cadence®

SD/eMMC DLL PHY IP

Internal Name: SD/eMMC PHY
Part Number: IP6185

NDA # _____

User Guide

Revision: 1.02

CADENCE CONFIDENTIAL

Confidentiality Notice

Cadence Design Systems, Inc. San Jose, CA 95134

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks

Restricted Permission

Disclaimer

Restricted Rights

Table of Contents

List of Figures

List of Tables

1. Overview

The Cadence Design IP - SD/eMMC Soft PHY subsequently referred to in this document as the PHY encapsulates all functionality required to interface to external SD/eMMC devices into a single module. The IP supports SD/eMMC speed modes (DEFAULT / HIGHSPD / UHSI_SDR12 / UHSI_SDR25 / UHSI_SDR50 / UHSI_SDR104 / UHSI_DDR50 / MMC_LEGACY / MMC_SDR / MMC_DDR / MMC_HS200 / MMC_HS400 / MMC_HS400_ES). This module is used to control the off-chip data capture and synchronization logic for the read data. This module performs the following functions:

- Contains all data registers used to launch data, address and control signals to the device and the memory controller.
- Controls the off-chip data capture and synchronization logic for the read data.
- Includes an all-digital DLL for timing.

The Cadence Design IP DLL PHY is built from a standard cell library and a digital DLL. Cadence provides the complete solution and manages the acquisition of each of these components.

The Cadence Design IP DLL PHY also incorporates at-speed manufacturing testability circuitry for the datapath elements which allows it to be easily self-checked at commercially available Automatic Test Equipment (ATE) speeds. Non-critical timing flip-flops can be checked through traditional DFT techniques if the customer wishes to insert these structures.

This chapter consists of the following sections:

1.1. Pin Index

This section provides a list of all the signals to/from the PHY.

1.1.1. ASIC to PHY Sideband Signals

The following table lists the clocks, resets and scan related signals to the PHY.

Table 1.1. ASIC to PHY Sideband Signals.

Signal Name	Size	Type	Description	Reset Value	Associated clock
clk_phy	1	Input	Clock input to PHY. This is the main clock signal of the frequency consistent with controller application and device work mode.	N/A	clk_phy
iddq_en	1	Input	Input signal used to control testing. This signal must be set to test IDDQ.	N/A	async
rst_n	1	Input	Reset input to PHY. This active-low reset, when asserted, resets most of the PHY logic besides the programmable registers and related logic.	N/A	async

Signal Name	Size	Type	Description	Reset Value	Associated clock
scanmode	1	Input	Scan mode active.	N/A	clk_phy
scan_en	1	Input	Scan enable.	N/A	clk_phy
scan_atspeed_tmode	1	Input	Scan test mode - set high for at-speed mode to run scan at operational frequency; set low for low frequency scan mode. For normal functional mode of the PHY this signal must be set high.	N/A	clk_phy

1.1.2. General Purpose I/O Signals

The following table lists the general purpose I/O signals. Typically, these signals can be used to control the static settings in the I/O pads and to get status information from the I/O pads.

Table 1.2. General Purpose I/O Signals.

Signal Name	Size	Type	Description	Reset Value	Associated clock
phy_gpio_reg_ctrl_0	32	Output	32 bits of general purpose I/O control signals.	32'h0	clk_phy
phy_gpio_reg_ctrl_1	32	Output	32 bits of general purpose I/O control signals.	32'h0	clk_phy
phy_gpio_reg_status_0	32	Input	32 bits of general purpose I/O status signals.	N/A	clk_phy
phy_gpio_reg_status_1	32	Input	32 bits of general purpose I/O status signals.	N/A	clk_phy

1.1.3. Tie off PHY Sideband Signals

The following table lists the Tie-off signals to the PHY. These signals inform the PHY of the polarity of the output enables required to drive data onto the bidirectional I/O pads to the Flash devices.

Table 1.3. Tie off PHY Sideband Signals.

Signal Name	Size	Type	Description	Reset Value	Associated clock
addr_cntrl_oe_polarity	1	Input	Output enable polarity parameter for control IO PADs ie. all signals except DATA IO PADs , CMD IO PAD , LPBK_DQS IO PAD and DQS IO PAD. 0 - Control signals pad OE active low. 1 - Control signals pad OE active high.	N/A	async
addr_cntrl_ie_polarity	1	Input	Input enable polarity parameter for control IO PADs ie. all signals except DATA IO PADs , CMD IO PAD , LPBK_DQS IO PAD and DQS IO PAD. 0 - Control signals pad IE active low. 1 - Control signals pad IE active high.	N/A	async
slice_oe_polarity	1	Input	Output enable polarity parameter for DATA IO PADs , CMD IO PAD , LPBK_DQS IO PAD and DQS IO PAD. 0 - Control signals pad OE active low. 1 - Control signals pad OE active high.	N/A	async
slice_ie_polarity	1	Input	Input enable polarity parameter for DATA IO PADs , CMD IO PAD , LPBK_DQS IO PAD and DQS IO	N/A	async

Signal Name	Size	Type	Description	Reset Value	Associated clock
			PAD. 0 - Control signals pad IE active low. 1 - Control signals pad IE active high.		

1.1.4. Controller/PHY Signals

The following tables list all the signals between the Controller and the PHY.

Table 1.4. Controller/PHY Signals

Signal Name	Size	Type	Description	Reset Value	Associated clock
dfi_ale	1	Input	Address latch enable.	N/A	clk_phy
dfi_rebar	1	Input	Read enable.	N/A	clk_phy
dfi_webar	1	Input	Write enable.	N/A	clk_phy
dfi_webar_high	1	Input	Write enable high.	N/A	clk_phy
dfi_wpbar_a	1	Output	Write protect - asynchronous passthrough the PHY (from device through PHY to controller).	1'h0	clk_phy
dfi_wrdata_en	8	Input	Write data-in valid.	N/A	clk_phy
dfi_wrdata	16	Input	Write data to the PHY.	N/A	clk_phy
dfi_wrcmd_en	1	Input	Write cmd-in valid.	N/A	clk_phy
dfi_wrcmd	2	Input	Write cmd to the PHY.	N/A	clk_phy
dfi_rddata	16	Output	Read data from the PHY.	16'h0	clk_phy
dfi_rddata_valid	1	Output	Read data from the PHY valid indicator.	1'h0	clk_phy
dfi_rddata_en	1	Input	Read data enable. Opens the DQS gate.	N/A	clk_phy
dfi_rdcmd_en	1	Input	Read data enable. Opens the DQS gate.	N/A	clk_phy
dfi_rddata_a	8	Output	Asynchronous passthrough from DQ (DATA) signal.	8'h0	clk_phy
dfi_rdcmd	1	Output	Read cmd from the PHY.	1'h0	clk_phy
dfi_rdcmd_valid	1	Output	Read cmd from the PHY valid indicator.	1'h0	clk_phy
dfi_rdcmd_a	1	Output	Asynchronous passthrough from CMD (COMMAND) signal.	1'h0	clk_phy
dll_rst_n	1	Input	Resets the FIFO pointers in the read path. After de-assertion, the master DLL begins searching for lock.	N/A	clk_phy
dfi_dqs_underrun	1	Output	Status signal to indicate that the logic gate had to be forced closed. It indicates that either the DQS strobe did not appear during read or rd_del_sel signal value is too low and dfi_rddata are corrupted. The dll_rst_n or rst_n clears this flag.	1'h0	clk_phy
dfi_dqs_overflow	1	Output	Status signal to indicate that the logic gate was closed too late ie. the number of DQS strobes exceed the capacity of the entry FIFO. It indicates that rd_del_sel signal value is too high and dfi_rddata are corrupted. The dll_rst_n or rst_n clears this flag.	1'h0	clk_phy

Signal Name	Size	Type	Description	Reset Value	Associated clock
dfi_dqs_cmd_underrun	1	Output	Status signal to indicate that the logic gate had to be forced closed. It indicates that either the DQS strobe did not appear during read or rd_del_sel signal value is to low and dfi_rddata are corrupted. The dll_rst_n or rst_n clears this flag.	1'h0	clk_phy
dfi_dqs_cmd_overflow	1	Output	Status signal to indicate that the logic gate was closed too late ie. the number of DQS strobes exceed the capacity of the entry FIFO. It indicates that rd_del_sel signal value is to high and dfi_rddata are corrupted. The dll_rst_n or rst_n clears this flag.	1'h0	clk_phy
param_extended_rd_mode	1	Input	Information to the PHY to keep RE# extended for a few controller clock cycles to guarantee data capture.	N/A	clk_phy
param_extended_wr_mode	1	Input	Informs the PHY that the WE# can follow the WE# from the controller and that the controller will take care of the timing associated with the data and WE#.	N/A	clk_phy
dfi_ctrlupd_req	1	Input	MC-initiated updates are used in the Cadence Flash PHY and the PHY will respond with a dfi_ctrlupd_ack assertion. This signal is set to update the slave DLLs and reset the FIFOs in the read path. Therefore, it is expected that the controller will assert this signal regularly during normal operation to compensate for any voltage/temperature drifts.	N/A	clk_phy
dfi_ctrlupd_ack	1	Output	PHY acceptance of a MC-initiated update.	1'h0	clk_phy
dfi_init_complete	1	Output	PHY initialization complete signal. This signal indicates that the PHY is initialized and ready to accept Flash memory device commands from the controller.	1'h0	clk_phy
dfi_rstbar	1	Input	Reset.	N/A	clk_phy
dfi_ctrl_0_a	1	Output	Control Signal - mem_ctrl_0_ipad port is directly routed to this port.	1'h0	clk_phy
dfi_ctrl_1	1	Input	Control signal 1.	N/A	clk_phy
dfi_1v8	1	Input	1v8 voltage enable signal to control the IO PAD.	N/A	clk_phy

Table 1.5. APB Register Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
reg_pclk	1	Input	Clock - The rising edge of PCLK times all transfers on the APB.	N/A	reg_pclk
reg_preset_n	1	Input	Reset - The APB reset signal is active LOW.	N/A	async
reg_paddr	16	Input	Address - This is the APB address bus.	N/A	reg_pclk
reg_psel	1	Input	Select - It indicates that the slave device is selected and that a data transfer is required.	N/A	reg_pclk
reg_penable	1	Input	Enable - This signal indicates the second and subsequent cycles of an APB transfer.	N/A	reg_pclk

Signal Name	Size	Type	Description	Reset Value	Associated clock
reg_pwrite	1	Input	Direction - This signal indicates an APB write access when HIGH and an APB read access when LOW.	N/A	reg_pclk
reg_prdata	32	Output	Read Data - The selected slave drives this bus during read cycles when PWRITE is LOW.	32'h0	reg_pclk
reg_pwdata	32	Input	Write data - during write cycles when PWRITE is HIGH.	N/A	reg_pclk
reg_pready	1	Output	Ready - The slave uses this signal to extend an APB transfer.	1'h0	reg_pclk

1.1.5. Pads/PHY Signals

The following tables list all the signals between the PHY and the I/O PADS to/from the Flash device.

Table 1.6. IO PADs Interface

Signal Name	Size	Type	Description	Reset Value	Associated clock
mem_ale_opad	1	Output	Address latch enable.	1'h0	clk_phy
mem_ale_oepad	1	Output	Address latch enable output enable.	1'h0	clk_phy
mem_ale_iepad	1	Output	Address latch enable input enable.	1'h0	clk_phy
mem_rebar_opad	1	Output	Read enable.	1'h1	clk_phy
mem_rebar_oepad	1	Output	Read enable output enable.	1'h0	clk_phy
mem_rebar_iepad	1	Output	Read enable input enable.	1'h0	clk_phy
mem_rebar_ipad	1	Input	Read enable input signal - used as internal lpbk_dqs.	N/A	clk_lpbk
lpbk_dqs_oepad	1	Output	lpbk_dqs pad output enable.	1'h0	clk_phy
lpbk_dqs_iepad	1	Output	lpbk_dqs pad input enable.	1'h0	clk_phy
lpbk_dqs_ipad	1	Input	lpbk_dqs pad input data - used as source of external lpbk_dqs, PCB-routed from RE output.	N/A	clk_lpbk
mem_webar_opad	1	Output	SDCLK - clock to the device.	1'h1	clk_phy
mem_webar_oepad	1	Output	Write enable output enable.	1'h0	clk_phy
mem_webar_iepad	1	Output	Write enable input enable.	1'h0	clk_phy
mem_wpbar_ipad	1	Input	Write protect input.	N/A	clk_phy
mem_wpbar_oepad	1	Output	Write protect output enable.	1'h0	clk_phy
mem_wpbar_iepad	1	Output	Write protect input enable.	1'h0	clk_phy
dqs_oepad	1	Output	DQS pad output enable.	1'h0	clk_phy
dqs_iepad	1	Output	DQS pad input enable.	1'h0	clk_phy
dqs_ipad	1	Input	DQS pad input data.	N/A	mem_dqs
data_opad	8	Output	DQ pad output data.	8'h0	clk_phy
data_oepad	8	Output	DQ pad output enable.	8'h0	clk_phy
data_iepad	8	Output	DQ pad input enable.	8'h0	clk_phy

Signal Name	Size	Type	Description	Reset Value	Associated clock
data_ipad	8	Input	DQ pad input data.	N/A	
cmd_opad	1	Output	CMD pad output data.	1'h0	clk_phy
cmd_oepad	1	Output	CMD pad output enable.	1'h0	clk_phy
cmd_iepad	1	Output	CMD pad input enable.	1'h0	clk_phy
cmd_ipad	1	Input	CMD pad input data.	N/A	
mem_ctrl_0_ipad	1	Input	ctrl_0 pad input data.	N/A	clk_phy
mem_ctrl_1_opad	1	Output	ctrl_1 pad output data.	1'h1	clk_phy
mem_ctrl_1_oepad	1	Output	ctrl_1 pad output enable.	1'h0	clk_phy
mem_ctrl_1_iepad	1	Output	ctrl_1 pad input enable.	1'h0	clk_phy
mem_rstbar_opad	1	Output	rstbar pad output data.	1'h1	clk_phy
mem_rstbar_oepad	1	Output	rstbar pad output enable.	1'h0	clk_phy
mem_rstbar_iepad	1	Output	rstbar pad input enable.	1'h0	clk_phy
tsel_dqs_0_opad	1	Output	Dynamic termination select signals sent to the data strobe I/O pins, controlled by the tsel_rd_value_dqs and tsel_off_value_dqs fields programmable registers. tsel_dqs_0_opad represents the value of tsel_rd_value_dqs[0] or tsel_off_value_dqs[0] bit of the phy_tsel_reg.	1'h0	clk_phy
tsel_dqs_1_opad	1	Output	Dynamic termination select signals sent to the data strobe I/O pins, controlled by the tsel_rd_value_dqs and tsel_off_value_dqs fields programmable registers. tsel_dqs_1_opad represents the value of tsel_rd_value_dqs[1] or tsel_off_value_dqs[1] bit of the phy_tsel_reg.	1'h0	clk_phy
tsel_dqs_2_opad	1	Output	Dynamic termination select signals sent to the data strobe I/O pins, controlled by the tsel_rd_value_dqs and tsel_off_value_dqs fields programmable registers. tsel_dqs_2_opad represents the value of tsel_rd_value_dqs[2] or tsel_off_value_dqs[2] bit of the phy_tsel_reg.	1'h0	clk_phy
tsel_dqs_3_opad	1	Output	Dynamic termination select signals sent to the data strobe I/O pins, controlled by the tsel_rd_value_dqs and tsel_off_value_dqs fields programmable registers. tsel_dqs_3_opad represents the value of tsel_rd_value_dqs[3] or tsel_off_value_dqs[3] bit of the phy_tsel_reg.	1'h0	clk_phy
tsel_data_0_opad	8	Output	Dynamic termination select signals sent to the data I/O pins, controlled by the tsel_rd_value_data and tsel_off_value_data fields programmable registers.	8'h0	clk_phy

Signal Name	Size	Type	Description	Reset Value	Associated clock
			Each bit of tsel_data_0_opad represents the value of tsel_rd_value_data[0] or tsel_off_value_data[0] bit of the phy_tsel_reg.		
tsel_data_1_opad	8	Output	Dynamic termination select signals sent to the data I/O pins, controlled by the tsel_rd_value_data and tsel_off_value_data fields programmable registers. Each bit of tsel_data_1_opad represents the value of tsel_rd_value_data[1] or tsel_off_value_data[1] bit of the phy_tsel_reg.	8'h0	clk_phy
tsel_data_2_opad	8	Output	Dynamic termination select signals sent to the data I/O pins, controlled by the tsel_rd_value_data and tsel_off_value_data fields programmable registers. Each bit of tsel_data_2_opad represents the value of tsel_rd_value_data[2] or tsel_off_value_data[2] bit of the phy_tsel_reg.	8'h0	clk_phy
tsel_data_3_opad	8	Output	Dynamic termination select signals sent to the data I/O pins, controlled by the tsel_rd_value_data and tsel_off_value_data fields programmable registers. Each bit of tsel_data_3_opad represents the value of tsel_rd_value_data[3] or tsel_off_value_data[3] bit of the phy_tsel_reg.	8'h0	clk_phy
v18	1	Output	1v8 voltage enable signal to control the IO PAD.	1'h0	clk_phy

1.1.6. Functional Description of I/O Pin to/from the SD/eMMC Device

The following table lists all the signals between the I/O and to/from the SD/eMMC device.

Table 1.7. Functional Description of each I/O Pin to/from the SD/eMMC Device

Signal Name	Direction (PHY to SD/eMMC)	Description
CLK	OUT	SD/eMMC PHY to card clock signal - mem_webar* implements this function.
CMD	BIDIR	Command/Response signal - cmd* signal implements this function.
DATA(7:0)	BIDIR	Data signal - data* signal implements this function.
DATA_STROBE	IN	Data strobe signal - dqs* signal implements this function.
RESET	OUT	eMMC Reset signal - mem_rstbar* implements this funcion.
BUS_POWER	OUT	Device bus power - mem_ctrl_1* implements this function.
CARD_DETECT	IN	Card detect input signal - mem_ctrl_0* implements this function.
WRITE_PROTECT	OUT	Write Protect output signal - mem_wpbar* implements this function.
PU_PD_DATA2	OUT	Pull-up/Pull-down signal for bit 2 of the DATA - mem_ale* implements this function.

1.2. Initialization Procedure

The Cadence Design IP DLL PHY is designed such that it requires a sequence for correct operation after all power to the ASIC and to the memory devices is stable. The PHY does not include circuitry to control the activation of power and ground to the system. The Cadence Design IP DLL PHY should be held in reset until the power and ground levels are stable.

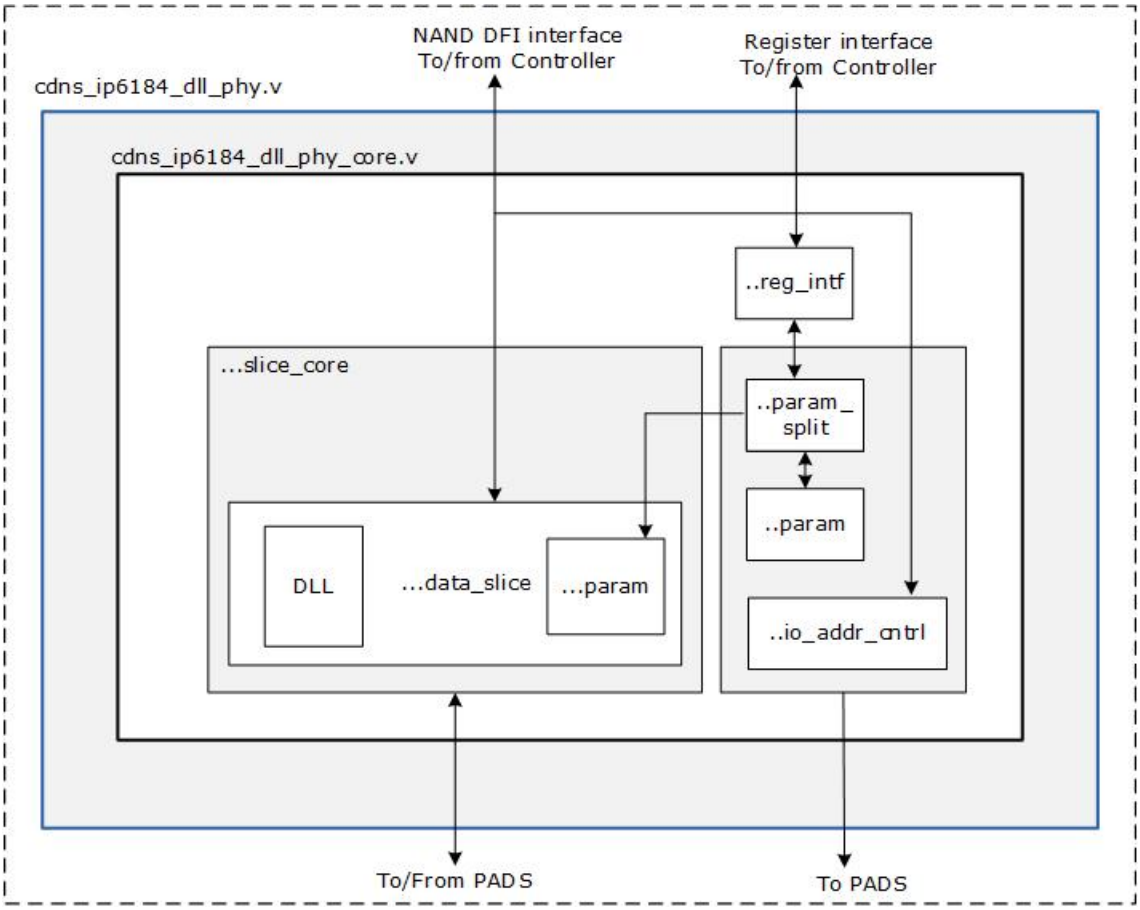
The controller requires notice that the PHY is fully initialized. Once the initialization has completed, the PHY will assert the **dfi_init_complete signal**. This indicates that the PHY and memory devices are ready to accept commands.

The procedure to initialize the Cadence Design IP DLL PHY is as follows:

- Assert the **rst_n** by driving it to ’b0. All programmable registers will be cleared and set to their default values. These signals may be asserted asynchronously and in any order.
- De-assert the **rst_n** signals (to ’b1) such that there are no reset recovery/removal violations after the reset trees have been inserted.
- Finish device initialization and basic operations that may be required incase the host is booting from the connected flash devices. All transactions to and from the device will be carried out in extended read/write modes. This will guarantee successful operations albeit with low performance.
- Assert the **dll_rst_n** by driving it to 'b0. The clocks to the PHY can be changed now.
- Program all the PHY registers. There is a script provided with the drop to assist in the generation of values that need to be programmed.
- De-assert the dll_rst_n signal (to ’b1). This signal is re-synchronized within each slice so there are no special requirements for its de-assertion.
- The PHY slice’s master DLLs will lock and then the PHY will assert the dfi_init_complete signal. This indicates that the PHY and memory devices are ready to accept commands
- The PHY is now ready for normal traffic.

1.3. Architecture

Figure 1.1. Top Level Diagram of the DLL PHY



1.3.1. Write Path

There are 8 data bits and one cmd bit in the data slice, and these are used to capture the dfi_wrdata and dfi_wrcmd signals on positive-edge flip-flops. The clock to these flip-flops is synchronous to the DFI clock; however a clean version is desired in this critical path, so balancing at the system level may be required. Also, this block uses both edges of the clk_phy. The clk_wr slave delay line is used to output DDR data to the IO pads. The clk_wr also controls output of the cmd signal to the IO pad.

Data transfers between write_data_h_reg/write_data_l_reg and wr_h_in/wr_l_in (input signal to wr_h/wr_l flops) depends on the relationship between the clk_phy and clk_wr signals. The wr_h_in/wr_l_in is transferred to the clk_wr domain, and a multiplexer that uses the clk_wr signal as the select converts the SDR data into a DDR data. The clk_wr domain and the clk_phy domain should always remain 1/4 cycle out of phase, but clk_wr can be shifted relative to the clk_phy during the write training operation. The signal half_cycle_shift is the sampled value of the clk_wr signal by the clk_phy signal. These lines must also be balanced for delay and minimum skew. Clock-type cells, or cells with very balanced rise/fall times, should be used for both the multiplexer and any buffers to the pads. Shielding and noise avoidance techniques should be applied to the write-path signals as well. The output of the eigth multiplexers should be treated as a skew group with constraints written to minimize the skew between the eight DQ lines as much as possible.

Figure 1.2. Block diagram of the write path in the PHY

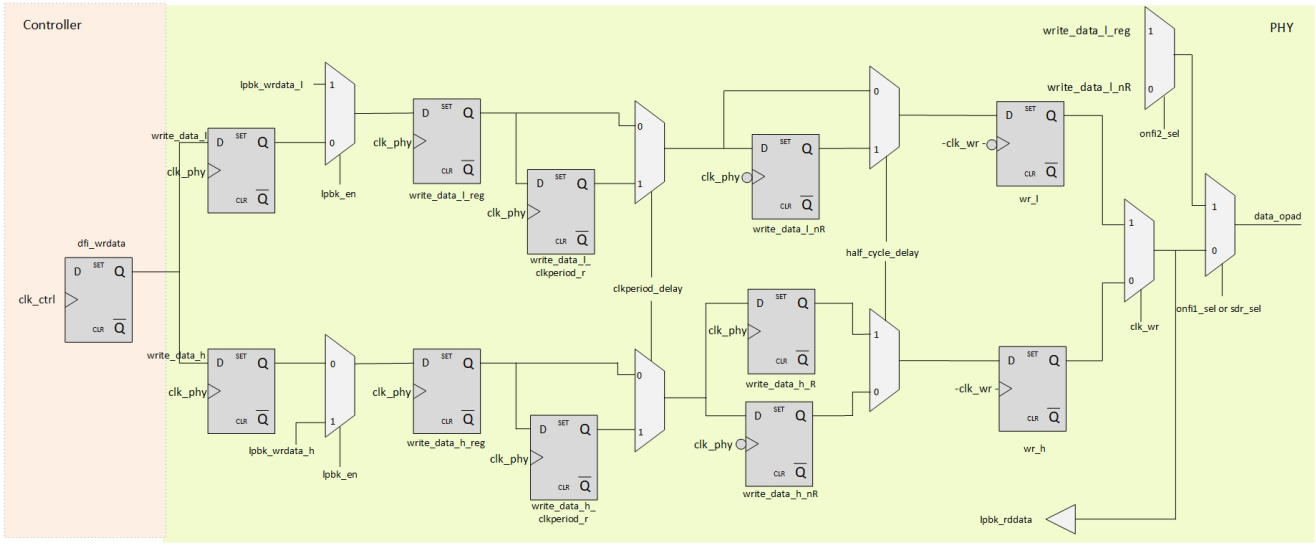
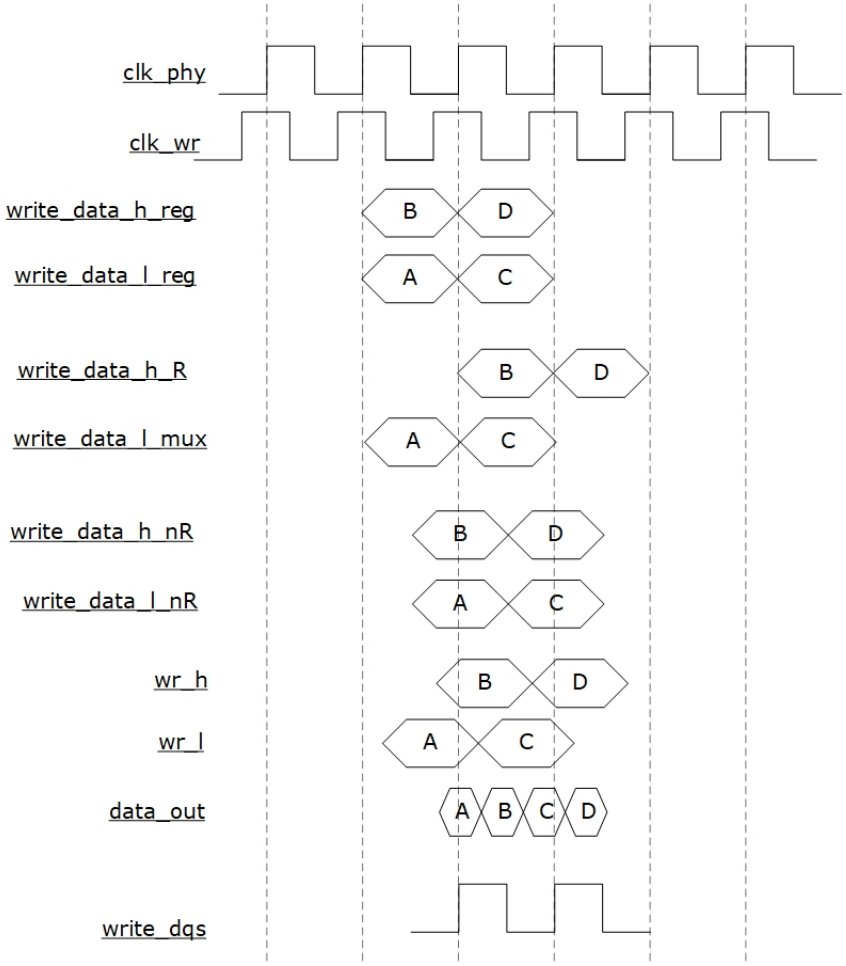


Figure 1.3. Write path timing diagram



1.3.2. Write DQS data

1.3.3. Read Path

The read data capture logic is responsible for capturing the DQ outputs from the SD/eMMC device and passing the data back to the system clock domain. The DQS strobe used to capture data is delayed to ensure that the rising and falling edges of the strobe is in the middle of the valid window of data.

In the scenario where the SD/eMMC device works in very slow modes (modes below SDR104/HS200/HS400/HS400ES) and there will be no DQS generated from the device, incoming data is latched by a version of the SDCLK that is looped back via two pads. The mux that switches between the looped back SDCLK and DQS from the device is controlled through a programmable register. For SD/eMMC using the loopback version of the device clock is the required approach for modes where DQS is not generated by the device.

For SD interface there is additional read path dedicated for CMD signal. It is build the same way as the DATA path and contains seperate clocking mechanism.

The DDR data for a particular entry is clock cycled every 8 DQS clock cycles. Data is transferred from the entry flops to the PHY clock cycle domain after the program delay which allows for the falling edge data to be received, and sufficient setup time for the PHY clock cycle to parameter both rising and falling edge data.

System analysis of the latest data arrival to the slice determines the programming value for the cycle in which data is read out of the FIFO. The entry parameters are valid for a minimum of 7 cycles after the receipt of data which provides significant margin to reliably transfer the DDR data to the internal PHY clk_phy domain. The FIFO read control advances the read pointer based upon a programmable delay of the dfi_rddata_en pulse from the DFI interface. The need to synchronize the data from the SD/eMMC devices to the system clock is based on factors that affect the Async_Delay. There is variability in when the data arrives from memory. The differences between the earliest and latest arrival times are caused by the large range between minimum and maximum delays of the input cells of the ASIC, the distance from the SD/eMCC device to the ASIC, and the other factors that comprise the Async_Delay. The Cadence Design IP - SD/eMMC Soft PHY is designed to handle variations in the delay when read data is returned by the external memory.

Figure 1.4. DQS Read Timing

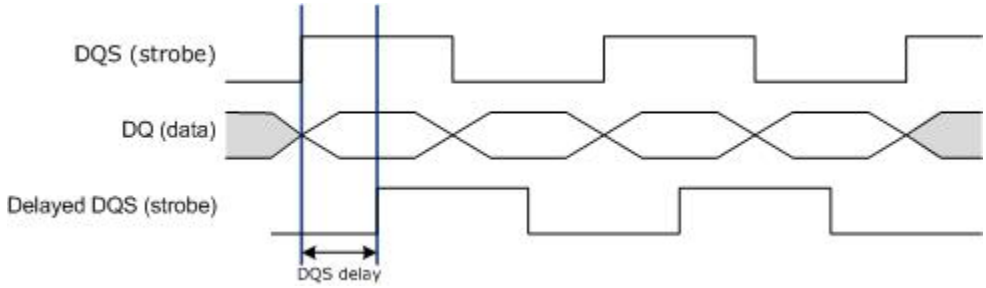


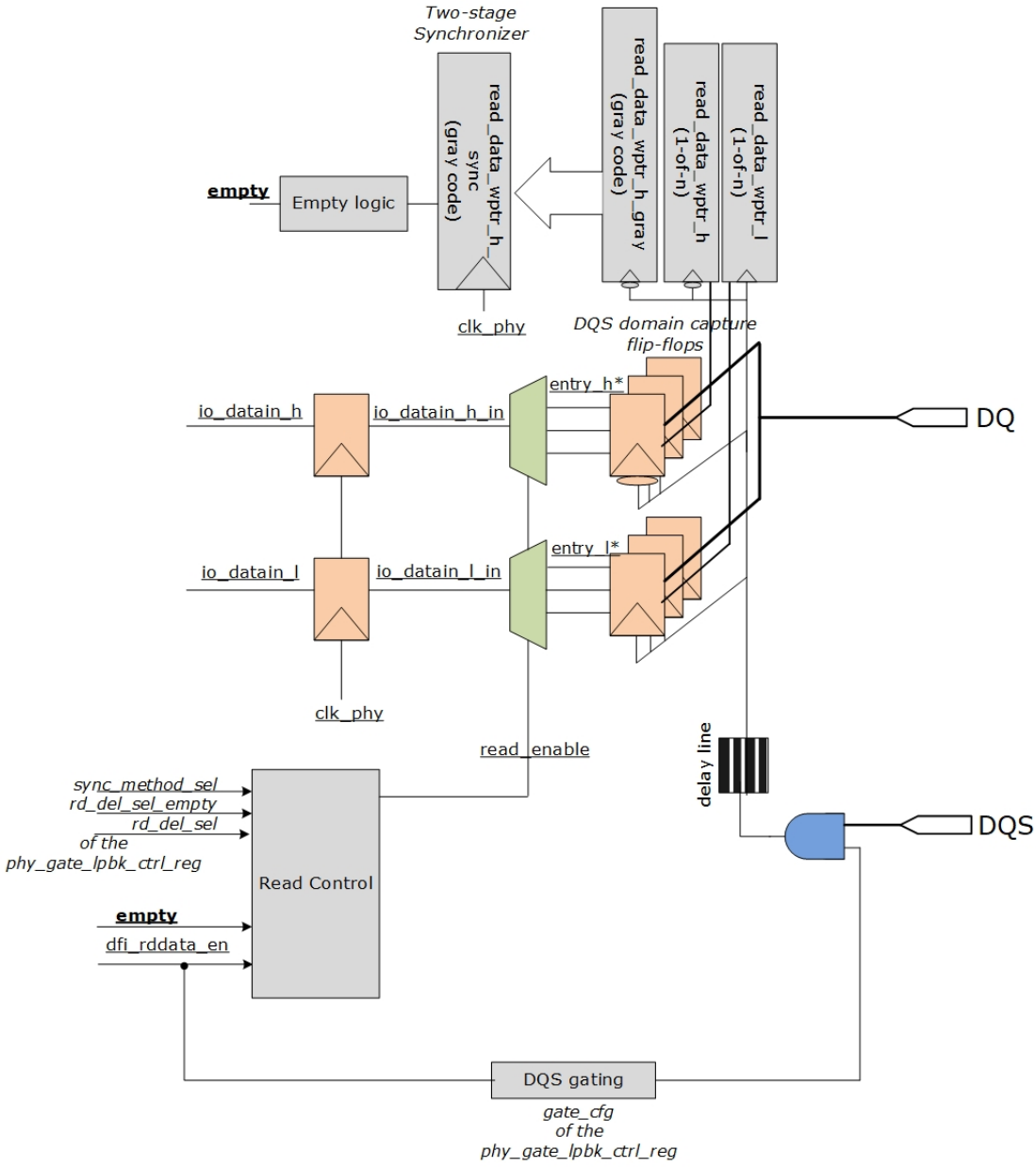
Table 1.8. Read Capture Signal Descriptions

Signal	Description
clk_phy	Main PHY clock.
delayed_dqs	Delayed version of the data strobe from the flash devices.
mem_data/mem_cmd	Incoming data/cmd from the flash devices.
io_datain_h/l[7:0]	Read data synchronized to the PHY clock domain.
read_data_rptr[7:0]	Selects which set of registers to choose when transferring data from the data strobe domain to the system clock domain. This signal is one-hot encoded and is synchronous to the data strobe read domain.

Signal	Description
read_data_wptr[7:0]	Enable signal that controls which set of registers to use when capturing data. This signal is always one-hot encoded. This signal transitions based on the timing of the data strobe read domain clock.
dqs_underrun	This signal indicates when the read_data_rptr and read_data_wptr are equal when rptr_upd is asserted which means that either DQS signal did not appear or rd_del_sel signal value is too low. After this signal goes high it stays high until dll_rst_n or rst_n is asserted low.
dqs_overflow	This signal the number of DQS strobes exceed the capacity of the entry FIFO which means that rd_del_sel signal value is too high. It is possible that overflow status is asserted with underrun status - in such case the overflow takes the precedence. After this signal goes high it stays high until dll_rst_n or rst_n is asserted low.

The basic idea for the read capture is to update the write pointers when valid DQS strobes arrive and read out the data from this FIFO using the read pointers after a set delay. This delay is a programmable value and is based on the board flight times as well as the worst case timing of the connected flash device to return valid data. The FIFO depth is 8 deep and the latency between DQS arriving from the device to data being read out of the FIFO can be 8 clock cycles. The block diagram for the read path in the PHY is shown below.

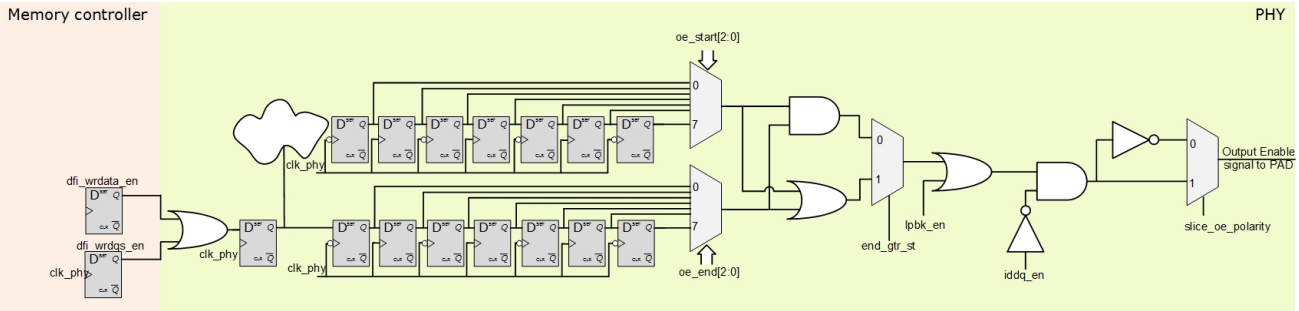
Figure 1.5. Block diagram of the read path in the PHY



1.3.4. Output Enable generation block

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 multiplexer selects are programmable values which allow for adjustment in the turn on/off time. The output enable generation logic is fairly non time-critical. The output enable and disable times can be adjusted with a 1/2-cycle resolution using the phy_dq_timing_reg or phy_dqs_timing_reg register settings. The output enable signal is fanned out to all DQ pins and one DQS pin. The signals data_oepad and dqs_oepad should be grouped into a skew group and balanced to each other as well as possible.

Figure 1.6. Output Enable generation block



1.3.5. TSEL/DQS gate relationship

TSEL will start after detection of the negative edge of the 'rebar_dfi' signal. The end of the TSEL signal is determined by the closing time of the internal DQS gate signal.

1) TSEL ON condition - The TSEL is ON minimum two clock cycles after the negative edge of the 'rebar_dfi' signal. The start time can be delayed by 'dqs_select_tsel_start' or 'data_select_tsel_start' fields. The delay resolution is half clock cycle.

2) TSEL OFF condition - The TSEL is OFF minimum one clock cycle after the internal DQS gate signal goes low. The end time can be delayed by 'dqs_select_tsel_end' or 'data_select_tsel_end' fields The delay resolution is half clock cycle.

The internal DQS gate signal is controlled by the 'gate_cfg'. The end of the DQS gate signal can be additionally delayed by 'gate_cfg_close'. The delay resolution is clock cycle for both parameters. The DQS gate signal is asserted high when 'dfi_rddata_en' goes high and de-asserted when 'rebar_dfi' is high or when there is no 'cebar_dfi' (ie. all 'cebar_dfi' signals are high)

1.3.6. Input Enable generation block

The input enable logic and the output enable logic are almost built in the same way. The Input enable and disable times can be adjusted with a 1/2- cycle resolution using the phy_ie_timing_reg register setting. The input enable signal is fanned out to all DQ pins and one DQS pin. The signals data_iepad [7:0] and dqs_iepad should be grouped into a skew group and balanced to each other as well as possible.

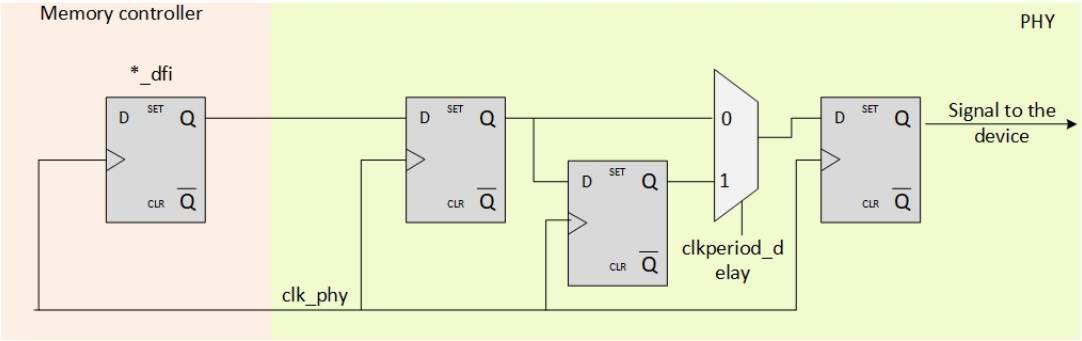
SD/eMMC - There is a danger that data transferred from the device will be received and last bit of transferred frame could be interpreted as begin of received frame. Even thought the standard defines minimal timings between transmitted and received data, the PHY cannot handle this because of delays which are unknown for host controller (I/O PAD and PCB). For SD an additional IO mask logic has been introduced to keep input in high state when it should be inactive (when transfer is performed to the device). The logic works in similar way to the input enable logic but the does not use the IO PAD function. The maximum where the mask is active is 20ns ie. the maximum delay from PHY to device and back cannot exceed this value. IO Mask logic is controlled by 'io_mask*' parameters in the phy_dq_timing_reg.

1.3.7. Control Signal generation block

1.3.7.1. CE#, WP# generation block

The following figure shows the launching of CE# and WP# signals to the device.

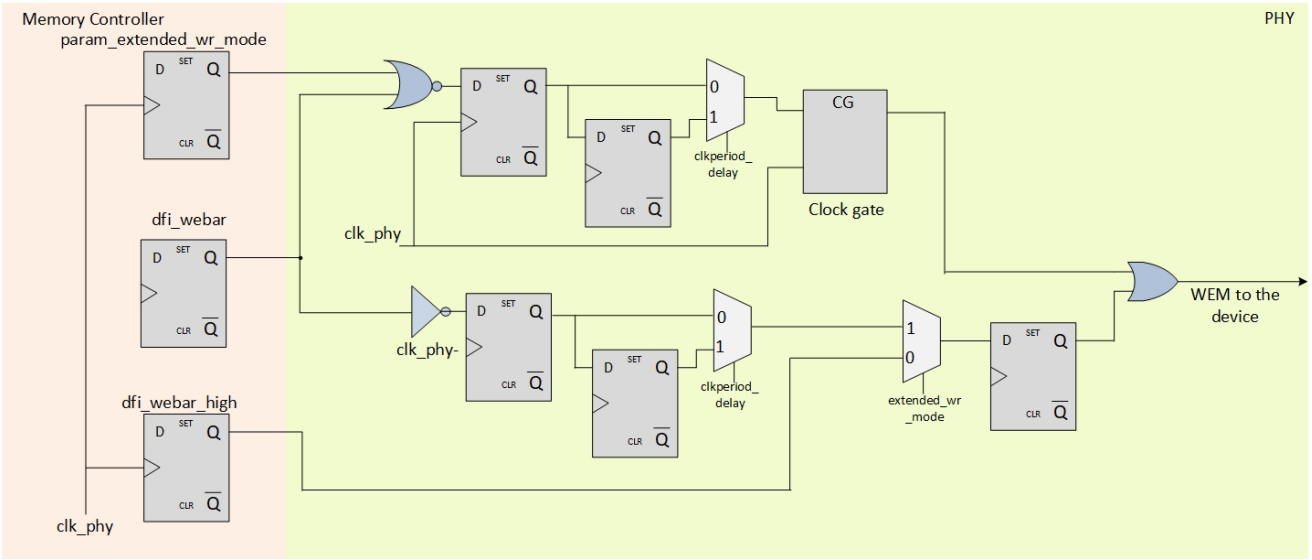
Figure 1.7. CE#, WP# generation block



1.3.7.2. WE# generation block

The following figure shows the launching of WE# signal to the device.

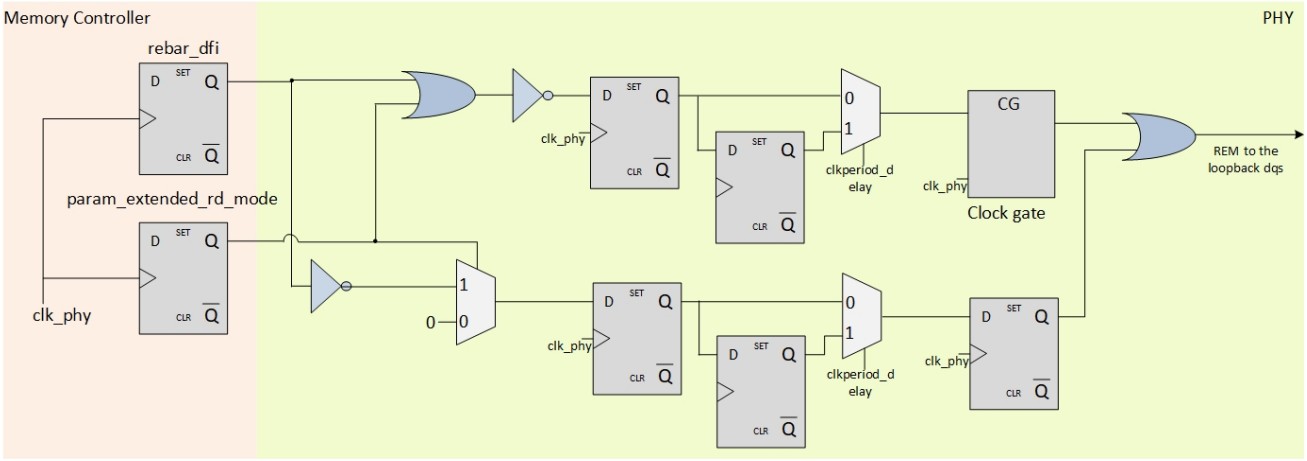
Figure 1.8. WE# generation block



1.3.7.3. RE# generation block

The following figure shows the launching of RE# signal to the device.

Figure 1.9. RE# generation block



1.3.8. Tuning for SD/eMMC interface

In SDR104 and HS200 (and in HS400 for CMD line only), the FIFO is used for DATA/CMD synchronization. The logic allows calibrating the sampling point at tuning sequence and sampling the DATA/CMD window in the center of the valid window.

1.4. Requirements

1.4.1. Clocking

1.4.1.1. Clocking Definitions

The Cadence Design IP DLL PHY is constructed of data slices. A data slice is created using the following clock definitions:

- *clk_phy*: This is the main clock used by the Cadence Design IP DLL PHY. The frequency of this clock need to be complaint with SD/eMMC device work mode.The maximum frequency for this clock is 200MHz. For SD/eMMC 1:1 clock ratio is supported for the clock frequency between PHY clock (clk_phy) and Memory device interface clock for SD/eMMC - minimum clock frequency in this case is 50MHz. For lower frequencies the oversampling is required ie. clk_phy must be 4 times higher than the SD/eMMC clock period for DDR work modes and 2 times higher for SDR work modes.
- *reg_pclk*: APB port clock signal ie. clock for register interface. Asynchronous to all other clocks in the IP.
- *mem_dqs*: This is the read path DQS clock which is driven by the flash devices to the ASIC.
- *clk_lpbk*: This is the read path clock which is driven by the loopbacked webar_opad signal which play roles of the SDCLK for SD/eMMC modes.
- *delayed_dqs*: This is the DLL-delayed and gated version of *read_mem_dqs* on the read path.
- *delayed_dqs_cmd*: This is the DLL-delayed and gated version of *mem_dqs* or *clk_lpbk* (depending on settings) on the read path for command line.
- *clk_wr*: This is the DLL-delayed version of *clk_phy* on the write data path.

1.4.1.2. PLL Requirements

The Cadence Design IP DLL PHY requires a clock source capable of running at the same speed as the device cycle time the user intends. For the highest speed a PLL capable of generating 200 MHz will be required. To reduce jitter from clock

distribution, the PLL should be located very close to the Cadence Design IP DLL PHY. The Cadence Design IP DLL PHY utilizes this clean reference clock as the DLL reference clock as well as all the critical off-chip timing paths. The Cadence Design IP DLL PHY utilizes a “normal” system clock that can be distributed across the chip as the interface to the DFI-like boundary flip-flops.

The clock connected to the PHY must have 50% input duty cycle.

1.4.2. Slave DLL Update Requirements

Since the PHY will periodically need to update the slave DLLs, the DFI update signal dfi_ctrlupd_req will be used as a signal to the PHY for this purpose. When the controller asserts this signal, the Cadence Design IP DLL PHY will update the slave DLLs and reset the FIFOs in the read path. Therefore, it is expected that the controller will assert this signal regularly during normal operation to compensate for any voltage/temperature drifts. A slave DLL update requires dfi_ctrlupd_req to be kept high for 8 to 16 clock cycles.

2. Special Function Registers

2.1. DLL PHY registers. (0x2000)

2.1.1. phy_dq_timing_reg (0x2000)

Description: This register controls the DQ related timing.

Table 2.1. phy_dq_timing_reg

Bits	SW	Name	Description	Reset
31	R/W	io_mask_always_on	Defines if the IO mask for DATA/CMD is always enabled. • 0 = disable ie. start/end defines the IO mask functionality. Recommended setting for SD/eMMC controller. • 1 = IO mask is always ON.	1'h1
30	R	RSVD	Reserved	1'h0
29:27	R/W	io_mask_end	Adjusts the ending point of the DQ/CMD pad input mask enable. Defines the delay after dfi_wrdata_en/dfi_wrcmd_en goes high when the mask is disabled (data/cmd are blocked and 1'b1 are passed to PHY).	3'h0
26:24	R/W	io_mask_start	Adjusts the starting point of the DQ/CMD pad input mask enable. Defines the delay after dfi_wrdata_en/dfi_wrcmd_en goes low when the mask is enabled (data/cmd are passed to PHY).	3'h0
23:17	R	RSVD	Reserved	7'h0
16	R/W	data_clkperiod_delay	Defines additional latency on the write datapath. It also adds a clock cycle delay for the data OE path which is equivalent of adding 2 to the data_select_oe_end and data_select_oe_start.	1'h0
15:12	R/W	data_select_tsel_start	Defines the DQ pad dynamic termination select enable time. Larger values add greater delay to when tsel turns on. Each bit changes the output enable time by a 1/2 cycle resolution.	4'h0
11:8	R/W	data_select_tsel_end	Defines the DQ pad dynamic termination select disable time. Larger values increase the delay to when tsel turns off. Each bit changes the output enable time by a 1/2 cycle resolution.	4'h0
7	R	RSVD	Reserved	1'h0
6:4	R/W	data_select_oe_start	Adjusts the starting point of the DQ pad output enable window. Lower numbers pull the rising edge earlier in time and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/2 cycle resolution.	3'h0
3	R	RSVD	Reserved	1'h0
2:0	R/W	data_select_oe_end	Adjusts the ending point of the DQ pad output enable window. Lower numbers pull the falling edge earlier in time and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/2 cycle resolution.	3'h2

2.1.2. phy_dqs_timing_reg (0x2004)

Description: This register controls the DQS related timing.

Table 2.2. phy_dqs_timing_reg

Bits	SW	Name	Description	Reset
31:23	R	RSVD	Reserved	9'h0
22	R/W	use_ext_lpbk_dqs	This bit is used in conjunction with bits 21 and 20 to control how read data is sampled by the PHY This bit is only valid when bit 20 = '1' and bit 21 = '1' and selects if the internally generated clock source should be looped back within the rebar pad (The PHY will then sample the read data using mem_rebar_ipad), or if the integrator should take the signal from the rebar pin, loop it back on the board back to the PHY using the lpbk_dqs pin. The PHY will then sample the read data using lpbk_dqs • 0 - use internal lpbk_dqs (mem_rebar_ipad) for data capture. • 1 - use external lpbk_dqs (lpbk_dqs connected to the lpbk_dqs_IO PAD) for data capture. Mandatory setting for SD/eMMC	1'h0
21	R/W	use_lpbk_dqs	This bit is used in conjunction with bits 22 and 20 to control how read data is sampled by the PHY. This bit is only valid when bit 20 is set to '1', meaning it is only relevant when read data is not being sampled by DQS from the memory device. If bit 20 is set to '0' this bit should also be set to '0'. When using the PHY with the SD/eMMC controller, it is recommended that this bit is set. This bit selects which internal source will be used by the PHY to sample the read data. This is internally generated and is passed out of the PHY via rebar_opad If bit 22 of this register is '0', then it will be internally looped back within the rebar pad. If bit 22 of this register is '1', the rebar_opad will be passed through the rebar pad and it will be the responsibility of the integrator to loop that back to the PHY into the lpbk_dqs pin. Refer to section 2 "Read Sampling lpbk_dqs". . • 0 - Use phony DQS for data capture. • 1 - Use lpbk_dqs for data capture. Mandatory setting for SD/eMMC.	1'h0
20	R/W	use_phony_dqs	This bit is used in conjunction with bits 22 and 21 to control how read data is sampled by the PHY. This bit selects whether the read data sent by the memory device will be sampled by DQS supplied by the memory device, or by a signal locally generated within the PHY. • 0 - Use DQS from device for data capture. • 1 - Use internally generated DQS for data capture.	1'h1

Bits	SW	Name	Description	Reset
19	R/W	use_phony_dqs_cmd	Bit to choose phony DQS (or lpbk_dqs) from the control slice logic or DQS from the device to capture command data for reads. • 0 - Use DQS from device for command data capture. • 1 - Use phony DQS or lpbk_dqs for command data capture.	1'h1
18:17	R	RSVD	Reserved	2'h0
16	R/W	phony_dqs_sel	If this bit is cleared the phony_dqs is synchronous with rising edge of the clk_phy before sending to the entry flops. If this bit is set high the phony_dqs is synchronous with falling edge of clk_phy before sending to the entry flops.	1'h0
15:12	R/W	dqs_select_tsel_start	Defines the DQ pad dynamic termination select enable time. Larger values add greater delay to when tsel turns on. Each bit changes the output enable time by a 1/2 cycle resolution.	4'h0
11:8	R/W	dqs_select_tsel_end	Defines the DQ pad dynamic termination select disable time. Larger values increase the delay to when tsel turns off. Each bit changes the output enable time by a 1/2 cycle resolution.	4'h0
7:0	R	RSVD	Reserved	8'h0

2.1.3. phy_gate_lpbk_ctrl_reg (0x2008)

Description: This register controls the gate and loopback control related timing.

Table 2.3. phy_gate_lpbk_ctrl_reg

Bits	SW	Name	Description	Reset
31	R/W	sync_method	Defines the method of transfering the data from DQS domain flops to the clk_phy clock domain. • if set low the read pointer advances based upon a programmable delay of the dfi_rddata_en pulse from the DFI interface. • if set high the read pointer advances based upon a programmable delay of the empty signal. Recommended setting for SD/eMMC controller.	1'h0
30:24	R	RSVD	Reserved	7'h0
23:19	R/W	rd_del_sel	Defines the read data delay. Holds the number of cycles to delay the dfi_rddata_en signal prior to enabling the read FIFO. After this delay, the read pointers begin incrementing the read FIFO. If 'sync_method' is set high the value of this field must take into account the synchronization time of the pointers in the entry FIFO (adding three clock cycles should be sufficient).	5'h1b
18	R/W	underrun_suppress	This field turns off the generation of the underrun signal when 'sync_method' is set high. For Cadence SD/eMMC controller this field need to be set high.	1'h0
17	R	RSVD	Reserved	1'h0

Bits	SW	Name	Description	Reset
16	R/W	rd_del_sel_empty	Defines the read data delay for the empty signal generated based on the incoming DQS strobes. For zero delay the data are passed from entry flops to the iodatain* flops one clock cycle after the !empty signals is asserted. Normally the zero value of this field is sufficient as the signal is generated based on the gray pointer synchronized with two stage synchronizer on clk_phy clock domain which gives minimum two clock cycle path from entry flop to the iodatain flop. Increasing the value of this field delays the moment of passing the data from entry flops to the iodatain flops. Increased value gives even more time to propagate the data but the bigger value the bigger probability to overflow the FIFO. Recommended value is zero.	1'h0
15:13	R/W	lpbk_err_check_timing	Sets the cycle delay between the LFSR and loopback error check logic to ensure that the LFSR sourced data and data being looped back arrive at the same clock cycle for comparison. This value is related to the rd_del_sel field, and is equal to 7 - rd_del_sel.	3'h0
12	R/W	lpbk_fail_muxsel	Selects data output type for phy_obs_reg_0[23:8]. • 0 = Return the expected data. • 1 = Return the actual data.	1'h0
11:10	R/W	loopback_control	Loopback control. • 0 = Normal Operation Mode. • 1 = lpbk_start; Enables loopback write mode. • 2 = lpbk_stop; Stop loopback to check error register. • 3 = clear; Clear loopback registers.	2'h0
9	R	RSVD	Reserved	1'h0
8	R/W	lpbk_en	Controls internal write multiplexer. • 0 = Normal Operation. • 1 = Enable loopback.	1'h0
7	R	RSVD	Reserved	1'h0
6	R/W	gate_cfg_always_on	This parameter cause the gate to be always on. Recommended setting for SD/eMMC controller is 1.	1'h0
5:4	R/W	gate_cfg_close	Normally the gate is closing when all bits of dfi_cebar are high or when dfi_rd_pre_post_amble and rebar_dfi are high. This parameter allows to extend the closing of the DQS gate. Recommended value is zero.	2'h0
3:0	R/W	gate_cfg	Coarse adjust of gate open time. This value is the number of cycles to delay the dfi_rddata_en signal prior to opening the gate in full cycle increments. Decreasing this value pulls the gate earlier in time. This field should be programmed such that the gate signal lands in the valid DQS gate window.	4'h0

2.1.4. phy_dll_master_ctrl_reg (0x200c)

Description: This register holds the control for the Master DLL logic.

Table 2.4. phy_dll_master_ctrl_reg

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23	R/W	param_dll_bypass_mode	DLL bypass mode control. Controls the bypass mode of the master and slave DLLs. The param_dll_bypass_mode is intended to be used only for debug. • 0 - Normal operational mode. DLL functioning in normal mode of operation where the slave delay line settings are used as fractional delay of the master delay line encoder reading of the number of delays in one cycle. • 1 - Bypass mode on. Delays are defined in phy_dll_slave_ctrl_reg. Master DLL is disabled with only 1 delay element in its delay line. The slave slave delay lines decode delays in absolute delay elements rather than as fractional delays. The dll_lock field (bit [0]) of the phy_dll_obs_reg_0 parameter will be forced high.	1'h1
22:20	R/W	param_phase_detect_sel	Selects the number of delay elements to be inserted between the phase detect flip-flops. Defaults to 0x0 although the recommended value is 2 elements but if a lock condition is not detected, the user should increase the number of delay elements. • 'b000 - One delay element. • 'b001 - Two delay element. • 'b010 - Three delay element. • 'b011 - Four delay element. • 'b100 - Five delay element. • 'b101 - Six delay element. • 'b110 - Seven delay element. • 'b111 - Eight delay element.	3'h0
19	R	RSVD	Reserved	1'h0
18:16	R/W	param_dll_lock_num	Holds the number of consecutive increment or decrement indications that will trigger an unlock condition and increment the dll_unlock_cnt field (bits [7:3]) and either the lock_dec_dbg (bits [23:16]) or lock_inc_dbg (bits [31:24]) fields of the phy_dll_obs_reg_0 parameter.	3'h0
15:8	R	RSVD	Reserved	8'h0
7:0	R/W	param_dll_start_point	This value is the initial delay value for the DLL. This value is also used as the increment value if the initial value is less than a half-clock cycle. This field should be set such that it is not greater than 7/8ths of a clock period given the worst case element delay. For example, if the frequency is 200MHz (5ns cycle time) with a worst case element 80ps delay, this field should be set to = 5 * (7/8) / .080 = 54 elements. This calculation helps determine the start point which achieves the fastest lock. However, a small value such as 0x04 may be used instead to ensure that the DLL does not lock on a harmonic. Note that with a small value like this, the initial lock time will be longer. Value smaller than 0x04 may cause no lock by DLL.	8'h0

2.1.5. phy_dll_slave_ctrl_reg (0x2010)

Description: This register holds the control for the slave DLL logic.

Table 2.5. phy_dll_slave_ctrl_reg

Bits	SW	Name	Description	Reset
31:24	R/W	read_dqs_cmd_delay	Controls the read command DQS delay which adjusts the timing in 1/256th of the clock period when in normal DLL locked mode. In bypass mode, this field directly programs the number of delay elements.	8'h0
23:16	R	RSVD	Reserved	8'h0
15:8	R/W	clk_wr_delay	Controls the clk_wr delay line which adjusts the write DQ and CMD bit timing in 1/256th steps of the clock period in normal DLL locked mode. In bypass mode, this field directly programs the number of delay elements.	8'h0
7:0	R/W	read_dqs_delay	Controls the read DQS delay which adjusts the timing in 1/256th of the clock period when in normal DLL locked mode. In bypass mode, this field directly programs the number of delay elements.	8'h0

2.1.6. phy_ie_timing_reg (0x2014)

Description: This register controls the DQS related timing.

Table 2.6. phy_ie_timing_reg

Bits	SW	Name	Description	Reset
31:21	R	RSVD	Reserved	11'h0
20	R/W	ie_always_on	Forces the input enable(s) to be on always.	1'h1
19	R	RSVD	Reserved	1'h0
18:16	R/W	dq_ie_start	Define the start position for the DQ input enable.	3'h0
15	R	RSVD	Reserved	1'h0
14:12	R/W	dq_ie_stop	Define the stop position for the DQ input enable.	3'h0
11	R	RSVD	Reserved	1'h0
10:8	R/W	dqs_ie_start	Define the start position for the DQS input enable.	3'h0
7	R	RSVD	Reserved	1'h0
6:4	R/W	dqs_ie_stop	Define the stop position for the DQS input enable.	3'h0
3:0	R/W	rddata_en_ie_dly	Specifies the number of clocks of delay for the dfi_rddata_en signal to line it up with the true (normal) DFI read data position. The MC must deliver an early version of the read data enable to allow time for the input pads to turn on and this field allows the PHY to create the original timing.	4'h0

2.1.7. phy_obs_reg_0 (0x2018)

Description: This register holds the following observable points in the PHY.

Table 2.7. phy_obs_reg_0

Bits	SW	Name	Description	Reset
31:28	R	RSVD	Reserved	4'h0
27	R	dqs_cmd_overflow	CMD Status signal to indicate that the logic gate was closed too late ie. the number of DQS strobes exceed the capacity of the entry FIFO. It indicates that rd_del_sel signal value is too high and dfi_rddata are corrupted. It is possible that overflow status is asserted with underrun status - in such case the overflow takes the precedence. The dll_rst_n or rst_n clears this flag.	1'h0
26	R	dqs_cmd_underrun	CMD Status signal to indicate that the logic gate had to be forced closed. It indicates that either the DQS strobe did not appear during read or rd_del_sel signal value is too low and dfi_rddata are corrupted. The dll_rst_n or rst_n clears this flag.	1'h0
25	R	dqs_overflow	Status signal to indicate that the logic gate was closed too late ie. the number of DQS strobes exceed the capacity of the entry FIFO. It indicates that rd_del_sel signal value is too high and dfi_rddata are corrupted. It is possible that overflow status is asserted with underrun status - in such case the overflow takes the precedence. The dll_rst_n or rst_n clears this flag.	1'h0
24	R	dqs_underrun	Status signal to indicate that the logic gate had to be forced closed. It indicates that either the DQS strobe did not appear during read or rd_del_sel signal value is too low and dfi_rddata are corrupted. The dll_rst_n or rst_n clears this flag.	1'h0
23:8	R	lpbk_dq_data	If errors are encountered in loopback test this field reports the actual data or the expected data, depending on the setting of the phy_gate_lpbk_ctrl_reg [12] parameter bit. This field is not clear by the clear state of the loopback. If there are no errors in loopback test the value is zero (or value from previous state).	16'h0
7	R	lpbk_cmd_data	If errors are encountered in loopback test this field reports the actual data or the expected data of the CMD line, depending on the setting of the phy_gate_lpbk_ctrl_reg [12] parameter bit. This field is not clear by the clear state of the loopback. If there are no errors in loopback test the value is zero (or value from previous state).	1'h0
6:2	R	RSVD	Reserved	5'h0
1:0	R	lpbk_status	Loopback Status • Bit0 - lpbk start; Defines the status of the loopback mode. 0 = Not in loopback mode; 1 = In loopback mode. • Bit1 - lpbk status; Defines the status of the loopback mode. 0 = Last Loopback test had no errors; 1 = Last loopback test contained data errors.	2'h0

2.1.8. phy_dll_obs_reg_0 (0x201c)

Description: This register holds the following observable points in the PHY.

Table 2.8. phy_dll_obs_reg_0

Bits	SW	Name	Description	Reset
31:24	R	lock_inc_dbg	Holds the state of the cumulative dll_lock_inc register when the dll_unlock_cnt field(bits [7:3]) of this parameter was triggered to increment or was last saturated at a value of 0x1f.	8'h0
23:16	R	lock_dec_dbg	Holds the state of the cumulative dll_lock_dec register when the dll_unlock_cnt field(bits [7:3]) of this parameter was triggered to decrement or was last saturated at a value of 0x1f.	8'h0
15:8	R	dll_lock_value	Reports the number of delay elements that the DLL has determined for lock in either full clock or half clock mode. In full clock mode, this value equals the number of delay elements in one cycle. In half clock mode, this value equals the number of delay elements in one half clock cycle. In saturation mode, this value equals the maximum number of delay elements. The slaves use this value to set up their delays for the clk_wr and read DQS signals. This value is valid only when locking mechanism is done.	8'h0
7:3	R	dll_unlock_cnt	Reports the number of times that the master DLL consecutive increment or decrement value programmed into the param_dll_lock_num field (bits [18:16]) of the phy_dll_master_ctrl_reg register has been triggered. The dll_unlock_cnt will saturate at a value of 0x1f. Asserting the dll_rst_n signal will reset this counter to 0.	5'h0
2:1	R	dll_locked_mode	Indicates status of DLL. Defines the mode in which the DLL has achieved the lock. • 'b00 - Full clock mode. The master delay line was long enough to lock on one full clock cycle of delay. In this mode, the dll_lock_value field (bits [15:8]) of this parameter indicates the number of delays in full clock cycles. • 'b01 - Reserved. • 'b10 - Half clock mode. The master delay line was not long enough to lock one full cycle of delay but could lock on a half-cycle of delay. In this mode, the dll_lock_value field (bits [15:8]) of this parameter indicates the number of delays in one half clock cycles. • 'b11 - Saturation mode. The master delay line was not long enough to lock on a full or a half-clock cycle. In this mode, the encoder value is fixed at the maximum delay line setting and the master DLL will be disabled. The slave delay lines continue to use the fractional delays based upon the fixed saturation value of the delay line.	2'h0
0	R	dll_lock	Indicates status of DLL. It indicates the DLL locking when the DLL lock logic found (not inc AND not dec) OR (an inc then dec) OR (a dec then inc). When param_dll_start_point is set smaller than half clock period the first found (a dec then inc) isn't the really DLL locking point but dll_lock is asserted. • 0 - DLL has not locked.	1'h0

Bits	SW	Name	Description	Reset
			• 1 - DLL is locked.	

2.1.9. phy_dll_obs_reg_1 (0x2020)

Description: This register holds the following observable points in the PHY.

Table 2.9. phy_dll_obs_reg_1

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:16	R	decoder_out_wr	Holds the encoded value for the clk_wr delay line for this slice.	8'h0
15:8	R	decoder_out_rd_cmd	Holds the encoded value for the CMD read delay line for this slice.	8'h0
7:0	R	decoder_out_rd	Holds the encoded value for the read delay line for this slice.	8'h0

2.1.10. phy_static_togg_reg (0x2028)

Description: This register controls the static aging feature of the PHY.

Table 2.10. phy_static_togg_reg

Bits	SW	Name	Description	Reset
31:23	R	RSVD	Reserved	9'h0
22:20	R/W	static_togg_enable	Control to enable the toggle signal during static activity. When low the feature is disabled. • bit 0 - master delay line enable. • bit 1 - read path delay line enable. • bit 2 - write path delay line enable.	3'h0
19:17	R	RSVD	Reserved	3'h0
16	R/W	static_togg_global_enable	Global control to enable the toggle signal during static activity.	1'h0
15:0	R/W	static_tog_clk_div	Clock divider to create toggle signal.	16'h0

2.1.11. phy_wr_deskew_pd_ctrl_0_reg (0x2034)

Description: This register holds the values of phase detect block for each DQ bit on the write path.

Table 2.11. phy_wr_deskew_pd_ctrl_0_reg

Bits	SW	Name	Description	Reset
31:7	R	RSVD	Reserved	25'h0
6	R/W	dq_sw_dq_phase_bypass	• 'b0 - Use phase detect circuit to determine the half_cycle_shift. • 'b1 - Use the clk_wr_delay delay line setting to determine the half_cycle_shift. A delay line setting of 0x00-0x7f	1'h0

Bits	SW	Name	Description	Reset
			means half_cycle_shift = 0 and a delay line setting of 0x80-0xff means half_cycle_shift = 1.	
5	R/W	dq_en_sw_half_cycle	Enables the software half cycle shift. This determines if write data is transferred to the clk_wr domain on the positive or negative edge of the PHY clock. This field is valid when dq_sw_dq_phase_bypass is low. • 'b0 - Hardware automatically controls any shifting needed for the write level delay line. • 'b1 - The setting in the dq0_sw_half_cycle_shift field this reg defines the shift. Note: If the user chooses to control the half cycle shift manually, it is important that the dq_sw_half_cycle_shift field (bit [4]) of the phy_wr_deskew_pd_ctrl_reg parameter be cleared to 'b0 if the delay is less than a 1/2 cycle and set to 'b1 if the delay is greater than a 1/2 cycle. It is recommended to allow the hardware to control this automatically.	1'h0
4	R/W	dq_sw_half_cycle_shift	• 'b0 - No effect. • 'b1 - Adds a half clock delay to the write data path.	1'h0
3	R	RSVD	Reserved	1'h0
2:0	R/W	dq_phase_detect_sel	DLL Phase Detect Selector for DQ generation to handle the clock domain crossing between the clock and clk_wr signal. Selects the number of delay elements to be inserted between the phase detect flip-flops. Defaults to 0x0. • 'b000 - One delay element. • 'b001 - Two delay element. • 'b010 - Three delay element. • 'b011 - Four delay element. • 'b100 - Five delay element. • 'b101 - Six delay element. • 'b110 - Seven delay element. • 'b111 - Eight delay element.	3'h0

2.1.12. phy_version_reg (0x2070)

Description: This register contains release identification number.

Table 2.12. phy_version_reg

Bits	SW	Name	Description	Reset
31:16	R	combo_phy_magic_number	Magic number.	16'h6182
15:8	R	phy_fix	Fixed number (minor revision number).	8'h1
7:0	R	phy_rev	PHY revision number.	8'h7

2.1.13. phy_features_reg (0x2074)

Description: This register shows available hardware features.

Table 2.13. phy_features_reg

Bits	SW	Name	Description	Reset
31:16	R	RSVD	Reserved	16'h0
15	R	asf_sup	Support for Automotive Safety Feature.	1'h0
14	R	pll_sup	Support for PLL.	1'h0
13	R	jtag_sup	Support for JTAG muxes.	1'h0
12	R	ext_lpbk_dqs	Support for external LPBK_DQS io pad.	1'h1
11	R	reg_intf	SFR interface type. This is an encoded value. • 0 - DFI. • 1 - APB.	1'h1
10	R	per_bit_deskew	Support for per-bit deskew.	1'h0
9	R	dfi_clock_ratio	Support for clock ratio on DFI interface. This is an encoded value. • 0 - 1:1 • 1 - 1:2	1'h0
8	R	aging	Support for aging in delay lines.	1'h1
7	R	dll_tap_num	Number of taps in delay line. This is an encoded value. • 0 - 128. • 1 - 256.	1'h1
6:5	R	bank_num	Maximum number of banks supported by hardware. This is an encoded value. • 0 - One bank. • 1 - Two banks. • 2 - Four banks. • 3 - Eight banks.	2'h0
4	R	sd_emmc	Support for SD/eMMC.	1'h1
3	R	xspi	Support for XSPI.	1'h0
2	R	sdr_16bit	Support for 16bit in ONFI SDR work mode.	1'h0
1	R	onfi_41	Support for ONFI4.1 - NAND Flash.	1'h0
0	R	onfi_40	Support for ONFI4.0 - NAND Flash.	1'h0

2.2. Control Timing Block registers (0x2080)

2.2.1. phy_ctrl_reg (0x2080)

Description: This register handles the global control settings for the PHY.

Table 2.14. phy_ctrl_reg

Bits	SW	Name	Description	Reset
31:22	R	RSVD	Reserved	10'h0

Bits	SW	Name	Description	Reset
21	R/W	pu_pd_polarity	Defines the polarity of the ALE port that in SD works as pull-up/pull-down signal for bit 2 of the DATA. • 0 - ALE port is a copy of dfi_ale. • 1 - ALE port is inverted version of dfi_ale.	1'h0
20:9	R	RSVD	Reserved	12'h0
8:4	R/W	phony_dqs_timing	The timing of assertion of phony DQS to the data slices. If the extended_read_mode is disabled the value should be zero. If the extended_read_mode is enabled the value should match the width of the rebar pulse in terms of clock PHY clock cycles reduced by 1. e.g. if rebar pulse width is 4 clock cycles the value of this field should be 3.	5'h18
3:1	R	RSVD	Reserved	3'h0
0	R/W	ctrl_clkperiod_delay	Defines additional latency on the control signals WE/RE/CE/WP.	1'h0

2.2.2. phy_tsel_reg (0x2084)

Description: This register handles the global control settings for the termination selects for reads.

Table 2.15. phy_tsel_reg

Bits	SW	Name	Description	Reset
31:24	R	RSVD	Reserved	8'h0
23:20	R/W	tsel_off_value_data	Termination select off value for the data.	4'h0
19:16	R/W	tsel_rd_value_data	Termination select read value for the data.	4'h0
15:12	R/W	tsel_off_value_dqs	Termination select off value for the data strobe.	4'h0
11:8	R/W	tsel_rd_value_dqs	Termination select read value for the data strobe.	4'h0
7:0	R	RSVD	Reserved	8'h0

2.2.3. phy_gpio_ctrl_0 (0x2088)

Description: This register is a general purpose register. The [31:0] vector is brought to the PHY I/Os. User may choose to use these pins to control any static settings that may be required for the connected I/O pads.

Table 2.16. phy_gpio_ctrl_0

Bits	SW	Name	Description	Reset
31:0	R/W	phy_gpio_ctrl_0_value	General purpose register field. The [31:0] vector is brought to the PHY I/Os. User may choose to use these pins to control any static settings that may be required for the connected I/O pads.	32'h0

2.2.4. phy_gpio_ctrl_1 (0x208c)

Description: This register is a general purpose register. The [31:0] vector is brought to the PHY I/Os. User may choose to use these pins to control any static settings that may be required for the connected IO pads.

Table 2.17. phy_gpio_ctrl_1

Bits	SW	Name	Description	Reset
31:0	R/W	phy_gpio_ctrl_1_value	General purpose register field. The [31:0] vector is brought to the PHY IOs. User may choose to use these pins to control any static settings that may be required for the connected IO pads.	32'h0

2.2.5. phy_gpio_status_0 (0x2090)

Description: This register is a general purpose register. A [31:0] vector is brought from the PHY IOs to this register. User may choose to use this as a status register.

Table 2.18. phy_gpio_status_0

Bits	SW	Name	Description	Reset
31:0	R	phy_gpio_status_0_value	This register is a general purpose register. A [31:0] vector is brought from the PHY IOs to this register. User may choose to use this as a status register.	32'h0

2.2.6. phy_gpio_status_1 (0x2094)

Description: This register is a general purpose register. A [31:0] vector is brought from the PHY IOs to this register. User may choose to use this as a status register.

Table 2.19. phy_gpio_status_1

Bits	SW	Name	Description	Reset
31:0	R	phy_gpio_status_1_value	This register is a general purpose register. A [31:0] vector is brought from the PHY I/Os to this register. User may choose to use this as a status register.	32'h0

3. Testability

3.1. DFT implementation

The Cadence Design IP DLL PHY is fully scanable. The table below describes all signals related with the scan. During scan insertion listed signals should be assigned accordingly to their funcion.

Table 3.1. Scan Interface

Signal Name	Description
clk_phy	Clock input to PHY used in scan.
rst_n, reg_rst_n	Active low reset signals used in scan.
scanmode	Active high scan mode input.
scan_en	Active high shift enable input.
scan_atspeed_tmode	Scan test mode - set high for at-speed mode to run scan at operational frequency; set low for low frequency scan mode. For normal functional mode of the phy this signal must be set high.

The PHY supports transition fault testing of the digital logic at-speed with a second pass at the lower frequency to target the faults in the configurable delay lines. Additional test mode has been introduced in the DLL PHY scan implementation for this purpose - this mode is selected with the scan_atspeed_tmode signal.

3.1.1. Scan Mode at operational frequency

This mode is selected by the scanmode port being high and the scan_atspeed_tmode being high. In this mode the delay lines are bypassed or gated which allows to run the test at operational frequency of the DLL PHY.

3.1.2. Scan Mode at lower frequency

This mode is selected by the scanmode port being high and the scan_atspeed_tmode being low. In this mode the delay value set on configurable delay lines is driven from the register and thus may achieve maximum values during scan. This mode must be run at lower clock frequency - the period should be higher than the maximum delay value of the delay line.

3.2. Loopback for Internal Testing (BIST)

This chapter consists of the following topics:

- Overview
- Loopback Testing Setup
- Starting and Stopping Loopback Testing
- Loopback Setup for ATE

3.2.1. Overview

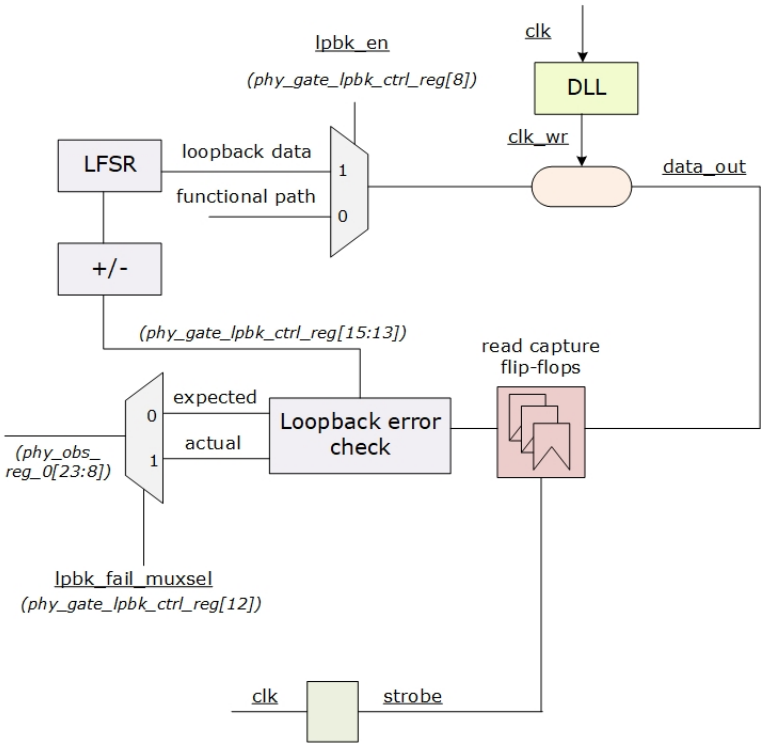
The Cadence Design IP DLL PHY contains logic that allows for at-speed testing and data eye training without adding additional test multiplexers in critical timing paths in the design. The loopback testing mechanism includes a vector

generation unit (Linear Feedback Shift Register, LFSR), which creates patterns for the write data path and then tracks the data back through the read data path.

Internal loopback testing can be used during production to verify that critical logic such as the DLL, write path, and read path are functioning correctly. External loopback can be used to test the at-speed connectivity of the I/O pads. External loopback will send the data and DQS signals through the bidirectional pads. During external loopback, the I/O pad termination and the input and output enables are all active. This causes the output data to be routed directly back to the PHY. If external loopback is initiated in a system where the ASIC is connected to a test head, the user must maintain proper termination and shielding of all data lines connected to the DQ and DQS pins. Any reflections or crosstalk caused by the test board will affect the signal quality and may cause the external loopback to fail.

External loopback is not intended to be used on systems where the ASIC is connected to actual flash parts. The reflections caused by the unterminated flash parts will most likely cause external loopback to fail.

Figure 3.1. Loopback Logic block Diagram



3.2.2. Loopback Testing Setup

Prior to initiating loopback testing, the user should program the following parameters for loopback testing:

Table 3.2. General Loopback Setup Parameters

PHY Signal	Control Parameter	Value
gate_cfg_close	phy_gate_lpbk_ctrl_reg[5:4]	0x3
lpbk_en	phy_gate_lpbk_ctrl_reg[8]	0x1
lpbk_internal	phy_gate_lpbk_ctrl_reg[9]	0x1 - Internal, 0x0 - External

The DLL should be defined for normal mode, except that the read DQS DLL should be set to 0x0. Once these settings have been defined, the MC should drive the dfi_ctrlupd_req signal. An assertion of this signal triggers the Cadence Design IP DLL PHY to update the slave delay lines based upon the register settings.

3.2.2.1. Internal Loopback Setup

The following additional settings are required for internal loopback setup:

Table 3.3. Internal Loopback Setup Parameters

PHY Signal	Control Parameter	Value
lpbk_internal	phy_gate_lpbk_ctrl_reg[9]	0x1
rd_del_sel	phy_gate_lpbk_ctrl_reg[23:19]	0x2 (for RTL simulation). Gate simulation and silicon may require different values.
gate_cfg	phy_gate_lpbk_ctrl_reg[3:0]	0x0 (for RTL simulation). Gate simulation and silicon may require different values.

3.2.2.2. External Loopback Setup

The following additional settings are required for external loopback setup:

Table 3.4. External Loopback Setup Parameters

PHY Signal	Control Parameter	Value
lpbk_internal	phy_gate_lpbk_ctrl_reg[9]	0x0
rd_del_sel	phy_gate_lpbk_ctrl_reg[23:19]	0x2 + lpbk_diff(for RTL simulation). Gate simulation and silicon may require different values. calculate lpbk_diff = integer ((IO_Delay_Out + IO_Delay_In + .75*clkperiod) / clkperiod)
gate_cfg	phy_gate_lpbk_ctrl_reg[3:0]	gate_cfg = integer ((IO_Delay_Out + IO_Delay_In) / clkperiod)

Note

The maximum value of the 'rd_del_sel' in the loopback mode is limited to 7. For IO PADs with very large delays the clock frequency may need to be reduced for external loopback test.

3.2.2.3. 'lpbk_err_check_timing' Setup

'lpbk_err_check_timing' depends on the 'rd_del_sel' value. The lower value is set the larger delay is introduced. The following table presents the valid 'lpbk_err_check_timing' values with respect to 'rd_del_sel':

Table 3.5. 'lpbk_err_check_timing' Setup

rd_del_sel	lpbk_err_check_timing
2	5
3	4
4	3

rd_del_sel	lpbk_err_check_timing
5	2
6	1
7	0

3.2.3. Starting and Stopping Loopback Testing

Starting, stopping and clearing loopback are all controlled through the parameter field gate_lpbk_ctrl_reg[11:10].

Table 3.6. Loopback Control(gate_lpbk_ctrl_reg[11:10])

Setting	Definition
'b00	Normal operation - no loopback testing
'b01	lpbk_start - Initiate a loopback test.
'b10	lpbk_stop - Stop loopback testing
'b11	Clear - Clear the loopback status registers.

3.2.4. Loopback Setup for ATE

In the ATE environment, internal signals are controlled through the pads. To initiate a loopback, a multiplexer used for testing must be installed between the controller and the loopback-related signals. The following table lists signals that can be used to control loopback. The user is not required to control every signal.

Table 3.7. Loopback control pins for ATE

PHY Pin	Direction	Description
dfi_sdr_cycle	Input	This bit needs to be set to '0' during loopback testing.
dfi_ctrlupd_req	Input	Clears the read data pointers. Assert this signal and then clear it to clear the read data pointers.
dfi_init_complete	Output	Loopback sequence may be initiated after this signal has been asserted.
dll_rst_n	Input	DLL reset. Assert high, wait for dfi_init_complete, then begin loopback testing.
reg_pclk	Input	Clock - The rising edge of PCLK times all transfers on the APB.
reg_preset_n	Input	Reset - The APB reset signal is active LOW.
reg_paddr	Input	Address - This is the APB address bus.
reg_psel	Input	Select - It indicates that the slave device is selected and that a data transfer is required.
reg_penable	Input	Enable - This signal indicates the second and subsequent cycles of an APB transfer.
reg_pwrite	Input	Direction - This signal indicates an APB write access when HIGH and an APB read access when LOW.
reg_prdata	Output	Read Data - The selected slave drives this bus during read cycles when PWRITE is LOW.

PHY Pin	Direction	Description
reg_pwdata	Input	Write data - This bus is driven by the peripheral bus bridge unit during write cycles when PWRITE is HIGH.
reg_pready	Output	Ready - The slave uses this signal to extend an APB transfer.

4. Build Guide

The Cadence Design IP DLL PHY is built from the bottom-up process by first building an individual PHY data slice, and then instantiating slices along with the PHY top level to create the Cadence Design IP DLL PHY. As a result, there are multiple sets of build constraints that will be created.

This chapter consists of the following topics:

- Cadence Design IP DLL PHY Components
- Cadence Digital DLL
- DLL locking
- Cadence Design IP DLL PHY Slice Floorplan

4.1. Logic Requirements

All signals defined by the DFI must be driven by registers clocked on the rising edge of the controller clock. The Cadence Design IP DLL PHY handles the memory interface timing for the read datapath, write datapath and all control signals. The Cadence Design IP DLL PHY will capture the DFI control/data signals on the next rising edge before transmitting from the following flip-flop.

Timing closure at a system level consists of solving flop-to-flop timing, balancing clock trees, managing special skew and delay paths, and closing timing between the Cadence Design IP DLL PHY and the SOC memory controller.

4.2. Cadence Design IP DLL PHY Components

The Cadence Design IP DLL PHY is built from the bottom-up:

- Digital delay line
- PHY data slice
- PHY top level

Cadence recommends hardening the PHY from the bottom-up. The hierarchy levels above may be combined as long as STA is met within each block.

4.3. Cadence Digital DLL

Due to the asynchronous nature of the flash devices, the timing requirements for capturing and receiving data between the ASIC and the flash devices must be addressed in any DDR PHY design. The Cadence Design IP DLL PHY contains a circuit that, in conjunction with I/O cell circuitry, can be used to meet the timing requirements for an ASIC design. The delay compensation circuit was designed with the following features:

- Programmable read clock delay specified as a percentage of a clock cycle.
- Programmable write data delays specified as percentages of a clock cycle.
- Delay compensation circuit re-sync circuitry activated during refresh cycles to compensate for temperature and voltage drift.

- Separate delay chains for each read DQS signal from the flash devices.

The delay compensation circuitry relies on a master/slave approach. There is a master delay line which is used to determine how many delay elements constitute a complete cycle. This count is used, along with the programmable fractional delay settings, to determine the actual number of delay elements to program into the slave delay lines. The master and slave delay lines are identical. This approach allows the memory controller to observe a clock and then delay other signals a fixed percentage of that clock.

The actual delay element for the delay chains is user-selectable. The RTL code for the DLL is structured in a netlist fashion so this cell can be easily inserted into the design. This allows this delay compensation circuit implementation to be migrated to several different process technologies. Before synthesis all **HIC** (Hand Instantiated Cell) modules have to be adapted to the target technology. Inside the **HIC** module the carefully selected library cell needs to be instantiated. The cell function must comply with the logic of the module and in case of the delay element must meet timing requirement for this element. A CDNSSYNTHESIS define should be used for synthesis.

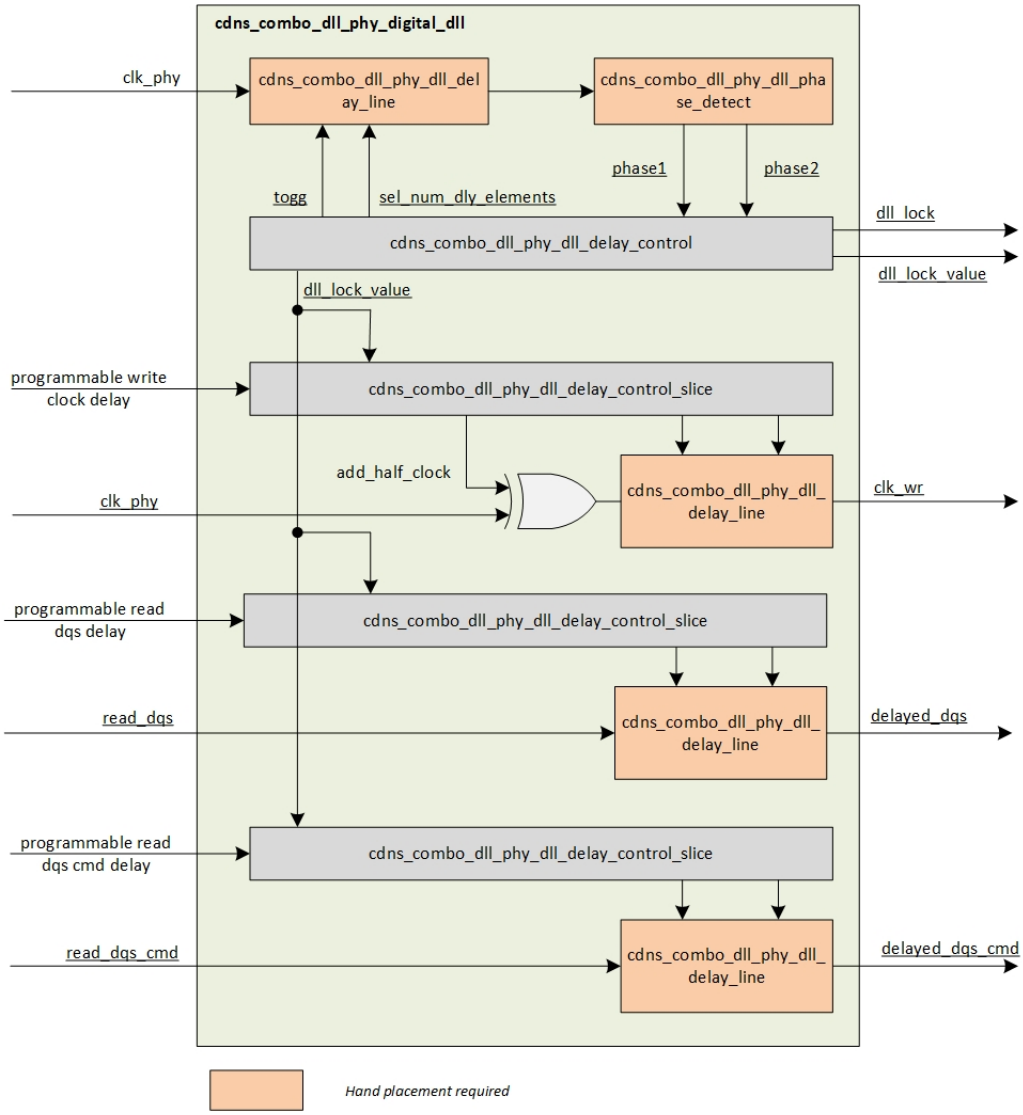
4.3.1. Delay Locked Loop Description

The design of the delay compensation circuit can be described as three steps:

- Determine the number of delay elements needed to capture an entire clock cycle.
- Determine the number of delay elements needed to delay the read DQS and write data from the programmable read and write delay parameters.
- Configure and design the delay chains.

The block diagram for the delay compensation circuit is found in the figure below.

Figure 4.1. Cadence Digital DLL block Diagram



DLL locking process begins by centering on a internal one-hot counter and a pair of flip-flops. These devices determine when the rising edge of the master clock meets the rising edge of the master clock after it is sent through its delay line. The one-hot counter (encoded to the delay select signal - sel_num_dly*) can insert or remove a delay element each cycle until the rising edges of the two signals are aligned. Once this is achieved, the number of delay elements needed to capture a clock cycle is known. This number determines the number of select lines needed in the encoder to control the delay and the maximum number of cycles that the delay compensation circuit needs for initialization.

The number of elements that are needed to capture an entire clock cycle is then converted into an unsigned integer named encoder [7:0]. This integer is used as the dividend for the read and write delay parameters. The actual delay setting for the delay lines is calculated by multiplying the encoder [7:0] integer by the parameter settings for each delay line and then dividing by 256 and rounding. These values are then encoded into a one-hot counter (and further to the delay select signal - sel_num_dly*) and updated at initialization and at every slave update interval.

The encoder [7:0] value can be used to determine if the DLL has locked on to a harmonic of the clock signal. To check this, the user must have knowledge of the approximate delay of a single delay element in the DLL at the current operating condition. Multiplying the single element delay by the encoder [7:0] value will roughly equal the period of the clock. If this product is a multiple of the clock, then the DLL has locked onto a harmonic.

4.3.2. cdns_combo_dll_phy_dll_phase_detect Block

The cdns_combo_dll_phy_dll_phase_detect block is connected to the master delay line and contains the phase detect circuitry. It is important to note that the clock to the delay line and the clocks to the phase detect modules should be driven off the same clock driver and be the same net to minimize skew.

Note : In GLS this block may produce 'X's and be a reason of simulation failure. Timing check for the first stage flops of the phase detect block can be disabled to avoid generation of the unknown values on those signals.

In operation, the phase detect flip-flops can be subjected to transitions which could promote metastability and therefore the two stage synchronizer has been instantiated in the IP. For this reason, the RTL includes a hand-instantiated cell cdnsdru_datasync_synth_ar, which is referenced in the RTL of the phase detector cdns_combo_dll_phy_phase_detect block.

Note : For synthesis a define CDNSDRU_DATASYNC_SYNTHESIS should be used along with the CDNSSYNTHESIS.

In the actual layout, place the synchronizers very close to each other and with the shortest amount of wiring between them. They should also be placed right next to the end of the delay line.

Figure 4.2. Block Diagram of the cdns_combo_dll_phy_dll_phase_detect Block

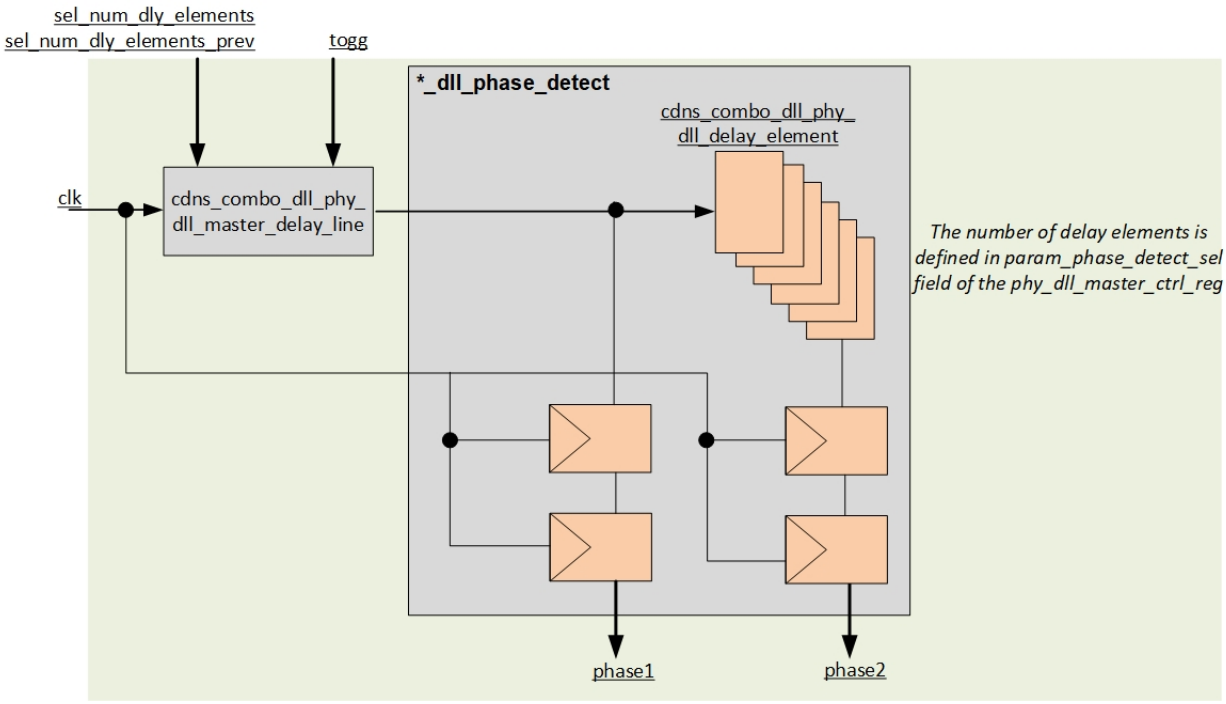


Table 4.1. Connections for the cdns_combo_dll_phy_dll_phase_detect Block

Signal	Direction	Description
clk_phy	Input	Master clock. IMPORTANT: Clock loads in this module should originate from the same clock driver to minimize skew.
phase1	Output	Phase detect signal for early clock.
phase2	Output	Phase detect signal for late clock.
sel_num_dly_elements[255:0]/ sel_num_dly_elements_prev[255:0]	Input	Delay value select lines from the cdns_combo_dll_phy_dll_delay_control block.

Signal	Direction	Description
togg[255:0]	Input	Toggle signal from the cdns_combo_dll_phy_dll_delay_control block.

4.3.3. cdns_combo_dll_phy_dll_delay_control Block

The cdns_combo_dll_phy_dll_delay_control block is responsible for locking on a frequency and distributing this lock value to the slave delay lines. Those lines use this information, along with the delay parameters, to set the delay for each individual delay line.

Figure 4.3. Block Diagram of the cdns_combo_dll_phy_dll_delay_control Block

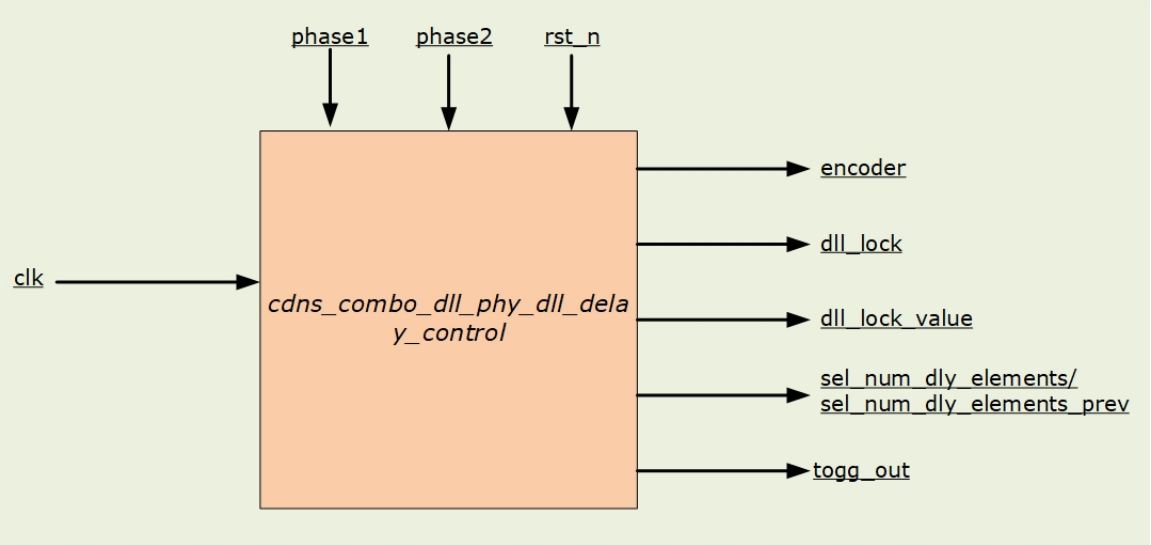


Table 4.2. Connections for the cdns_combo_dll_phy_dll_delay_control Block

Signal	Direction	Description
clk_phy	Input	Master clock. IMPORTANT: Clock loads in this module should originate from the same clock driver to minimize skew.
dll_lock	Output	Indicates if the DLL has achieved lock. 0 - Not locked, 1 - locked.
dll_lock_value[7:0]	Output	Actual value of encoder sent to the read-only parameter in the memory controller.
encoder[7:0]	Output	Encoded delay output to slave delay lines.
phase1	Input	Clock phase signal input for early clock.
phase2	Input	Clock phase signal input for late clock.
rst_n	Input	Active-low reset.
sel_num_dly_elements [255:0]	Output	Delay value select lines for master delay line.
sel_num_dly_elements_prev [255:0]	Output	Shifted version of the delay value select lines for master delay line.
togg_out [255:0]	Output	Toggle signal for master delay line.

4.3.4. cdns_combo_dll_phy_dll_delay_line Block

The cdns_combo_dll_phy_dll_delay_line block is comprised of multiple cdns_combo_dll_phy_dll_delay_element blocks. A breakdown of this block is shown in the figure below -

Figure 4.4. Block Diagram of the cdns_combo_dll_phy_dll_delay_element Block

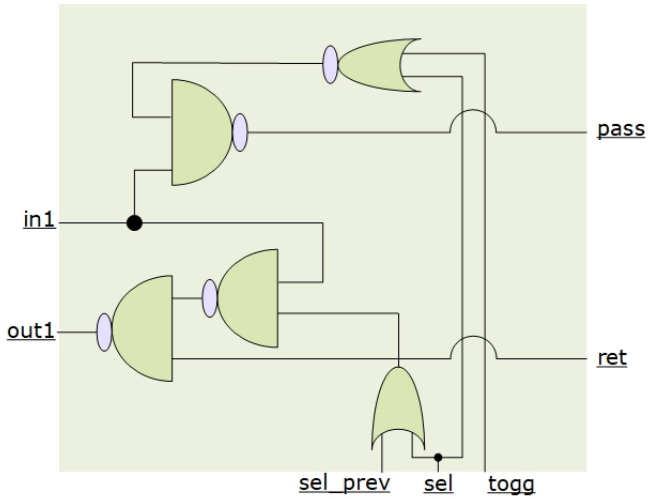


Table 4.3. Connections for the cdns_combo_dll_phy_dll_delay_element Block

Signal	Direction	Description
in1	Input	Input signal. Tied to the pass output from previous delay element.
out1	Output	Output signal. Tied to the ret signal of the previous delay element.
pass	Output	Tied to the in signal of the next delay element.
ret	Input	Return signal. Tied to the out output from the next delay element's out.
sel	Input	Affects the output signals out and pass.
sel_prev	Input	Affects the output signal out.
togg	Input	Introduced additional toggling signal for symetrical aging.

The cdns_combo_dll_phy_dll_delay_line block contains the actual delay lines for the master and slave read and write slices. Each delay line consists of identical delay elements.

Figure 4.5. Block Diagram of the cdns_combo_dll_phy_dll_delay_line Block

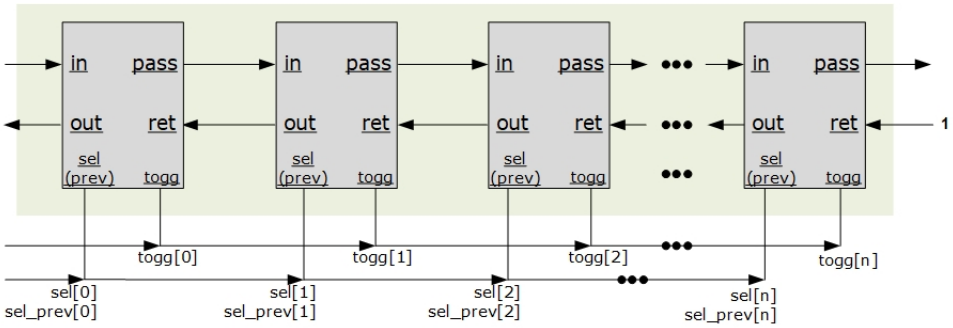


Table 4.4. Connections for the cdns_combo_dll_phy_dll_delay_line block

Signal	Direction	Description
in1	Input	Input signal to be delayed.
out1	Output	Resulting delayed signal.
sel[255:0]/sel_prev[255:0]	Input	Delay value select signals.
togg[255:0]	Input	Toggle signal for symetrical aging.

4.4. DLL locking

DLL Locking is controlled through the register interface. When the DLL is reset by asserting the dll_rst_n signal, the master DLL will perform a locking procedure starting with the programmed value in the param_dll_start_point field (bits [7:0]) of the phy_dll_master_ctrl_reg parameter and the current frequency of operation. The param_dll_start_point field should be programmed with a value that does not exceed 7/8ths of a clock period given the worst case element delay.For example, if the frequency is 400MHz (2.5ns cycle time) with a worst case element 80ps delay, this this field should be set to = 2.5 * (7/8) / .080 = 27 elements.

With this setting, the master DLL is guaranteed to correctly lock for all frequencies below 400MHz and in any process corner If the delay provided by the delay line is enough to cover a full clock cycle, the master DLL is in full clock mode. In this case, the dll_lock_value field (bits [15:8]) of the phy_dll_obs_reg_0 parameter reports the number of delay element in one full clock cycle. This number is used by the slave delay line fractional settings to determine the number of elements of delay to add to the slave delay lines. For example, if the dll_lock_value = 50 = 0x32 and the slave delay line percentage = 25% = 0x40 then the slave delay line delay value = 50 * .25 = 12.5 elements (rounded to integer).

If the frequency of operation is such that the delay line is not long enough to accommodate a full cycle of delay, the master DLL will automatically detect this situation and switch to half clock mode. In this mode, the master DLL attempts to lock when the delay in the delay line reaches a half-clock cycle. If lock is achieved in half clock mode, the slave delay lines are automatically adjusted to program a fractional delay of a fullcycle. There is no need to change the slave delay settings based upon the lock mode of the master DLL. For example, if the dll_lock_value = 50 = 0x32 and the slave delay line percentage = 25% = 0x40 then the slave delay line delay value = (50x2) * .25 = 25 elements.

If the frequency of operation is such that the delay line is not long enough to capture a half-clock cycle of delay, the master DLL will indicate lock and set the number of delays to the maximum length of the delay line. This is called saturation mode. There is no need to change the slave delay settings in this mode. The slave delay settings will be fixed at the fractional delay based upon the maximum delay of the delay line times 2. For example, if the dll_lock_value = MAX = 255 = 0xFF and the slave delay line percentage = 25% = 0x40 then the slave delay line delay value = (255x2) * .25 = 128 elements.

Note

In half clock mode or saturation mode, if the slave delay line is programmed to more than 50% delay, the input clock to the slave delay line is inverted and the percentage delay used by the slave delay line is reduced by 50%. For example, if the dll_lock_value = 50 = 0x32, and the slave delay line percentage = 85% = 0xD9, then the slave delay line delay value = (50 x 2) * (.85 - .50) = 43 elements + one half-clock. This feature does not apply to the read DQS delay line since the delay needed for the read dqs is always between 0 and 50%.

4.4.1. Lock Conditions

Once locked, the master DLL will remain locked until it is reset. The master DLL will achieve an initial lock under the following conditions:

1. The param_dll_bypass_mode field (bit [23]) of the phy_dll_master_ctrl_reg parameter is set to 1.

2. The master DLL phase detect block indicates that the current number of delay elements in the delay line is such that the edge leaving the delay line is within the delta delay generated by the phase detect delay line.

Once one of these situations is achieved, the master DLL is considered locked and the dfi_init_complete signal will be asserted. Ideally the phase detect delay should be just wide enough so that when in full clock mode, the first flip-flop in the phase detect captures a 1 and the second phase detect flip-flop captures a 0. The phase detect should not be so wide as to create a situation where there are multiple delay values of the slave delay line that have the first flip-flop read a 0 and the second flip-flop read a 1. This would create more error in the slave delay lines. Conversely, if the phase detect is too narrow, the first and second phase detect flip-flops would always return the same value. This would create an oscillation between adding or subtracting multiple increments or decrements. This would also increase the error in the slave delay line settings since there would not be an accurate value of the number of delay elements in one cycle.

4.4.2. DLL Lock Debugging

The master DLL locking logic includes a debug feature to observe the increment and decrement profile of the DLL lock logic over an 8 sample running window. The lock_dec_dbg (bits [23:16]) and lock_inc_dbg (bits [31:24]) fields of the phy_dll_obs_reg_0 parameter hold the last 8 results of the master DLL phase testing. Bit 0 corresponds to the latest sample, bit 1 corresponds to the previous sample, etc. and bit 7 hold the 8th previous sample of the increment and decrement procession of the master DLL locking function. See the Table below to know the meaning of these bits.

Table 4.5. DLL Locking Bit Settings

lock_dec_dbg	lock_inc_dbg	Description
0	0	The master DLL did not change the tap setting of the delay line for this sample period.
0	1	The master DLL added one tap setting to the delay line for this sample period.
1	0	The master DLL subtracted one tap setting from the delay line for this sample period.
1	1	Error condition. This combination should never occur.

The master DLL functions with the most accuracy when the phase detect window is just large enough to cause the locking mechanism to respond with a “00” setting indicating no change in the tap setting. If the phase detect window is too small, the increment/decrement debug fields will never respond with a “00” in the same cycle. As elements are added to the phase detect logic, the param_phase_detect_sel field of the *phy_dll_master_ctrl_reg* parameter will increase and the increment/decrement debug fields will start responding with “00”.

A debug feature is also included that provides visibility into the magnitude of the local drift of the master DLL once it is locked. The same increment/decrement debug fields can be sampled to detect N consecutive increments of decrements in a 8 sample window. The value of N is determined by the setting of the param_dll_lock_num field (bits [18:16]) of the *phy_dll_master_ctrl_reg* parameter. Each time the increment/decrement debug fields detect N consecutive increment or decrement functions, the value of the dll_unlock_cnt field (bits [7:3]) of the *phy_dll_obs_reg_0* parameter is incremented. The dll_unlock_cnt value is reset by asserting and de-asserting the dll_reset_n signal to the PHY. The dll_unlock_cnt value will saturate at a value of 0x1f. Any time the dll_unlock_cnt field is triggered to increment, the current state of the lock_inc_dbg and lock_dec_dbg are stored in these fields.

4.5. DLL static aging

When devices age, its characteristics change based on whether it toggles or not. If the device consistently changes state while aging, the delta in rise/fall propagation is negligible. If the device is in a static state while aging, the delta in rise/fall propagation can be significant. This leads to duty cycle impacts on clocking within the PHY. Because the PHY uses both edges of the clock for the memory DQS/DQ, the duty cycle impact affects the timing characteristics to the Flash device in DDR modes. The delay lines can be impacted in multiple ways:

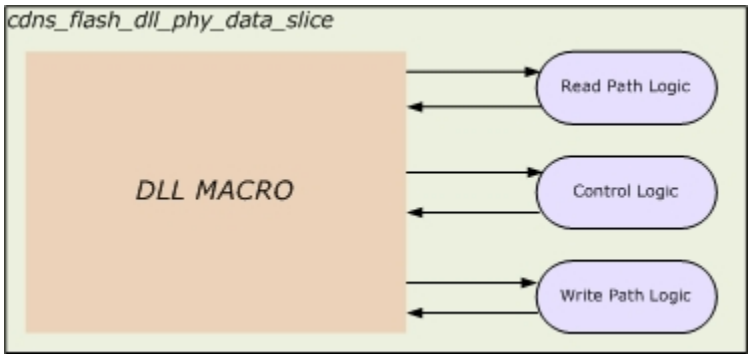
- Even when a delay line is always active, the part of the delay line not being used is in a static state. PVT changes or a frequency change can cause more delay elements to be used after a period of aging, affecting the duty cycle of the output.
- When delay lines are inactive (during normal operation or low power scenarios), the entire delay line is in a static state and will age asymmetrically.

An extra enable is added to the delay element to split the control over the return path and the transmission path. This allow control over the unused elements. The mentioned enable signal allows for a control signal to create a toggling signal in the delay line. The low frequency toggle can be supplied from the PHY level. There are no requirements on the toggle clock, no balancing or timing required.

4.6. Cadence Design IP DLL PHY Slice Floorplan

The Cadence Design IP DLL PHY slice contains the read and write datapath as well as the asynchronous timing paths. The following figure shows a block diagram of the data slice. This diagrm is not intended to show physical placement guidelines.

Figure 4.6. Data Slice Overview



4.6.1. PHY Slice Routing

The data slice will be required to route on lower levels of metal to maximize portability to various designs without requiring a re-route step.

The following rules should be used for pin placement:

- The pins should be placed on the highest level metal available that meets this constraint. For example, if the highest permitted level of metal for the data slice is metal 4, then all the pins should be placed on metal 4.
- Pins should be placed on the grid. Off-grid pin placement can cause problems for top level integration.
- Pins should be 2x the minimum metal width high, and 2x the minimum metal width wide for easier top level routing.
- The DLL macro will also be at the top level of metal, so routing cannot go over it.

4.6.2. PHY Slice Layout

Within the PHY slice is the read capture path that is timing-critical. The goal for placement is to route the modules such that the skew between the data signals arriving in the slice is minimized. Also, the absolute delay of the DQS (clock) associated with the data bits needs to be matched. For this, the grouping of the read path capture flip-flops is critical; these should be grouped and placed relatively close to the DQS (clock) from the DLL.

The paths from the read DQS DLL output to the clock pin of each entry_* reg should be balanced. This will not happen naturally since some of the paths will have an extra inverter. One way of achieving balance is to use clock tree synthesis with the output of the DLL treated as a clock and the clock pins of these flip-flops treated as endpoints. The Cadence STA scripts measure the quality of this balancing regardless of the method used. This path should be minimized in general.

All of paths from the DQS pad to endpoints within the PHY (such as the ddr_close and dqs_count regs) should be balanced against each (but not against the entry_*regs above) with minimal latency.

It is useful to group the “read” and “write” signals into groups and to physically locate them adjacent to the appropriate pins.

Table 4.6. Pin Grouping for the Cadence Design IP DLL PHY Data Slice

Group Name	Instances
read_reg_0	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}0
read_reg_1	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}1
read_reg_2	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}2
read_reg_3	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}3
read_reg_4	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}4
read_reg_5	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}5
read_reg_6	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}6
read_reg_7	dll_phy_slice_core/data_slice_0/dfi_read_datablk/read_datablk_fifo/dll_entry_flop_h(l)_{0/1/2../7}7
read_reg_cmd	dll_phy_slice_core/data_slice_0/dfi_read_cmdblk/read_cmdblk_fifo/dll_entry_flop_l_{0/1/2../7}0
write_data_oe_0	io_datacell_0/*
write_data_oe_1	io_datacell_1/*
write_data_oe_2	io_datacell_2/*

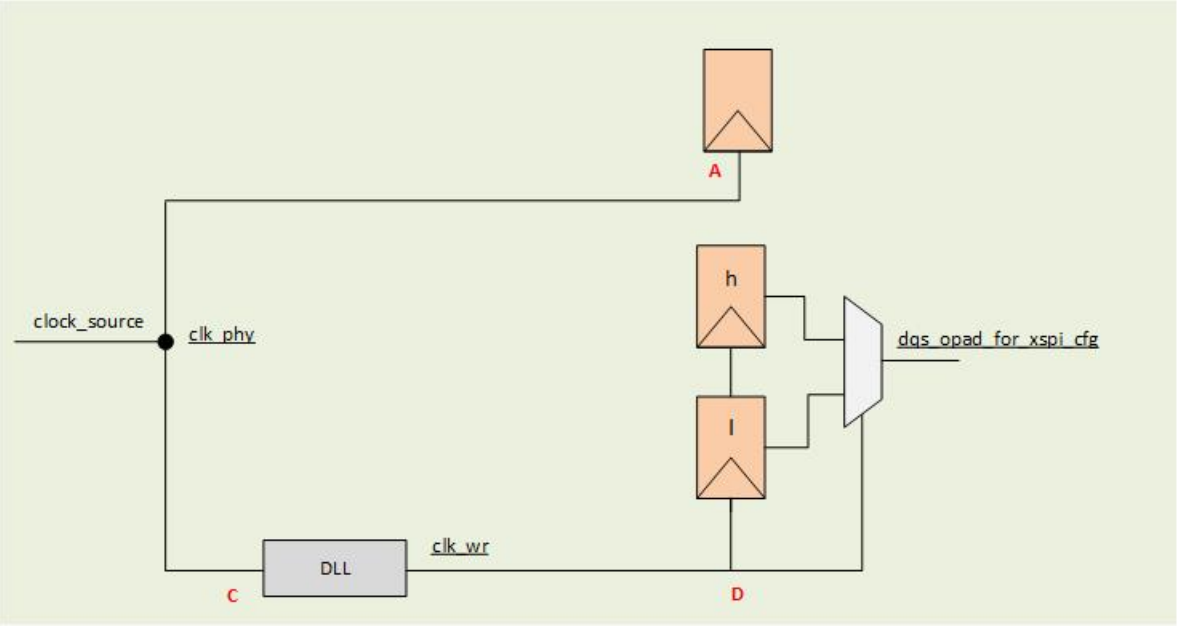
Group Name	Instances
write_data_oe_3	io_datacell_3/*
write_data_oe_4	io_datacell_4/*
write_data_oe_5	io_datacell_5/*
write_data_oe_6	io_datacell_6/*
write_data_oe_7	io_datacell_7/*
write_cmd_oe	io_cmdcell/*

For each DQ bit of the PHY slice, 16 capture flip-flops are used (8 deep FIFO). The flip-flops are all fed by the mem_data signal and should be placed physically close together. For CMD signal 8 capture flip-flops are used (CMD line is sampled only on rising edge of the sampling clock). The flip-flops are all fed by the mem_cmd signal and should be placed physically close together.

4.6.3. PHY Slice Clock Tree Synthesis

The following diagram shows a block diagram of the clocks in the PHY slice.

Figure 4.7. Data Slice Clock Tree Synthesis Relationships



The STA scripts assume that the relationship A=C+D is present, where:

- A is the clock pin and sync point for all flip-flops and clock gate cells that are driven by the clk_phy signal as well as the mux input in the webar cell in the control slice.

  dll_phy_addr_cntrl_core/dfi_io_addr_cntrl/webar/hic_io_mux_ca/hic_dnt_mux2/B driven by clock clk_phy

- C is the clock tree delay to the DLL input.

  dll_phy_slice_core/data_slice_0/dll/dll_delay_line_clk_wr/delay_0/r1/hic_dnt_nand2/A1 driven by clock clk_phy

- D is the clk_wr clock tree delay from the DLL to the data_out multiplexer select inputs driven by the clk_wr signal and to the data_out flops wr_h/wr_l

from dll_phy_slice_core/data_slice_0/dll/dll_delay_line_clk_wr/delay_0/r2/hic_dnt_nand2/Y to dll_phy_slice_core/data_slice_0/io_data*cell_?/hic_io_mux_ddrmux/hic_dnt_mux2/S0 and dll_phy_slice_core/data_slice_0/io_data*cell_?/wr_h/clk_phy and and dll_phy_slice_core/data_slice_0/io_data*cell_?/wr_l/clk_phy

A, C+D should have the same clock tree delay and be considered sync points for clock tree synthesis (CTS).

Cadence recommends the use of inverters only in this clock tree to achieve better transition characteristics with the smallest clock tree topology. In addition, the user should avoid the use of cells with the highest drive strengths to prevent electromigration problems. The highest metal layers should be used for clock net routing, and extra space should be included around the clock nets to prevent crosstalk. In addition, the following suggestions are recommended for the individual clocks:

- **clk_phy in control slice -**

  The following multiplexer input pin should be used as sync pins to balance with other clock tree leafs:

  - dll_phy_addr_cntrl_core/dfi_io_addr_cntrl/webar/hic_io_mux_ca/hic_dnt_mux2/I1

- **clk_wr -**

  The following multiplexer select pins should be used as sync pins to balance with other clock tree leafs:

  - dll_phy_slice_core/data_slice_0/io_datacell_0/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_datacell_1/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_datacell_2/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_datacell_3/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_datacell_4/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_datacell_5/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_datacell_6/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_datacell_7/hic_io_mux_ddrmux/hic_dnt_mux2/S0
  - dll_phy_slice_core/data_slice_0/io_cmdcell/hic_io_mux_ddrmux/hic_dnt_mux2/S0

- The following flop clock input pins should be used as sync pins to balance with other clock tree leafs:
  - dll_phy_slice_core/data_slice_0/io_datacell_0/wr_l*

    dll_phy_slice_core/data_slice_0/io_datacell_0/wr_h*
  - dll_phy_slice_core/data_slice_0/io_datacell_1/wr_l*

    dll_phy_slice_core/data_slice_0/io_datacell_1/wr_h*
  - dll_phy_slice_core/data_slice_0/io_datacell_2/wr_l*

    dll_phy_slice_core/data_slice_0/io_datacell_2/wr_h*
  - dll_phy_slice_core/data_slice_0/io_datacell_3/wr_l*

    dll_phy_slice_core/data_slice_0/io_datacell_3/wr_h*
  - dll_phy_slice_core/data_slice_0/io_datacell_4/wr_l*

dll_phy_slice_core/data_slice_0/io_datacell_4/wr_h*

- dll_phy_slice_core/data_slice_0/io_datacell_5/wr_l*

  dll_phy_slice_core/data_slice_0/io_datacell_5/wr_h*

- dll_phy_slice_core/data_slice_0/io_datacell_6/wr_l*

  dll_phy_slice_core/data_slice_0/io_datacell_6/wr_h*

- dll_phy_slice_core/data_slice_0/io_datacell_7/wr_l*

  dll_phy_slice_core/data_slice_0/io_datacell_7/wr_h*

- dll_phy_slice_core/data_slice_0/io_cmdcell/wr_l*

  dll_phy_slice_core/data_slice_0/io_cmdcell/wr_h*

The loopback multiplexer dll_phy_slice_core/data_slice_0/dqs/hic_io_lpbk_mux/hic_dnt_mux2 is a crossover cell between the clocks mem_dqs and clk_phy. The clock tree should be stopped at dll_phy_slice_core/data_slice_0/dqs/hic_io_lpbk_mux/hic_dnt_mux2/I1 pin.

5. Programming the PHY registers

It is important to program the PHY registers after controller initialization is done for it to work properly.

A script is provided in software directory to assist in the programming of the PHY. Please read the REAMDE* file to get information on how to use the script.

NOTE: It is important to program the values generated by the script into the PHY registers before starting any operations to the Flash device after controller initialization is done.

Appendix A. FAQs

1. **In Dll Bypass mode, for a range of delays in the delay elements for the corners, I would require to set different values in the slave DLL (+-1) for the simulations to pass. This would make it difficult later for firmware when we began to do lab test. It is there any fix for this? What is your spec. range for the delays?**

   For bypass mode, we may never be able to hit the 1/4 cycle delay that is required. So, typically what is done is, use a high frequency clock (say 200Mhz) and see how many elements it takes for the DLL to lock (from the obs registers).

   Typically, most of the delay variation is related to process and VT contributes only to around 30% of variation. So, with the obs register values, you will know how much delay is a delay element contributing (ball park number).

   Using this value, program the slave delay lines with a setup time required in that mode. The data valid windows are huge at these frequencies and there will be enough hold time to account for the variations of delay elements due to VT.

2. **The double sync logic used in the flash PHY phase detector, as shown in Figure 4.2 in the databoook - This is a phase detector, the path is not a regular CDC path. For a specific delay line setting, the phase of the double sync FF input is fixed. So if a meta-stability occurs, it may occur in consecutive clock cycles. Is this true and could this mess up the phase detector? And why?**

   It won't mess up the phase detection. Depending on the value that phase1 or phase2 steady on to, the new delay value to the delay chain may get incremented or decremented. Ultimately, the sampling would be in the steady region of the signal.

3. **Is it possible that meta-stability might occur on both phase1 and phase2 sync stage at the same time?**

   Typically, phase1 and phase2 are away from each other by around 3-4 delay elements (programmable from 1 to 8). So, both of them going into meta-stability is not possible.

4. **Assume the answer to above is No. If meta-stability occurs on phase1 sample flop in consecutive clock cycles, phase1 signal will be 0 or 1 in a random fashion. Could you elaborate on the following step that phase detector will do and how the meta-stability is flushed out?If the answer to above is yes - If meta-stability occurs on both phase1 and 2 sample flops in consecutive clock cycles, phase1 and 2 signals will be 0 or 1 in a random fashion. Could you elaborate on the following step that phase detector will do and how the meta-stability is flushed out for both phase logic?**

   The logic to determine if there is a requirement for a delay value increment/decrement to the delay chain is activated only once every 4 clocks. So, if there is back to back meta-stability, the 'delay value' computed in the previous cycle will not change. Most probably, this new value would have put the phase1 and phase2 into stable regions and the logic determines there is a lock. Even in the scenario where there is meta-stability observed in consecutive 'phase1/phase2' analysis clock cycle, ultimately the delay value will converge to a number that will give a DLL lock as well as have phase1/phase2 being captured in their respective stable regions.

5. **How are PVT updates done in the flash PHY?**

   A controller can inititate updates to the PHY periodically using the "dfi_ctrlupd_req" signal.

6. **How to choose delay cells for the delay line?**

   The delay value of one element is the sum of the delays of the dly2, dly4 and two nand gates.

   delay = dly2 + dly4 + 2*NAND

   The delay introduced by the delay element depends on the frequency range the PHY will work with, desired sampling resolution and area. The minimum frequency that should be covered in full clock mode will determine the minimum delay of each delay element. The minimum delay can be calculated in the following way:

Min.delay = (max_clock_period_in_full_clock_mode + margin) / 256

The maximum delay value should be determined by the minimum tDVW (data valid window) and desired number of samples for that data window.

Max.delay = tDVW_min / no_of_samples

The minimum delay value is a critical requirement to meet in BC corner.

The maximum delay value is just an estimation which should not be exceed in WC corner. The delay value in WC corner should be as close as possible to the results of BC corner.

Note that the wider frequency range you want to cover the lower delay range you will get (the lower frequency you want to cover the higher minimum delay you will have in one delay element while the max delay is constant; also area will be bigger)

7. **How to transcode a delay values in the phy_dll_slave_ctrl_reg into a time units?**

   a) If the param_dll_bypass_mode is set to 1 then delay value can be calculated as:

   DLLdly * ('register_value' + 1), where

   DLLdly is a delay introduced by a single delay element

   'register_value' is a value stored in read_dqs_delay, clk_wr_delay or clk_wrdqs_delay

   b) If the param_dll_bypass_mode is set to 0 delay value depends on dll_locked_mode (in the phy_dll_obs_reg_0 register):

   locked in full clock mode - the delay value will be equal to: Tnfclk * ('register_value' + 1)/ 256, where:

   Tnfclk - period of the nf_clk clock signal 'register_value' is a value stored in read_dqs_delay, clk_wr_delay or clk_wrdqs_delay

   locked in half clock mode - same as for full clock mode but the 'register_value' must be lower than 128 (which means the maximum delay equal to half clock cycle)

   saturation - delay can be calculated in the same way as for param_dll_bypass_mode set to 1 (this case is typical for low values of the nf_clk frequency)

   Please note this only refers to the delay line - it doesn't take into account other delays inside the SOC structure (i.e. logic's propagation delays).

8. **How does *tsel_start and *tsel_end influent on TSEL delay.**

   TSEL will start after detection of the negative edge of the 'rebar_dfi' signal or negative edge of the 'dfi_rd_pre_post_amble signal'. The end of the TSEL signal is determined by the closing time of the internal DQS gate signal.

   1) TSEL ON condition - The TSEL is ON minimum two clock cycles after the negative edge of the 'rebar_dfi' signal or negative edge of the 'dfi_rd_pre_post_amble' signal. The start time can be delayed by 'dqs_select_tsel_start' or 'data_select_tsel_start' fields. The delay resolution is half clock cycle.

   2) TSEL OFF condition - The TSEL is OFF minimum one clock cycle after the internal DQS gate signal goes low. The end time can be delayed by 'dqs_select_tsel_end' or 'data_select_tsel_end' fields. The delay resolution is half clock cycle.

   The internal DQS gate signal is controlled by the 'gate_cfg'. The end of the DQS gate signal can be additionally delayed by 'gate_cfg_close'. The delay resolution is clock cycle for both parameters. The DQS gate signal is asserted high when

'dfi_rddata_en' goes high and de-asserted when 'dfi_rd_pre_post_amble' and 'rebar_dfi' are high or when there is no 'cebar_dfi' (ie. all 'cebar_dfi' signals are high or 'cle_dfi' is high in NV-DDR2/NV-DDR3/ToggleMode work mode)

9. **During the dll loopback mode test, the DQ and DQS IO pad will be acting as both driver and receiver. What kind of value will be on signals tsel_data_X_opad and tsel_dqs_X_opad, will it be tsel_off_value or tsel_rd_value?**

   The dfi_rddata_en is ORed internaly with lpbk_start and the TSEL logic is driven from this ORed signal. The value of the TSEL port will be as defined in the tsel_rd_value. lpbk external mode "is not intended to be used on systems where the ASIC is connected to actual flash parts" so the ODT does not need to be turned on.

10. **The dqs_underrun assert high - what does it mean?**

    It means that PHY tries to output data on DFI interface but the internal Data FIFO is empty. It may happen in 2 cases:

    * the delayed DQS signal was not propagated to the internal Data FIFO - either by device error or issue in DQS connectivity or DQS gating configuration (i.e. decrease the gate_cfg)

    * the delayed DQS signal arrived to the Data FIFO too late - increase the rd_del_sel to postpone outputting data by the PHY.

11. **The dqs_underrun/dqs_overflow do not assert but the dfi_rddata are corrupted. What to do next?**

    Check if data are generated correctly by the device.

    Check if the delay for DQS is set incorrect ie. check the relation of Data FIFO data bus and data strobe to determine how to adjust the delayed DQS.

    You may check the following signals:

    * strobe: dll_phy_slice_core/data_slice_0/dfi_read_datablk/clk_dqs

    * data bus: dll_phy_slice_core/data_slice_0/dfi_read_datablk/mem_data

    Please note that those signals doesn't take into account the per-bit deskew delays.

    The PHY read DQS delay line can be controller by read_dqs_delay field in the phy_dll_slave_ctrl_reg.

12. **I see that DLL locking takes a lot of time. Is it possible to short it?**

    Yes, you can change the value of param_dll_start_point field in the phy_dll_master_ctrl_reg register.

13. **Has External loopback been enabled for the xSPI variant?**

    Yes.Controlled by bit 9 of the register phy_gate_lpbk_ctrl_reg. Setting bit 9 to "0" enables external loopback and setting it to "1" enables internal loopback

Appendix B. Document Revision History

Table B.1. Document Revision History

Revision	Modification	Data	Author
1.0	First Release	06 Mar 2019	ppietron@cadence.com
1.01	Updated SD/eMMC DQS Settings	08 Mar 2021	eandrews@cadence.com
1.02	Minor updates for the SD/eMMC variant	25 Jan 2022	rousset@cadence.com