew. This approach allows for easy monitoring of the timing margin. When these delays are minimized, the write data setup and hold times relative to the DQS will fall out. Doing this check may be beneficial, although it is not automatically performed by the timing scripts.

```
Write setup = PERIOD/2 + Wdata - Wdqs - DLL_DELAY
Write hold = Wdqs + DLL_DELAY - Wdata
```

Wdata and Wdqs are defined in Write path and DLL_DELAY is the delay that the DLL will provide.

5.5.6. Read Data Setup and Hold

The read data setup and hold STA scripts check the read data by verifying delay and skew. This approach allows for easy monitoring of the timing margin. When these delays are minimized, the read data setup and hold times relative to the DQS will fall out. Doing this check may be beneficial, although it is not automatically performed by the timing scripts.

```
read setup = PERIOD/2 + Rdata - Rdqs - DLL_DELAY + ReadFlopSetup
read hold = Rdqs + DLL_DELAY - Rdata + ReadFlopHold
```

Rdata and Rdqs are defined in Section 5.4.1.2, “*Read Path*” and DLL_DELAY is the delay that the DLL will provide. The ReadFlopSetup and ReadFlopHold are the setup and hold values of the entry flip-flop. The setup and hold times are provided in the time reports **cdns_combo_dll_phy.suhld.slice_X.rpt** and **cdns_combo_dll_phy_data_slice.suhld.slice.rpt**.

5.5.7. Memory Clock Pulse Width

The pulse width of the write DQS can be calculated as follows:

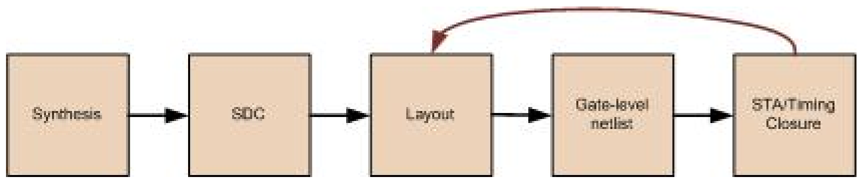
```
High MEMCLK Pulse Width = Period/2 - (MEMCLK_rise - MEMCLK_fall)
Low MEMCLK Pulse Width = Period/2 - (MEMCLK_fall - MEMCLK_rise)
```

MEMCLK_rise and MEMCLK_fall are reported in the report file **cdns_combo_dll_phy.skew_summary.addrcntrl.rpt** and **cdns_combo_dll_phy.skew_summary_table.rpt**. These are denoted by the STA variables actual_memclock_rise and actual_memclock_fall.

5.6. Synthesis Methodology

SDC file an be used in synthesis and in layout. Layout generates a gate-level netlist which is the input to the STA scripts. The layout may need to be modified to resolve STA issues. The general design flow is shown in the following figure.

Figure 5.9. Flow Diagram



5.7. Special Notes

This section lists few of the additional information on method of using *.lib* files, limitations of DLL *.lib* files, notes on TSEL timing and Timing loop.

5.7.1. Method for using .lib files

STA and synthesis assume a bottom-up approach. The last level is the digital DLL. It is assumed that the DLL is laid out first, and a *.lib* file is generated for it. This *.lib* file is expected to show only the fastest path through the DLL. This approach is used to ensure the constraints for the DLL are correct. In addition, this methodology mirrors that of designs using a third party DLL, which allows leverage of the same scripts.

5.7.2. Limitations of DLL .lib Model

Since the DLLs in this design are intended for tuning timing, it is not possible to run STA with the exact values that will be used in the lab. To approximate the delay of the DLL, an ideal value is used and applied to the clock generation. For STA, the generated clock for the DLLs are created at the input of the DLL to ensure that STA sees the clock insertion time to the DLL. This is unlike the synthesis scripts, which place the generated clock at the output of the DLL to allow synthesis to run without having a DLL model in the synthesis run. Since synthesis runs with ideal clocks, the insertion delay to the DLL is not needed.

The ideal shift on the clocks is typically 1/4 clock (3/4 for the clk_wr signal). Since the DLL is not able to achieve a perfect 1/ 4-clock shift, there is a certain amount of error that is introduced. This error is referenced as DLL_ERROR and is applied to the clocks as an uncertainty term. DLL_ERROR should be set to one delay element (for a digital DLL) or the delay of the difference between 2 settings for an analog DLL.

5.7.3. TSEL Timing

At the PHY level, the timing of TSEL is from the core logic to the IO pad. Some IO models do not model this path. In this case, the file **sta_skew_check_slice.tcl** may be modified to point to the tsel pin of the IO buffer instead of using the port.

Appendix A. Document Revision History

Table A.1. Document Revision History

Revision	Modification	Data	Author
1.0	First Release	06 Mar 2019	ppietron@cadence.com
1.1	Added byte masking capability	24 May 2021	rousset@cadence.com