DDR Tuning and Calibration Guide on Zynq-7000

ScanWorks® Processor-based Functional Test for DDR

Tune-to-Test

by Larry Osborn

ASSET™

Larry Osborn

Larry Osborn, Senior Product Manager, at ASSET InterTech, has over 30 years of experience in product management, hardware/software product design and development, product delivery to the marketplace and user support. Over the years, Larry has a proven track record for identifying user needs and opportunities in the marketplace, providing innovative solutions and exceeding the expectations of users. At ASSET, Larry is responsible for the profit and loss for a product group. Prior to ASSET, he has held positions with Lockheed Martin, OCD Systems, Wind River, Hewlett-Packard, Ford Aerospace and Intel® Corporation. He holds a Bachelor's Degree in Computer Science from the University of Kansas and various technical and marketing training certifications.

Table of Contents

Table of Figures

DDR Tuning and Calibration with ScanWorks Processor-based Functional Test for DDR

DDR Calibration Background

As noted in my blog, *DDR Tuning and Calibration with ScanWorks*, configuration of DDR memories in a board design involves dozens-to-hundreds of register-writes to the memory controller. The memory controller registers set the timing parameters and operating mode of the DDR interface. There are registers that need only be set once; these values can be thought of as those that are static, not subject to board, component or environmental conditions. The other register set are variable; specifically used to account for timing variations caused by board layout, or environmental conditions. Calibration/tuning is not just writing a value in a register to adjust the DQS delay or other timing parameter, it involves repetition where a timing parameter is changed, testing memory access to see that it passes and repeating the process until the memory access fails. The failure will indicate that the timing settings are either too slow or too fast for the memory to respond. This would suggest an algorithmic approach is required and the level of confidence of the optimal settings is determined by the robustness of the memory test.

Programming of the memory controller is influenced by several aspects of the board design. If any of these items are changed, then the memory controller settings must also change.

1. Processor/memory controller used
2. Number and organization of physical memory chips used
3. Speed grade of the memory chips
4. Operating speed of the DDR interface
5. Board layout, including trace length and impedance considerations
6. Memory Vendor change

To get more background on Write Leveling, DQS gating, Read data eye and Write data eye calibration operations please refer to the blog.

To address the tedious nature of entering the register values, pouring over the technical reference manuals for both the SoC and the memory devices, and then finding the optimal values, ASSET has a solution to shorten the design engineering effort and provide robust memory testing to ensure the manufacturing process efficiency. This paper provides the details about that solution.

Theory of Operation

Overview

The primary purpose of ScanWorks® Processor-based Functional Test for DDR (PFTDDR) is twofold. At the highest level the concept is tune-to-test. Tuning to be accomplished by the design engineering team and test to be conducted by the manufacturing test team. However, in order to have a valid tuning process, testing must accompany the tuning at the design phase to validate the tuning values chosen. At the basic level, PFTDDR is used to enter DDR controller settings, automatically tune DDR settings, interactively validate those settings, and then provide a means to export these settings and desired test sequenced for automated testing. PFTDDR software is comprised of two major components: PFx developer host software, and PFTDDR target agent. These components communicate with each other through a communication channel via a JTAG connection as shown in Figure 1.

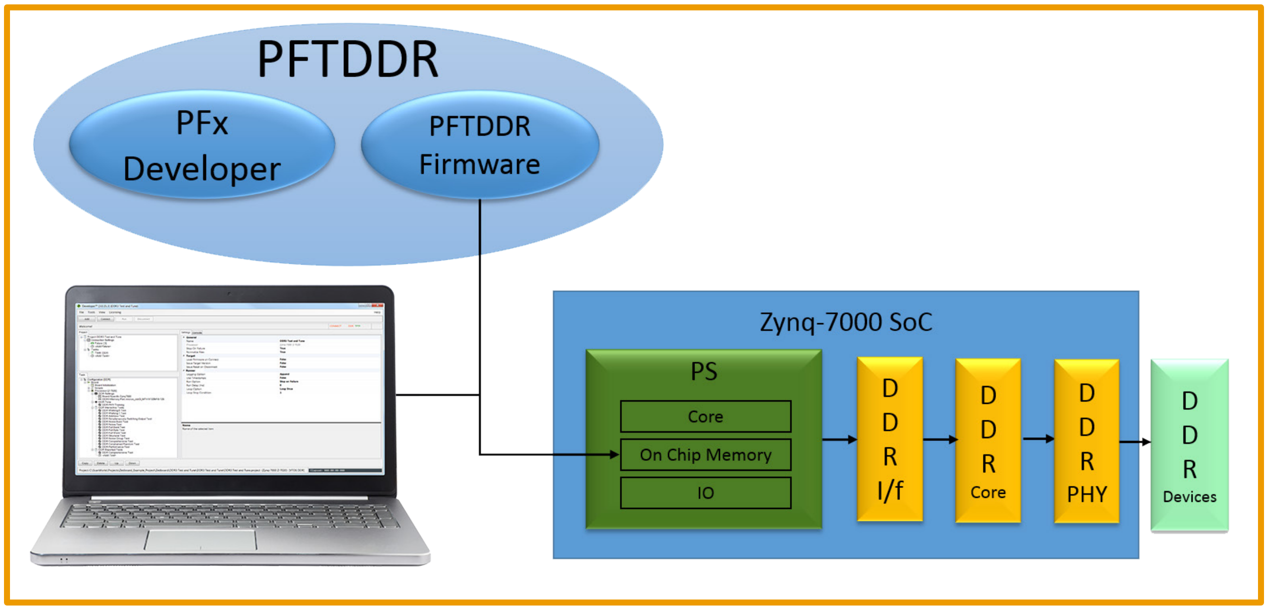


Figure 1: PFx System Overview

The PFTDDR target agent runs from the on-chip memory of your embedded system, allowing you to fully test your external DDR memory using the embedded processor. The agent provides a rich set of capabilities for configuring, testing, and optimizing your DDR memory and PHY.

The PFx developer host software runs on your Windows host PC and provides a user interface to the PFTDDR firmware. The PFx developer provides two main functions: a tool for optimizing and validating DDR memory; and a tool for exporting DDR test sequences for use within the ScanWorks platform.

PFTDDR Software Based Tuning

ScanWorks PFTDDR provides commands to tune and calibrate DDR memory. These commands automatically adjust one or more calibration parameters and test memory at each setting, generating a scoreboard of settings that pass and fail.

The specific tuning process varies based on the processor type and the features of the memory controller. However, the general tuning process follows this procedure.

1. Set the write leveling delay for one-byte lane to a fixed value
2. Optional - calibrate other DDR3/DDR4 parameters (DQS gating, read data eye, write data eye, etc.)
3. Run a memory cell test, using burst transactions
4. Run a memory bus noise test, using burst transactions
5. Log the results of the memory tests
6. Increment the write leveling delay value and repeat at step 2

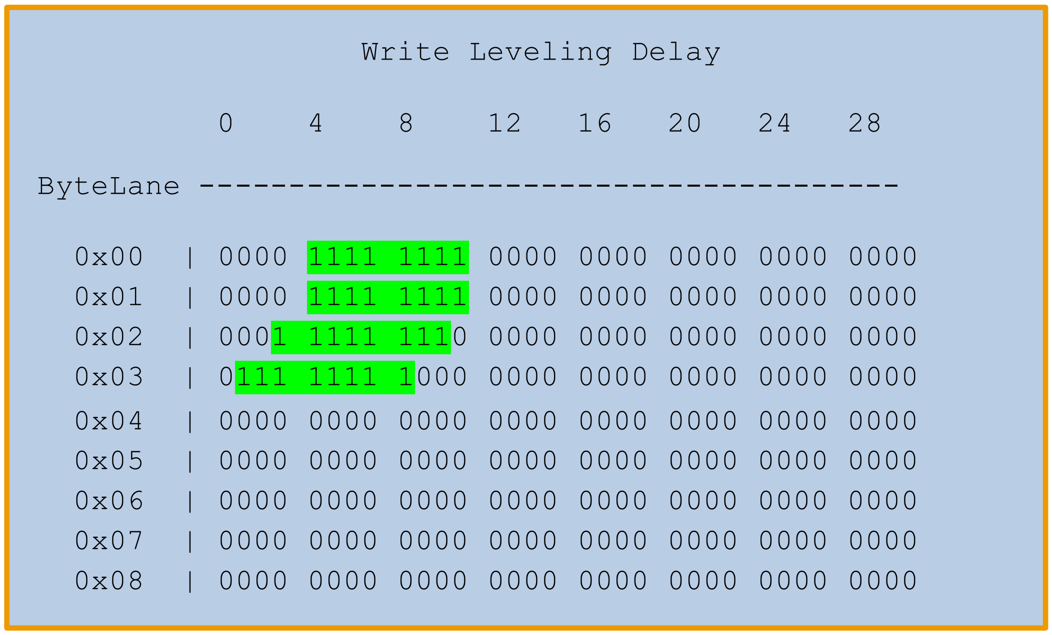
This procedure is repeated for all valid byte lanes in the design, including an ECC byte. For example, if the board uses 32-bit wide data bus with ECC disabled, the procedure is repeated 4 times. If the board uses a 64-bit wide data bus with ECC enabled, the procedure is repeated 9 times. The tuning process used by the PFx developer is fully software based. The value at each data point of an empirical memory test is run and does not rely on any hardware feedback signals to determine if the memory setting passed or failed. This addresses the drawbacks to relying on hardware-assisted write leveling noted later in the paper under *Hardware-Assisted Tuning*.

Write Leveling Examples

At the completion of DDR tuning, the PFx developer generates a summary of the tuning results. The following sections illustrate specific examples of write leveling calibration performed by PFx developer. The output format will vary based upon the architecture.

Point-to-point Topology with DDR4 Memory

The first example is the write leveling calibration result for a reference board using DDR4 memory chips using point-to-point routing for all signals. In the table below (Figure 2) each write leveling delay value represents 1/8 of a clock period. Byte Lane 0 passed with delay values from 4/8 clocks to 11/8 clocks, while Byte Lane 3 passed with delay values from 1/8 clocks to 8/8 clocks.



```
                  Write Leveling Delay

           0    4    8    12   16   20   24   28

ByteLane ---------------------------------------

  0x00   | 0000 1111 1111 0000 0000 0000 0000 0000
  0x01   | 0000 1111 1111 0000 0000 0000 0000 0000
  0x02   | 0001 1111 1110 0000 0000 0000 0000 0000
  0x03   | 0111 1111 1000 0000 0000 0000 0000 0000
  0x04   | 0000 0000 0000 0000 0000 0000 0000 0000
  0x05   | 0000 0000 0000 0000 0000 0000 0000 0000
  0x06   | 0000 0000 0000 0000 0000 0000 0000 0000
  0x07   | 0000 0000 0000 0000 0000 0000 0000 0000
  0x08   | 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 2: Write Leveling Pass Matrix

After completion of tuning, PFx developer recommended the following settings, which represent the middle of the operating window for each byte lane as shown in Figure 3.

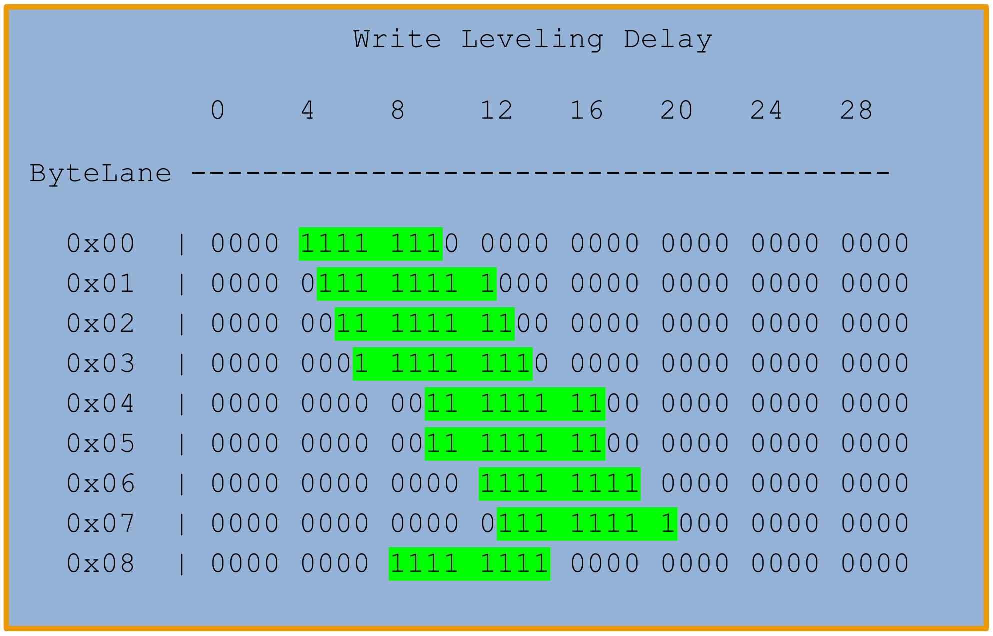
Byte Lane	Write Level Delay
0	7/8 clocks
1	7/8 clocks
2	6/8 clocks
3	4/8 clocks

Figure 3: Write Level Delay DDR

Note that even though the board layout used the point-to-point topology, empirical testing showed that the optimal write level delay varied by 3/8 of a clock between byte lane 0 and byte lane 3.

Fly-by Topology with DDR3 Memory

The second example, shown in Figure 4, is the write leveling calibration result for a reference board using DDR3 memory chips on a DIMM module. The DIMM module has a total of 9 memory chips (8 for data, 1 for ECC) and the signals are routed using the fly-by topology. As in the other example, each delay value represents 1/8 of a clock period.



```
                  Write Leveling Delay

          0    4    8    12   16   20   24   28

ByteLane ---------------------------------------

  0x00  | 0000 1111 1110 0000 0000 0000 0000 0000
  0x01  | 0000 0111 1111 1000 0000 0000 0000 0000
  0x02  | 0000 0011 1111 1100 0000 0000 0000 0000
  0x03  | 0000 0001 1111 1110 0000 0000 0000 0000
  0x04  | 0000 0000 0011 1111 1100 0000 0000 0000
  0x05  | 0000 0000 0011 1111 1100 0000 0000 0000
  0x06  | 0000 0000 0000 1111 1111 0000 0000 0000
  0x07  | 0000 0000 0000 0111 1111 1000 0000 0000
  0x08  | 0000 0000 1111 1111 0000 0000 0000 0000
```

Figure 4: Write Leveling Example 2

After completion of tuning, PFx developer recommended the following settings (Figure 5), which represent the middle of the operating window for each byte lane.

Byte Lane	Write Level Delay
0	7/8 clocks
1	8/8 clocks
2	9/8 clocks
3	10/8 clocks
4	13/8 clocks
5	13/8 clocks
6	15/8 clocks
7	16/8 clocks
8 (ECC byte)	11/8 clocks

Figure 5: Write level Delay DDR3

As expected when using the fly-by topology, there is a lot of variation in the optimal write leveling setting for each byte lane. Of particular note is that byte lane 0 and byte lane 7 do not share any common write leveling delay value that passed all memory tests, which makes the tuning process an absolute requirement.

Hardware-Assisted Tuning

The JEDEC specification for DDR3 and DDR4 memory details a hardware-assisted process for calibrating the write leveling setting. This process involves putting the memory chips into write leveling mode and then having the memory controller sample a specific data bit (called the prime data bit) while the write leveling delay value is changed.

There are some drawbacks to relying on the hardware-assisted write leveling:

- Not all memory controllers support the automatic sampling of the prime data bit or have reported errata related to this capability.
- The prime data bit must be routed to a specific pin on the processor for correct operation. Specifically, DIMM modules can swap data signals to improve signal integrity causing the prime data bit to get mapped to the wrong pin on the processor.

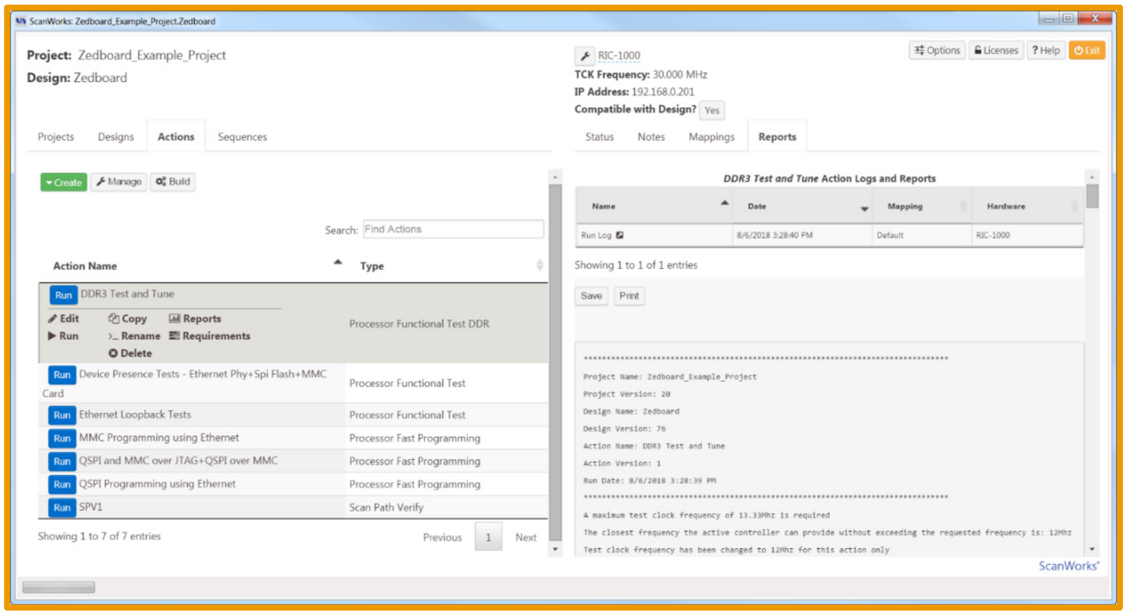
Zynq-7000 Application

The board and SoC setup

There are several setup steps required before we begin the DDR PHY training process. The best way to understand this process is via example. The example provided here is conducted on the ZedBoard. Should you require a more active visual approach, you may register for to get a technical video.

This setup will cover the major steps used in the tuning of the DDR PHY. This is not intended to be a step for step process. The information provided will provide enough data so that the process can be understood.

The process begins by loading the project file for the ZedBoard within ScanWorks and then selecting edit of the DDR3 Test and Tune action (Figure 6). For more information about the ScanWorks platform, check us out.


ScanWorks: Zedboard_Example_Project.Zedboard

Project: Zedboard_Example_Project
Design: Zedboard

RIC-1000
TCK Frequency: 30.000 MHz
IP Address: 192.168.0.201
Compatible with Design? Yes

Action Name	Type
Run DDR3 Test and Tune	Processor Functional Test DDR
Run Device Presence Tests - Ethernet Phy+Spi Flash+MMC Card	Processor Functional Test
Run Ethernet Loopback Tests	Processor Functional Test
Run MMC Programming using Ethernet	Processor Fast Programming
Run QSPI and MMC over JTAG+QSPI over MMC	Processor Fast Programming
Run QSPI Programming using Ethernet	Processor Fast Programming
Run SPV1	Scan Path Verify

Showing 1 to 7 of 7 entries

DDR3 Test and Tune Action Logs and Reports

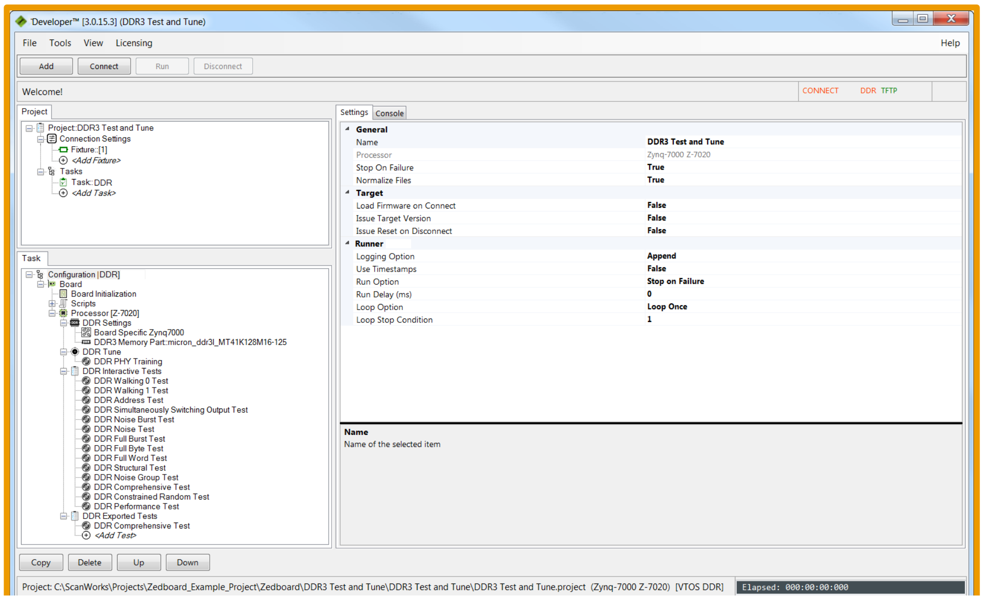
Name	Date	Mapping	Hardware
Run Log	8/6/2018 3:28:40 PM	Default	RIC-1000

Showing 1 to 1 of 1 entries

```
********************************************************************************
Project Name: Zedboard_Example_Project
Project Version: 20
Design Name: Zedboard
Design Version: 76
Action Name: DDR3 Test and Tune
Action Version: 1
Run Date: 8/6/2018 3:28:39 PM
********************************************************************************
A maximum test clock frequency of 13.33Mhz is required
The closest frequency the active controller can provide without exceeding the requested frequency is: 12Mhz
Test clock frequency has been changed to 12Mhz for this action only
```

Figure 6 ScanWorks Zedboard Example Project

The purpose of editing the action is so the user can see all the aspects of what is provided by the interface to support tuning and testing of the DDR memories. Editing the action will launch the PFx developer user interface as shown below in Figure 7.


Developer™ [3.0.15.3] (DDR3 Test and Tune)

```
Project::DDR3 Test and Tune
  Connection Settings
    Fixture::[1]
    <Add Fixture>
  Tasks
    Task::DDR
    <Add Task>
```

```
Configuration [DDR]
  Board
    Board Initialization
    Scripts
    Processor [Z-7020]
      DDR Settings
        Board Specific Zynq7000
        DDR3 Memory Part::micron_ddr3l_MT41K128M16-125
      DDR Tune
        DDR PHY Training
      DDR Interactive Tests
        DDR Walking 0 Test
        DDR Walking 1 Test
        DDR Address Test
        DDR Simultaneously Switching Output Test
        DDR Noise Burst Test
        DDR Noise Test
        DDR Full Burst Test
        DDR Full Byte Test
        DDR Full Word Test
        DDR Structural Test
        DDR Noise Group Test
        DDR Comprehensive Test
        DDR Constrained Random Test
        DDR Performance Test
      DDR Exported Tests
        DDR Comprehensive Test
        <Add Test>
```

General	
Name	DDR3 Test and Tune
Processor	Zynq-7000 Z-7020
Stop On Failure	True
Normalize Files	True
Target	
Load Firmware on Connect	False
Issue Target Version	False
Issue Reset on Disconnect	False
Runner	
Logging Option	Append
Use Timestamps	False
Run Option	Stop on Failure
Run Delay (ms)	0
Loop Option	Loop Once
Loop Stop Condition	1

Name
Name of the selected item

Project: C:\ScanWorks\Projects\Zedboard_Example_Project\Zedboard\DDR3 Test and Tune\DDR3 Test and Tune\DDR3 Test and Tune.project (Zynq-7000 Z-7020) [VTOS DDR]

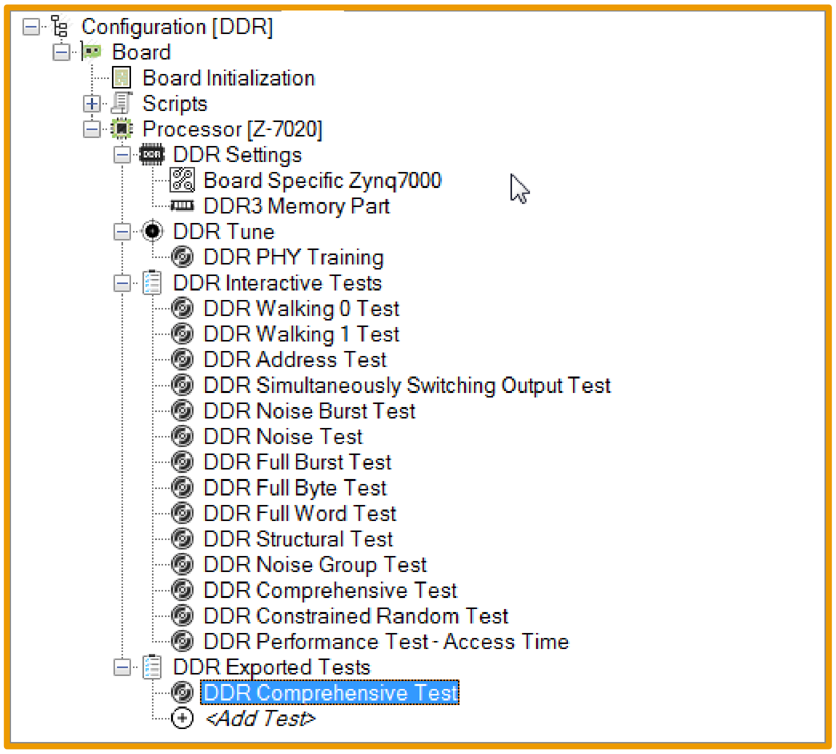
Elapsed: 000:00:00:000

Figure 7: PFx Developer

We'll start by examining the configuration tree.

Understanding the Configuration Tree

The configuration tree (Figure 8) maintains an optional board initialization file, optional script files and commands, DDR settings, and a list of memory tests to be exported. The tree is organized as a single Configuration node which contains a single Board node. Each node can be expanded or collapsed (shown expanded). The Board node contains a Board Initialization node, a Scripts node, and a Processor node. The Processor node contains a DDR Settings node, DDR Tune node, DDR Interactive Test node, and a DDR Exported Tests node. The DDR Settings node contains a Board Specific node and a Memory part node. The DDR Tune node contains the command for DDR PHY Training. The DDR Interactive Tests contain multiple nodes that are used to run a memory test interactively. The Exported Tests node can contains zero or more memory tests to be exported. In our example the Comprehensive Test is exported.

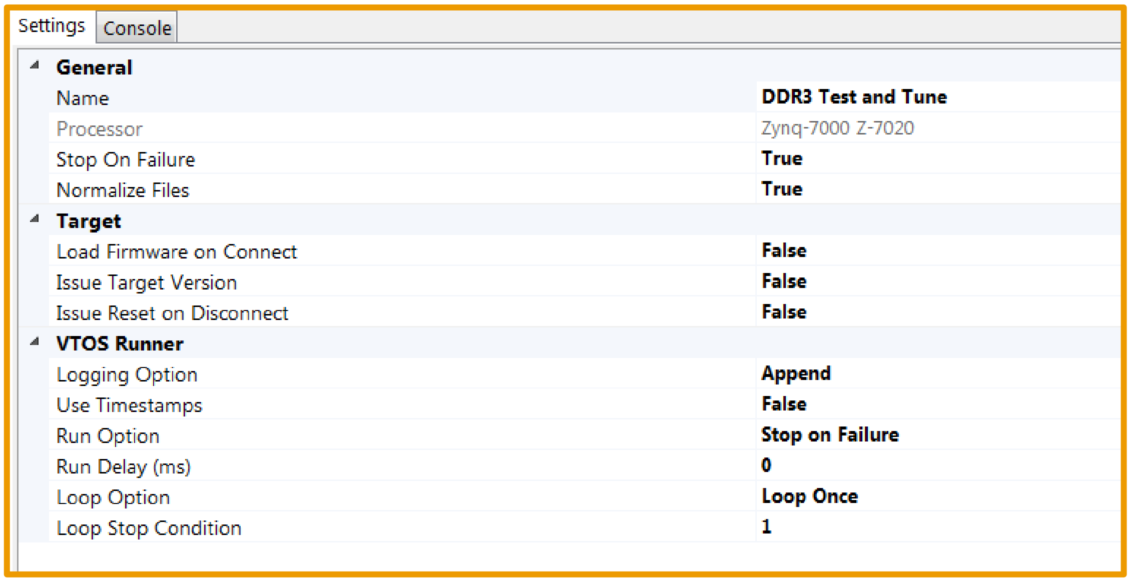


```
Configuration [DDR]
  Board
    Board Initialization
    Scripts
    Processor [Z-7020]
      DDR Settings
        Board Specific Zynq7000
        DDR3 Memory Part
      DDR Tune
        DDR PHY Training
      DDR Interactive Tests
        DDR Walking 0 Test
        DDR Walking 1 Test
        DDR Address Test
        DDR Simultaneously Switching Output Test
        DDR Noise Burst Test
        DDR Noise Test
        DDR Full Burst Test
        DDR Full Byte Test
        DDR Full Word Test
        DDR Structural Test
        DDR Noise Group Test
        DDR Comprehensive Test
        DDR Constrained Random Test
        DDR Performance Test - Access Time
      DDR Exported Tests
        DDR Comprehensive Test
        <Add Test>
```

Figure 8: Configuration Tree

Note: The Memory Part node will vary based on the type of DDR memory selected under DDR Settings.

Selecting a node in the configuration tree, with the **Settings Mode** tab selected, displays all the properties available for that node. Use the **Settings Mode** panel to update the value of a given property for the highlighted node (Figure 9).



General	
Name	**DDR3 Test and Tune**
Processor	Zynq-7000 Z-7020
Stop On Failure	**True**
Normalize Files	**True**
Target	
Load Firmware on Connect	**False**
Issue Target Version	**False**
Issue Reset on Disconnect	**False**
VTOS Runner	
Logging Option	**Append**
Use Timestamps	**False**
Run Option	**Stop on Failure**
Run Delay (ms)	**0**
Loop Option	**Loop Once**
Loop Stop Condition	**1**

Figure 9: Settings Mode

Board Initialization

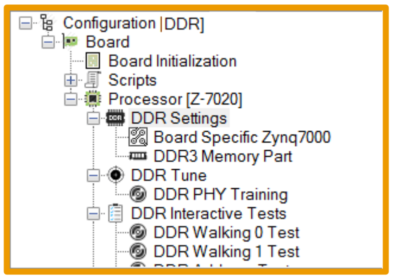
The next step is to specify if you want to run a script file that prepares the board for proper operation. This script may disable a watchdog timer, configure GPIO signals, enable clocks, configure power, or other board specific steps. If this step is enabled, the board initialization script will be executed once a new connection is established. There are no board initialization steps needed for the ZedBoard.

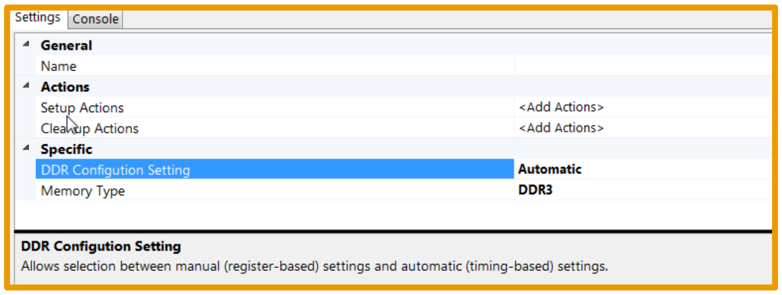
Configuring the DDR Controller

PFTDDR requires several DDR controller parameters that differ by circuit board design. These parameters are stored in the Board Specific and Memory Part nodes.

Choose DDR Configuration Settings

The first step for any new configuration is to choose your DDR configuration setting (Figure 10). PFx developer allows two choices: Automatic and Manual.





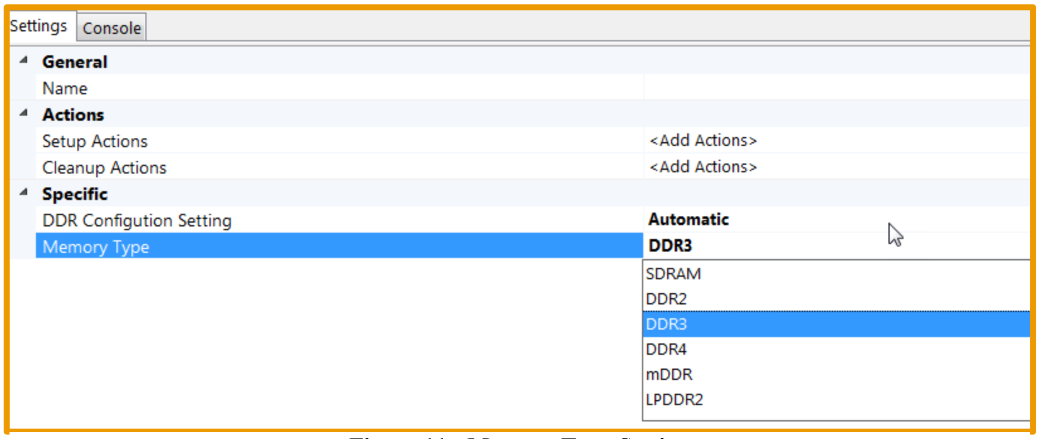
DDR Configution Setting: Automatic
Memory Type: DDR3

DDR Configution Setting
Allows selection between manual (register-based) settings and automatic (timing-based) settings.

Figure 10: DDR Settings Node and Setting Content

The Automatic setting uses values for memory controller registers that are derived from memory timing specifications. The manual setting uses values for memory controller registers as specified directly from numeric values. If you are starting with timing specifications from a memory data sheet as the basis for configuring the memory controller, select Automatic. If you are starting with a raw register dump, select manual. For this Zedboard example we will select Automatic.

Next, click on the pull-down menu opposite of the Memory Type parameter (Figure 11). Use this pull-down to select from the various memory types, such as SDRAM, DDR2, DDR3, DDR4, and LPDDR2. For the Zedboard the setting is DDR3.



DDR Configution Setting: Automatic
Memory Type: DDR3

Figure 11: Memory Type Setting

Loading and Running PFTDDR Firmware

Now that we have examined the key parameters of the configuration process, we need to load the target agent onto the SoC. PFx developer provides a user interface to the firmware running on your embedded device. This firmware is loaded and executing on the ZedBoard using JTAG. And is accomplished by selecting "**Connect**" button within the PFx developer UI (Figure 12).

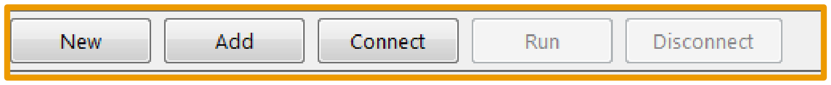
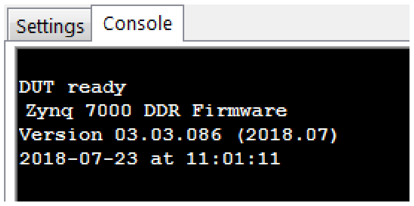


Figure 12: Connect

When the firmware successfully boots, it prints "DUT Ready" in the console window (Figure 13). The PFTDDR agent typically boots and is ready in under one second.



```
DUT ready
 Zynq 7000 DDR Firmware
Version 03.03.086 (2018.07)
2018-07-23 at 11:01:11
```

Figure 13: Target Agent Response

Update Board Specific Settings

Click on the board specific node under DDR settings (Figure 14) and then click on the settings tab (Figure 15) to view and modify all automatic and manual settings. Depending on your Auto versus Manual setting, certain sections will be collapsed. Under the General section at the top, several read-only parameters are provided for convenience. Click on any parameter and the panel at the bottom is updated with help text.

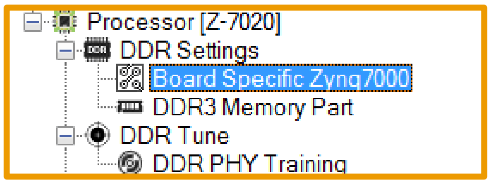


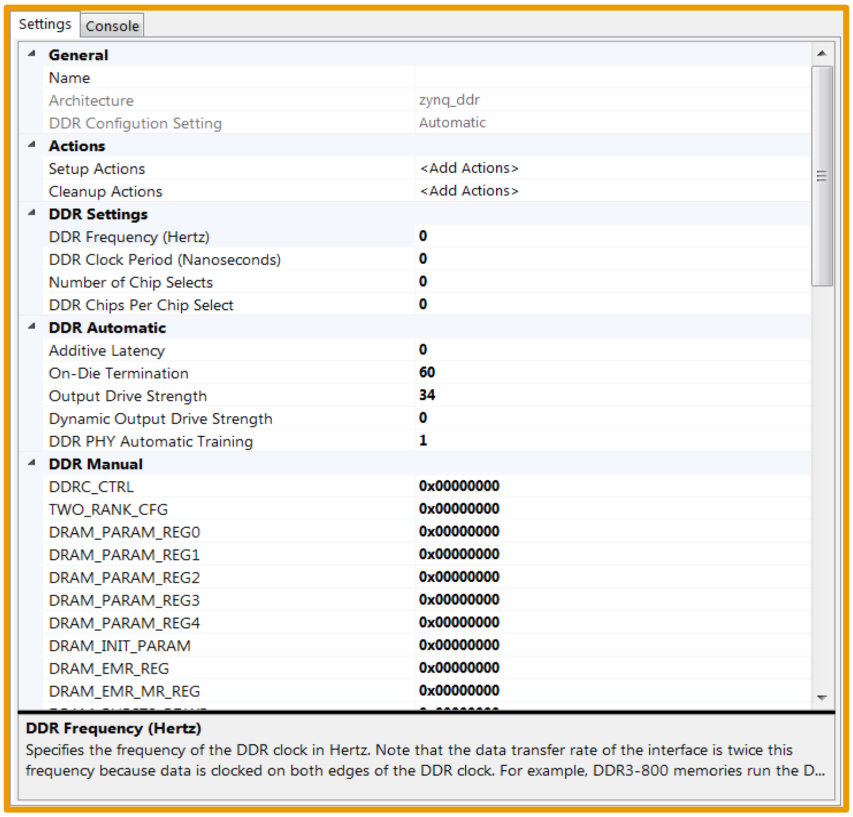
Figure 14: Zynq 700 Board Specific Node

The parameters that need to be set for the Zedboard are:

DDR Frequency	*533333333*
Number of Chip Selects	*1*
DDR Chips Per	*2*
On-Die Termination	*60*
Output Drive Strength	*30*

Then under the **DDR Automatic & Manual** settings for the Zedboard are:

CPU Frequency	*667000000*



General	
Name	
Architecture	zynq_ddr
DDR Configution Setting	Automatic
Actions	
Setup Actions	<Add Actions>
Cleanup Actions	<Add Actions>
DDR Settings	
DDR Frequency (Hertz)	0
DDR Clock Period (Nanoseconds)	0
Number of Chip Selects	0
DDR Chips Per Chip Select	0
DDR Automatic	
Additive Latency	0
On-Die Termination	60
Output Drive Strength	34
Dynamic Output Drive Strength	0
DDR PHY Automatic Training	1
DDR Manual	
DDRC_CTRL	0x00000000
TWO_RANK_CFG	0x00000000
DRAM_PARAM_REG0	0x00000000
DRAM_PARAM_REG1	0x00000000
DRAM_PARAM_REG2	0x00000000
DRAM_PARAM_REG3	0x00000000
DRAM_PARAM_REG4	0x00000000
DRAM_INIT_PARAM	0x00000000
DRAM_EMR_REG	0x00000000
DRAM_EMR_MR_REG	0x00000000

DDR Frequency (Hertz)
Specifies the frequency of the DDR clock in Hertz. Note that the data transfer rate of the interface is twice this frequency because data is clocked on both edges of the DDR clock. For example, DDR3-800 memories run the D...

Figure 15: Board Specific Setting

Note: The data for the DDR Settings and DDR automatic were determined by the schematic and user guide for the Zedboard.

The data for the **DDR Manual** settings (Figure 15) will initially be zero. Upon completion of the tuning process the data is imported by the user with a mouse click as shown in Figure 16.

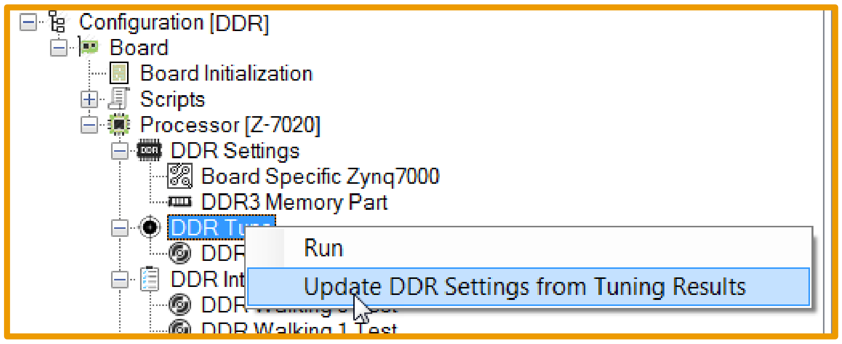


Figure 16: DDR Update After Tuning

Update Memory Part Settings

Next click on the **Memory Part** node under **DDR Settings (**Figure 17**)** and then click on the **Setting** tab (Figure 18) to view or modify all memory part settings.

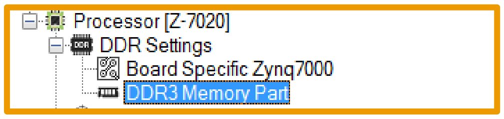
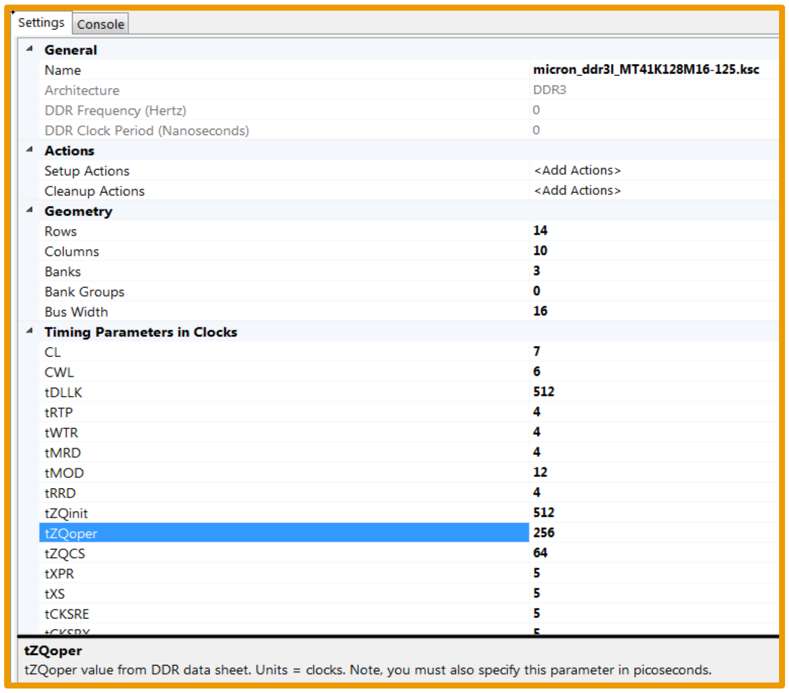


Figure 17: DDR3 Memory Part Selection



General	
Name	micron_ddr3l_MT41K128M16-125.ksc
Architecture	DDR3
DDR Frequency (Hertz)	0
DDR Clock Period (Nanoseconds)	0
Actions	
Setup Actions	<Add Actions>
Cleanup Actions	<Add Actions>
Geometry	
Rows	14
Columns	10
Banks	3
Bank Groups	0
Bus Width	16
Timing Parameters in Clocks	
CL	7
CWL	6
tDLLK	512
tRTP	4
tWTR	4
tMRD	4
tMOD	12
tRRD	4
tZQinit	512
tZQoper	256
tZQCS	64
tXPR	5
tXS	5
tCKSRE	5

tZQoper
tZQoper value from DDR data sheet. Units = clocks. Note, you must also specify this parameter in picoseconds.

Figure 18: DDR3 Memory Part Setting

Under the General section at the top, several read-only parameters are provided for convenience. Click on any parameter and the panel at the bottom is updated with help text. By starting with the Zedboard example project, we have already imported the device library for the Micron DDR3 device used in the Zedboard design. All the values needed are extracted from the data sheet of the Micron device and provided to the user within the library folders which contains many memory vendors and parts.

Configure DDR Controller

You must configure the memory controller each time you load and run the PFTDDR firmware. PFTDDR calculates the proper register values for the memory controller based on the data specified in this configuration file, converting timing information into register values as necessary.

To apply the register values to the controller hardware, select "**Configure DDR Controller**" from the **DDR Setting** right-mouse menu option (Figure 19). PFTDDR will write the values to the hardware in the proper sequence to accomplish the specified configuration.

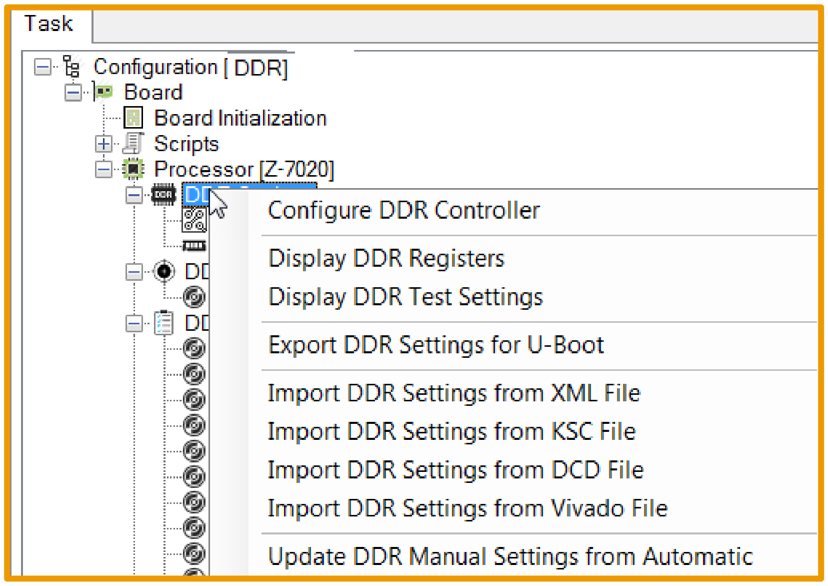
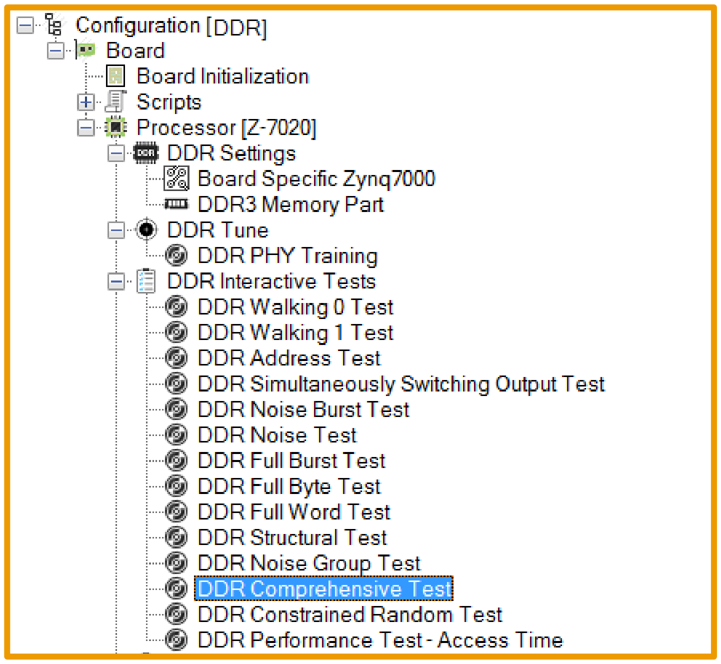

Task
Configuration [DDR]
Board
Board Initialization
Scripts
Processor [Z-7020]
Configure DDR Controller
Display DDR Registers
Display DDR Test Settings
Export DDR Settings for U-Boot
Import DDR Settings from XML File
Import DDR Settings from KSC File
Import DDR Settings from DCD File
Import DDR Settings from Vivado File
Update DDR Manual Settings from Automatic

Figure 19: Configure DDR Controller

Note: You must be connected to a target in order to run the configure operation.

DDR Configuration Validation

The final step is to ensure that the DDR controller and memory can operate properly. It is important to remember that we need to have the DDR memory system working before we try to find the optimal settings via the tuning. This step involves running a memory test. We recommend using the Comprehensive Memory test as it includes both structural and stress testing. (Figure 20)



```
Configuration [DDR]
  Board
    Board Initialization
    Scripts
    Processor [Z-7020]
      DDR Settings
        Board Specific Zynq7000
        DDR3 Memory Part
      DDR Tune
        DDR PHY Training
      DDR Interactive Tests
        DDR Walking 0 Test
        DDR Walking 1 Test
        DDR Address Test
        DDR Simultaneously Switching Output Test
        DDR Noise Burst Test
        DDR Noise Test
        DDR Full Burst Test
        DDR Full Byte Test
        DDR Full Word Test
        DDR Structural Test
        DDR Noise Group Test
        DDR Comprehensive Test
        DDR Constrained Random Test
        DDR Performance Test - Access Time
```

Figure 20: DDR Memory Test Prior to Tuning

With the DDR configuration complete, downloaded to the SoC, and verified operational, we can now begin the turning process.

DDR PHY Training

The DDR PHY implemented in the Zynq processor requires training of the following parameters:

- Write leveling
- Read DQS gate training
- Read data eye training

The memory controller can automatically perform hardware-based training using the above parameters, shown in Figure 18, every time the DDR memory is initialized. PFTDDR also provides software-based automatic training for the PHY. The software-based training finds the optimum values for these parameters:

- Write leveling
- Write data eye training
- Read DQS gate training
- Read data eye training

Tip: It is recommended that when starting with a new Zynq board design, or changing the memory part, that you enable hardware-based automatic training. After you verify all DDR memory tests pass you can then perform the software based PHY training and then modify your board file to use the optimal setting.

Executing the Software Based DDR PHY Training

By selecting DDR Tune -> Run this will start the software process running in OCM of the SoC to find the optimum values for the DDR PHY (Figure 21).

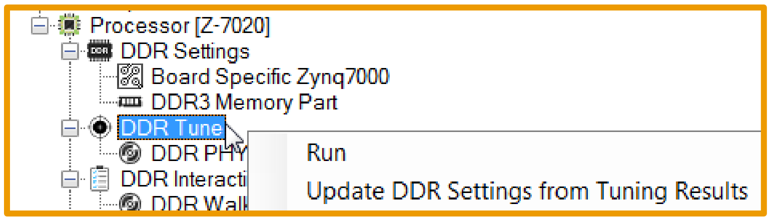


Figure 21: Start Tuning

The command performs the following steps:

1. Train the write leveling parameter by adjusting the WR DQS signal relative to CLK
    a. Start with byte lane 0, set WR DQS delay to a seed value
    b. Search outward from the seed values, first increasing the WR DQS delay value and then decreasing the WR DQS value
    c. At each delay setting, run a memory cell test and a memory bus noise test,
    d. Repeat steps a – c for byte lanes 1 – 3
2. Train the write data eye parameter by adjusting the delay of the write data relative to WR DQS
    a. Start with byte lane 0, set write data delay to a seed value
    b. Search outward from the seed values, first increasing the write data delay value and then decreasing the write data value
    c. At each delay setting, run a memory cell test and a memory bus noise test,
    d. Repeat steps a – c for byte lanes 1 – 3
3. Train the read DQS gate parameter
    a. Starting with byte lane 0, set the read DQS delay to a seed value

   b. Search outward from the seed value, first increasing the delay and then decreasing the delay
   c. At each delay setting, run a memory cell test and a memory but noise test, logging the result
   d. Repeat steps a – c for byte lanes 1 – 3
4. Train the read data eye parameter by adjusting the delay of the read data relative to RD DQS
   a. Starting with byte lane 0, set the read data delay to a seed value
   b. Search outward from the seed value, first increasing the delay and then decreasing the delay
   c. At each delay setting, run a memory cell test and a memory bus noise test, logging the results
   d. Repeat steps a – c for byte lanes 1 – 3

The software-based DDR PHY training can take significant time to run, as much as 10 minutes or more to complete. The procedure is typically run on new board designs, when changing to a new memory part, or changing the DDR interface space.

Interpreting the Training Results

At the completion of the software DDR PHY training, PFTDDR prints out a summary of the results. The training results from a ZedBoard are shown in Figure 22. Portions of the output are truncated here to save space.

```
-------------------------------------------------
Test WR DQS Delay for byte lane 0
-------------------------------------------------

++++Update register offset 0x154 <= 0x80
DDRC configuration complete
SDRAM: Full burst (32 bit) [00000000_00000000 -
00000000_000fffff]

// output truncated...

------------------------------------------------- WR DQS
Delay for byte lane 0 passed
-------------------------------------------------
-------------------------------------------------
Test WR DQS Delay for byte lane 1
-------------------------------------------------
// output truncated...
```

Figure 22: Part 1 of Tuning Results

During the tuning process the test being conducted will output data as shown in Figure 22. This data is the pass/fail output as the algorithm works through the testing of each register setting required for that byte lane.

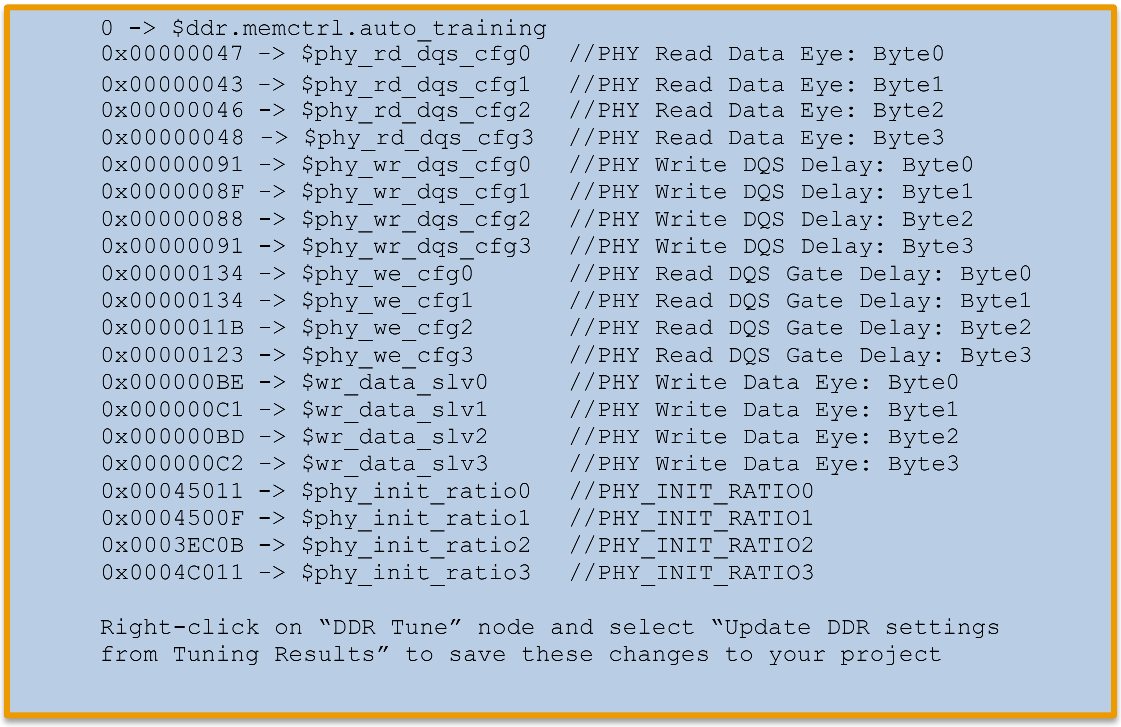
```
--------------------------------------------------
Final DDR PHY Tuning Results
--------------------------------------------------

       Write DQS/Write leveling Measurement results:

Byte 0: center 145, window (99-192) 93/256ths clock
Byte 1: center 143, window (96-190) 94/256ths clock
Byte 2: center 139, window (91-188) 97/256ths clock
Byte 3: center 144, window (92-196) 104/256ths clock

  Write Data Eye Measurement results:

Byte 0: center 190, window (145-236) 91/256ths clock
Byte 1: center 192, window (146-239) 93/256ths clock
Byte 2: center 189, window (141-238) 97/256ths clock
Byte 3: center 192, window (140-244) 104/256ths clock

      Read DQS Gate Delay Measurement results:

Byte 0: center 307, window (140-475) 335/256ths clock
Byte 1: center 309, window (134-484) 350/256ths clock
Byte 2: center 283, window (109-457) 348/256ths clock
Byte 3: center 291, window (118-464) 346/256ths clock

      Read Data Eye Measurement results:

Byte 0: center 71, window (20-122) 102/256ths clock
Byte 1: center 67, window (13-122) 109/256ths clock
Byte 2: center 69, window (15-124) 109/256ths clock
Byte 3: center 72, window (20-125) 105/256ths clock
```

Figure 23: Part 2 of Tuning Results

Next PFTDDR first displays the operating window found for each PHY parameter and for all byte lanes as a summary of the tuning as shown in Figure 23.

Update Board Specific Zynq7000 Setting

Next PFTDDR displays a summary of PHY tuning registers as shown in Figure 24.



```
0 -> $ddr.memctrl.auto_training
0x00000047 -> $phy_rd_dqs_cfg0   //PHY Read Data Eye: Byte0
0x00000043 -> $phy_rd_dqs_cfg1   //PHY Read Data Eye: Byte1
0x00000046 -> $phy_rd_dqs_cfg2   //PHY Read Data Eye: Byte2
0x00000048 -> $phy_rd_dqs_cfg3   //PHY Read Data Eye: Byte3
0x00000091 -> $phy_wr_dqs_cfg0   //PHY Write DQS Delay: Byte0
0x0000008F -> $phy_wr_dqs_cfg1   //PHY Write DQS Delay: Byte1
0x00000088 -> $phy_wr_dqs_cfg2   //PHY Write DQS Delay: Byte2
0x00000091 -> $phy_wr_dqs_cfg3   //PHY Write DQS Delay: Byte3
0x00000134 -> $phy_we_cfg0       //PHY Read DQS Gate Delay: Byte0
0x00000134 -> $phy_we_cfg1       //PHY Read DQS Gate Delay: Byte1
0x0000011B -> $phy_we_cfg2       //PHY Read DQS Gate Delay: Byte2
0x00000123 -> $phy_we_cfg3       //PHY Read DQS Gate Delay: Byte3
0x000000BE -> $wr_data_slv0      //PHY Write Data Eye: Byte0
0x000000C1 -> $wr_data_slv1      //PHY Write Data Eye: Byte1
0x000000BD -> $wr_data_slv2      //PHY Write Data Eye: Byte2
0x000000C2 -> $wr_data_slv3      //PHY Write Data Eye: Byte3
0x00045011 -> $phy_init_ratio0   //PHY_INIT_RATIO0
0x0004500F -> $phy_init_ratio1   //PHY_INIT_RATIO1
0x0003EC0B -> $phy_init_ratio2   //PHY_INIT_RATIO2
0x0004C011 -> $phy_init_ratio3   //PHY_INIT_RATIO3

Right-click on "DDR Tune" node and select "Update DDR settings
from Tuning Results" to save these changes to your project
```

Figure 24: DDR PHY Register Tuning Results

Finally, a reminder to update DDR settings with the register values displayed as shown at the bottom of Figure 24.

The manual way

Should the user want to explicitly set the values for all the DDR PHY registers, this can be accomplished, however it is not covered in this paper, but just briefly mentioned here for awareness of the capability.

Register Based Board Files

Use a Register-Based DDR board file if you want to explicitly specify all register values programmed into the DDR Memory Controller. See the user manual for more data on register-based DDR board files.

Timing Based Files

Use a Timing-Based DDR board file to have PFx DDR automatically calculate the register settings for the DDR memory controller. Register settings are based on the memory part file selected and the selected speed of the DDR interface.

In your Timing Mode board file, you must provide the following items:

- Specify the memory part
- Specify the DDR clock frequency
- Specify the number DDR chips used
- Specify DDR termination
- Specify whether hardware based DDR training is used or whether manual DDR training settings are used.

Refer to the user manual for more details.

Conclusions

DDR tuning is a process that cannot be taken lightly. Ensuring the settings for the DDR controller and PHY are the optimal for your design is critical to produce a quality product and stay competitive. ASSET has presented a software approach to DDR tuning/calibration to eliminate the tedium and potentially error prone method of translating datasheet information. The software also provides robust memory testing, which is critical to the tuning processes and is supported on all Zynq-7000 SoC regardless of board architecture.

The process involved these six steps.

1. Setting the board information
    a. DDR Frequency
    b. Number of chip selects
    c. Number of memory devices
    d. CPU Frequency
2. Loading the memory device from the library
3. Run the initial DDR configuration
4. Testing the initial configuration
5. Run the Tuning Agent
6. Copying the tuning data to the project/design

Other items to consider are the environmental requirements of your product and the consequences in production with an Engineering Change Order (ECO). Either chamber testing or ECO will necessitate a methodology for tuning the DDR memory system. If the ECO involves the memory subsystem, trace lengths or memory device changes are but two areas to be mindful of, tuning must again be addressed. PFTDDR is a tool that can handle all these scenarios and many more with only minimal data provided to the tool by user.