**DDR PHY test**

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Description | Author |
| 0.1 | 2016-9-13 | Initial | maji |
| 0.2 | 2016-9-26 | add pll test output pin, add circuit requirement | maji |
|  |  |  |  |

# DDR PHY test overview

Apart from the BIST loopback test modes (see “BIST Loopback”), there are various other PUB test modes that are used to facilitate the testing of various features of the PHY. These test modes include ATPG scan mode, delay line oscillator mode, and PLL test mode. The guidance provided in this chapter for silicon testing and characterization of Digital Delay Lines (DDLs) and Phase Locked Loops (PLLs) can be applied to either product characterization and/or production testing.

This chapter describes the following:

❖ “ATPG Test Mode”

❖ “DDL Testing”

❖ “PLL Testing”

❖ “SSTL I/O Testing”

❖ “BIST Loopback”

# Test Pins & Test voltage

The test will use following SPI interfaces to configure DDR PHY:

❖SPI\_MS3\_DI

❖SPI\_MS3\_DO

❖SPI\_MS3\_SCLK

❖SPI\_MS3\_CS0N

PLL test will use following interfaces to output test clk:

❖DDR\_DTO0 : PLL reference clk

❖DDR\_DTO1 : PLL output clk

The voltage for test is as follows:

❖DDR\_VDDQ: 1.2V

❖DDR\_PLL\_VDD: 1.8V

# Circuit requirement

Connect DDR\_ZQ pin through 240ohm±1% resistor to ground.



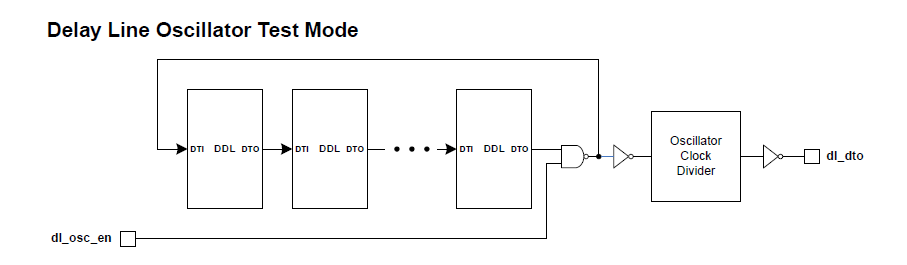
# ATPG Test Mode

All registers in the AC, ACX48, DATX8 and DATX4X2 blocks are put on scan chains. There are several scan chains in each block. Each scan chain is based on an internal functional clock group with no mixing of clock edges. Different functional clocks which have the same edge and their clock trees are functionally balanced are treated as one clock group. For example, the combination of either the delayed CTL clock or the delayed DDR clock (i.e. clocks after the DDLs). These clocks are all active rising edges and are functionally balanced and timed. All scan chains have lockup latches at their outputs. The scan chains will be integrated into whole chip scan chain.

# DDL Testing

All DDR4 multiPHY PHY blocks (AC, ACX48, DATX8, DATX4X2) contain multiple embedded adjustable DDLs. The DDL is an adjustable delay line, with a basic logic function of a buffer.

To facilitate an exhaustive DDL test, logic has been incorporated within all the PHY blocks and the PUB to provide a DDL ring oscillator test mode. In the ring oscillator test mode all of the NDLs, BDLs and LCDLs within the DATX8 or DATX4X2 block are connected together into a single ring. This is done by connecting the digital test output (DTO) of one delay line to the digital test input (DTI) of another delay line. The DTO of the last delay line is connected to a NAND gate and looped back to the DTI of the first DDL to form an oscillator. This is illustrated in Figure 6-1 on page 891. The oscillator output is then divided down by a programmable ratio and is output as the delay line test output. The same is done independently on the AC or ACX48 block. Each DATX8/PHYDATX4X2/PHYAC/PHYACX48 block has its own ring, permitting parallel testing of the entire interface. The ring has a control point to enable/disable oscillation. On the output of the ring there is a divider to slow down the clock signal produced by the ring in order to be able to process it in the low-speed CTL clock domain in which the PUB may be running during testing. By varying the delay line select inputs of either one or multiple delay lines, one can characterize and test the existence, linearity and range of each delay line. To amplify a smaller change in the delay to a measurable change in the period of the delay test output signal, the oscillator output is divided down by a programmable ratio that ranges from 1 to 65536. The choice of the divide ratio may depend on the desired test resolution and test time. The PUB contains extra logic that automatically measures the period of the delay line test output. The measured period can be read from a register and is expressed as a code relative to the period of the PUB controller (ctl\_clk) clock.



The suggested methodology for testing DDLs is as follows:

1. Setup the PUB and PHY macros for DDL lest:

a. If there is a plan to observe the DDL oscillator test mode output on DTO I/O, enable the DTO I/O and make sure it not in power down:

i. DSGCR.DTOPDR = 1’b0

ii. DSGCR.DTOOE = 1’b1

b. Optionally place the PLLs and I/Os in power down mode by writing to their respective powerdown control registers.

c. Inhibit VT calculation and disable VT drift compensation by setting the VT drift limit to 0:

i. PGCR6.INHVT = 1’b1

ii. PGCR6.DLDLMT = 9’h00

d. Reset the PHY macros without resetting the PUB logic and disable address/command lane MDL calibrations. Note that resetting the PHY macros requirement is only for simulation purposes to prevent read data being erroneously signaled back to the DFI controller. In normal testing, the DFI controller is inactive and therefore does not matter whether the FIFOs are in reset or not:

i. PGCR1.PHYHRST = 1’b0

ii. PGCR1.MDLEN = 1’b0

e. Disable byte lanes MDL calibrations and disable byte lanes duty cycle correction:

i. DXCCR.MDLEN = 1’b0

ii. DXCCR.DXDCCBYP = 1’b1

iii. Disable address/command duty-cycle correction: PGCR4.ACDCCBYP = 1’b1

f. De-assert the reset on the PHY macros (this reset is not self-clearing):

i. PGCR1.PHYHRST = 1’b1

g. Enable delay line oscillator test mode:

i. PGCR0.DLTMODE = 1’b1

h. Wait a minimum of 100ns.

i. Set all DDLs in the PHY block to minimum delay (set to 0x0).

2. Test the DDL:

a. Choose on DDL

b. Enable DDL oscillation with the desired oscillator divide ratio. If there is a plan to observe the DDL oscillator test mode output on the DTO I/O, select the PHY block whose test output should go out onto the DTO I/O by setting the register PGCR0.DTOSEL. If the chosen DDL is an LCDL, make sure only that LCDL is selected by setting PGCR0.OSCWDL, PGCR0.OSCWDDL or PGCR0.OSCACDL to the correct setting:

i. PGCR0.OSCEN = 1’b1

ii. PGCR0.OSCDIV = desired oscillator divide ratio

iii. PGCR0.DTOSEL = PHY instance to output DTO

iv. PGCR0.OSCWDL = 2’b01 or 2’b10 depending on LCDL being selected

v. PGCR0.OSCWDDL = 2’b01 or 2’b10 depending on LCDL being selected

vi. PGCR0.OSCACDL = 2’b01 or 2’b10 depending on LCDL being selected

c. Wait for TBD time.

d. Trigger DDL test output period measurement:

i. PGCR0.DLTST = 1’b1

e. Wait for DDL test output period measurement to complete by polling PGSR1.DLTDONE = 1’b1 for AC/ACX48 and DXnGSR1.DLTDONE = 1’b1 for DATX8/DATX4X2.

f. Read the measured period from PGSR1.DLTCODE (for AC/ACX48) and DXnGSR1.DLTCODE for DATX8/DATX4X2.

g. Disable DDL test oscillation and measurement:

i. PGCR0.OSCEN = 1’b0

ii. PGCR0.DLTST = 1’b0

h. Increment the fine delay element by one step and repeat Steps 2b to 2h.

i. Once all fine steps settings of the DDL have been completed, set the fine delay setting back to minimum (0x0) and increment the course delay elements by one step. Repeat Steps 2b to 2i skipping step 2h (i.e. keeping fine delay code at minimum.

j. Once all coarse steps have been completed, you now have a record of values for each fine and coarse step setting of the DDL under test.

k. Set the delay code for the DDL under test back to its minimum and move to another DDL in the PHY block. Repeat this until all DDLs have been completed.

# PLL Testing

The digital test outputs from the PLL provide test access directly to the PLL. The control of the digital test outputs (signal and PHY selection) is provided by PLLCR/PLLCR0.DTC and PGCR0.DTOSEL. The digital test outputs from the AC/ACX48 and DATX8/DATX4X2 are signals pll\_dto[1:0] and dl\_dto which then become output phy\_dto[1:0] of the PHY\_top.

Test Steps:

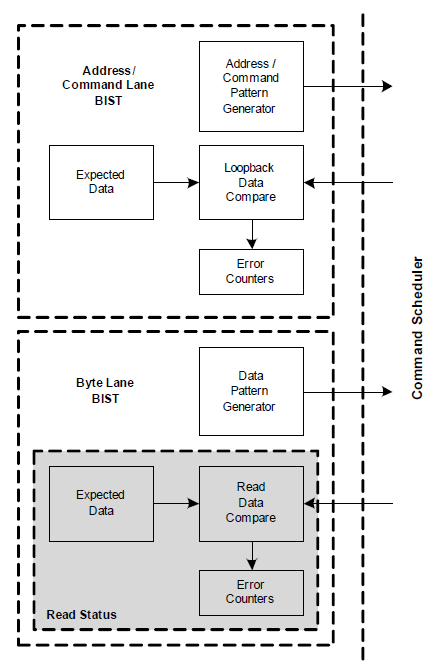
1. Configure PLL to desired frequency through SPI ports.
2. Configure PLLCR/PLLCR0.DTC and PGCR0.DTOSEL to select the tested PLL to DDR\_DTO pins.
3. Measure the clock on DDR\_DTO0 and DDR\_DTO1 pins
4. Repate step 1~3 for other PLL test

# SSTL I/O Testing

As the I/O cell contains embedded boundary scan support logic, the user may connect standard boundary scan logic to these ports and use the boundary scan logic for Testing.

# BIST Loopback Testing

The BIST block is a programmable built-in-self-test engine that gives the application a method of testing the PHY and I/Os for at-speed production testing.



Before triggering BIST, the PHY must be initialized as in normal mission mode. This is done automatically after de-assertion of configuration reset. Also, the relevant registers in the PUB must be set to the correct settings for the particular DDR system. BIST run sequence is as follows:

1. After configuration reset is de-asserted, wait for PHY to finish initialization by polling a ‘1’ status on PGSR0.IDONE. Other PUB registers can be configured while waiting for the PHY initialization to be completed. This is similar to the procedure used in mission mode.

2. Indicate to the PUB that SDRAM initialization will not be triggered by writing ‘1’ to PIR.INIT and ‘1’ to PIR.CTLDINIT.

3. Wait for the PUB to finish or bypass the DRAM initialization by polling a ‘1’ status on PGSR0.IDONE.

4. Specify the number of patterns that should be generated during the BIST run by writing to BISTWCR.BWNCT. The default value of 32 is adequate for test coverage when using random BIST patterns. Therefore this register write is optional.

5. Address/command I/O receivers are by default powered down. If planning to run pad-side loopback, make sure the address/command lane I/O receivers are not powered down by writing 0x20000000 to ACIOCR0.

6. Reset the PHY FIFOs by writing PGCR0.PHYFRST = 1’b0. This resets the read FIFOs, to ensure that there is no residual bad data or state in the FIFO (which may occur due to disabling/enabling IO receivers, etc.) before beginning BIST. Wait for a minimum of 8 cfg\_clk cycles. Then clear the reset by writing PGCR0.PHYFRST = 1’b1.

7. Trigger BIST to start running by writing 3’b001 to BISTRR.BINST. The BISTRR register also contains various BIST configuration fields that must be set to the desired configurations at the same time that the BIST run is triggered. This includes configurations such as loopback mode versus DRAM mode, enabling BIST run on address/command lane or byte lanes, enabling data mask to be included in BIST runs, selecting the byte lane to be used in the run, and so on. Refer to the BISTRR register for other selections that can be made during BIST run using this register.

9. Wait for BIST to finish by polling a ‘1’ status on BISTSR.BDONE.

10. Read BISTWER.ACWER or BISTWER.DXWER to check if there were any errors on the address/command lane or the selected byte lane. If there are no errors, then the test is done.