FPGA TECH SOLUTION

SOLUTION AHEAD

Your First DDR Interfacing Project

Introduction

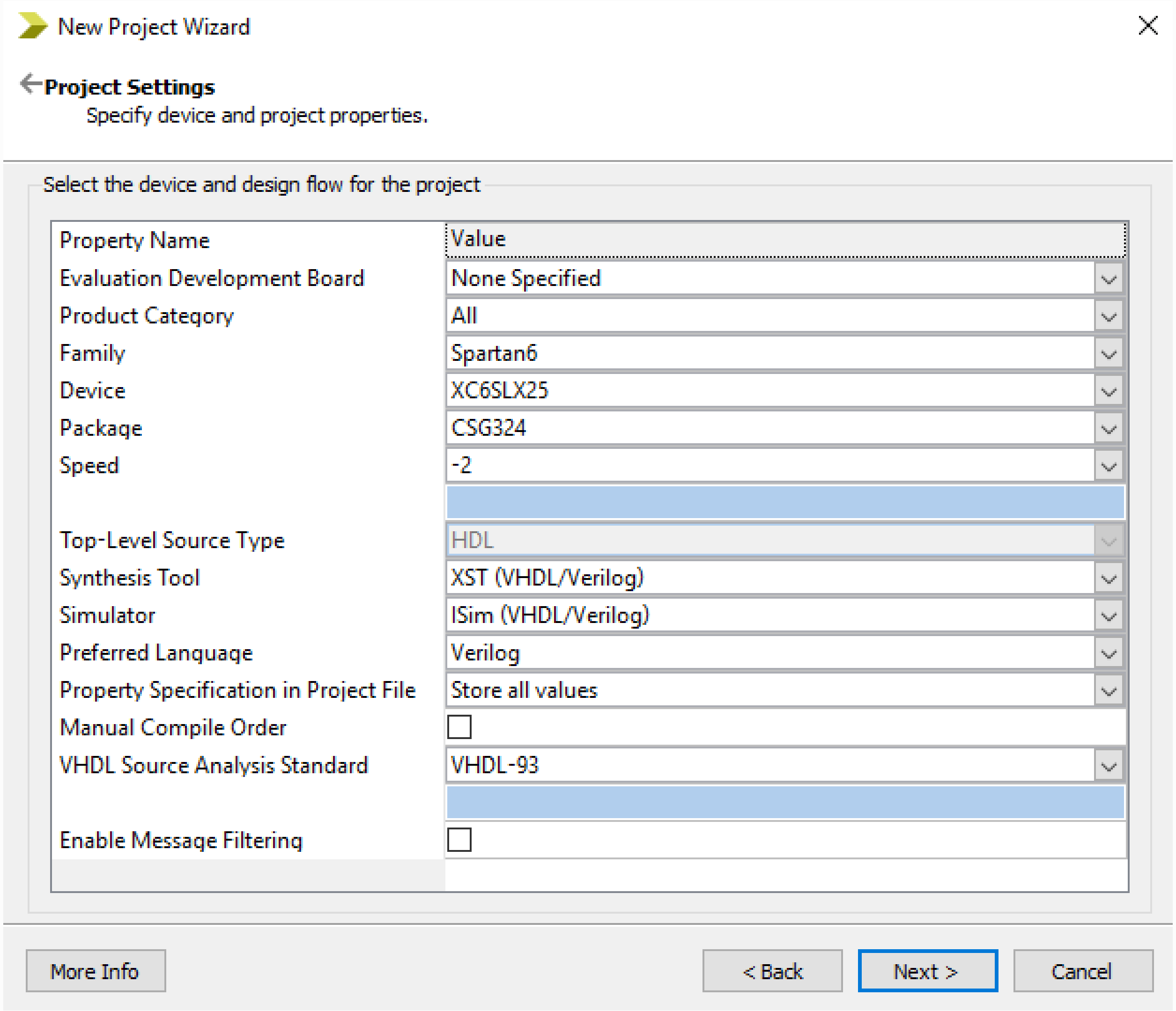
Designing and building a FPGA development board that has two BGA devices was very challenging. But what came next did seem equally challenging to us in the beginning. That challenge was testing the functionality of DDR interface. We had no prior experience of successfully building a board with a DDR device. And those who know about DDR interface can tell that it is not very easy implementing. Thanks to the memory controller built in to Spartan 6 FPGAs and Xilinx MIG tool which helped us generate sample code. Searching through websites we found out that there are not many tutorials and good example projects that are easy to follow. And we thought we should share with our readers what we learned. This post is an attempt to help beginners to get their first DDR interfacing project up running quickly, if possible without writing any code at all . This is possible because Xilinx provides a complete and working example project along with their IP. So we won’t write any code or learn how DDR works in this tutorial. This is not intended to be a tutorial on the working of DDR or FPGAs, rather a quick practical tutorial on how to run your first project that uses DDR memory.

Xilinx Spartan 6 FPGAs has hard DDR memory controller built-in which makes working with DDR easier. And ISE Core Generator supports configuring and generating code for the memory controller. The software wizard that helps with configuring and generating code for memory controller is called MIG (Memory Interface Generator). We will use MIG to generate code and will build the example project that is generated. And program and test the code on a

http://fpgatechsolution.com/product/sp6-lpddr/
This is a low cost FPGA development platform created by fpgatechsolution sp6-lpddr has a Xilinx Spartan 6 FPGA in CSG324 package and a 512Mbit LPDDR memory with lots of GPIOs for external interfacing. It has a SPI flash memory for configuration storage and 100MHz crystal oscillator as clock source. User manual and other tools for this is available at
the **https://github.com/fpgatechsolution/Spartan6_LPDDR**

Memory Controller IP configuration and code generation using MIG

Ok, that is enough introduction, lets get back to work. The only piece of software you will need is Xilinx ISE which is available for download for free from http://www.xilinx.com. Start ISE and select new project from File menu. The project wizard will pop up. Type in a project name and path in the first page. Select appropriate FPGA device in the second page. Sp6-lpddr-v1 has a Spartan 6 LX16/LX25/LX45 device (XC6SLX25). Settings as shown in the image below should work fine.



New Project Wizard

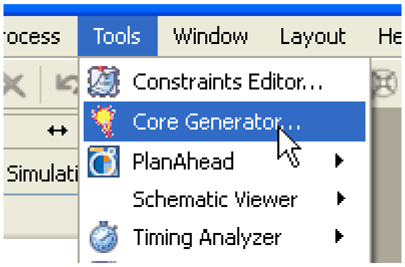
Project Settings

Specify device and project properties.

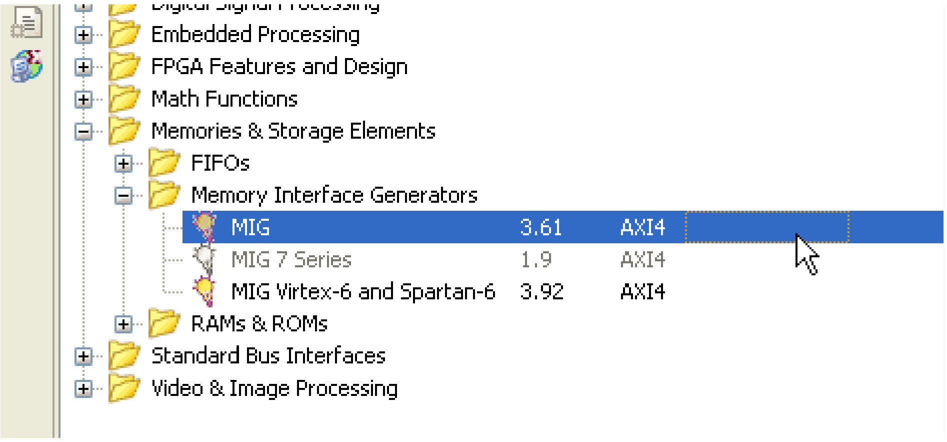
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX25
Package	CSG324
Speed	-2
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	□
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	□

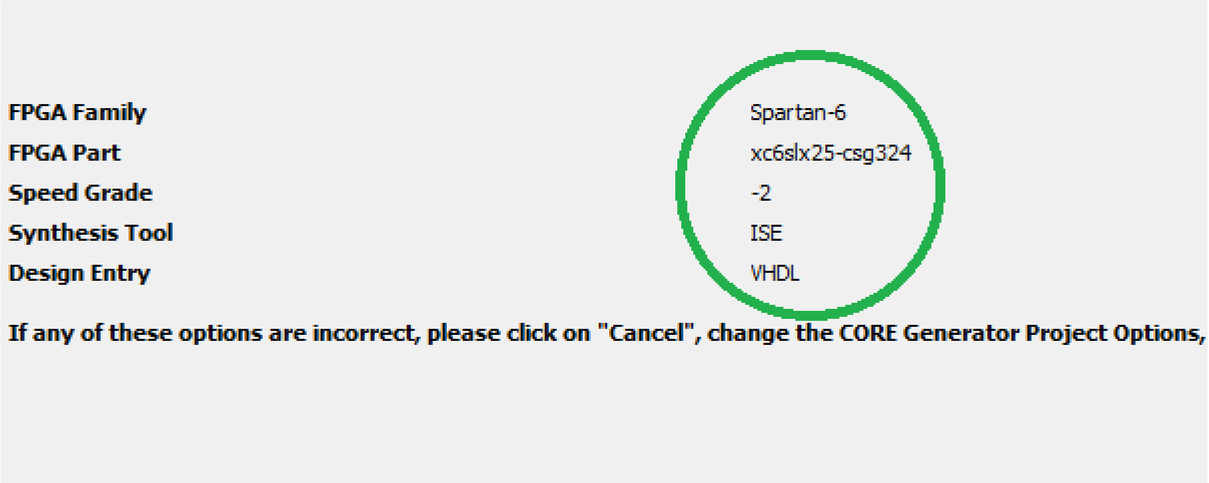
Click next and finish the wizard. Surprisingly we are not going to create or add any source files to the project and our use of ISE ends once we start the Coregen tool. To start Coregen tool, go to the Tools menu and select "Core Generator".



If Core Generator does not create a project automatically, create project by selecting File>New Project. You will need to select the FPGA and its package when creating the project. Select Verilog as design entry method. This is to make sure that Core Generator generates code in Verilog. In the IP catalog window, Find MIG under "Memory & Storage Elements" Category.



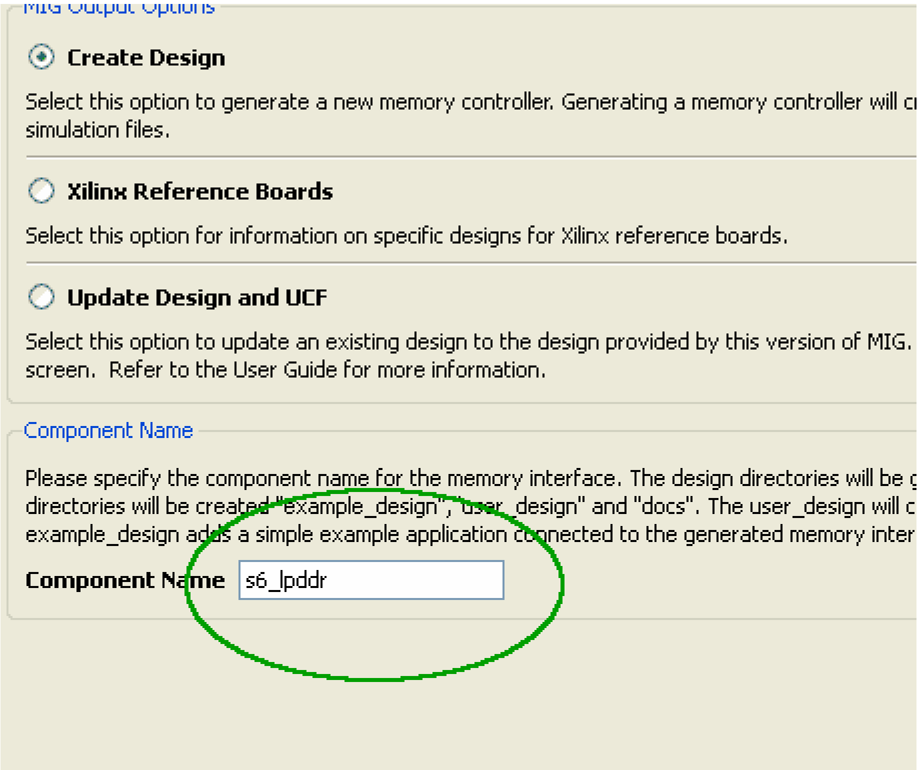
Double click to run Memory Interface Generator wizard. On the first screen, make sure that the selected FPGA device and other settings are correct. The settings should look like in the imagbelow (if you are using LX25 version).



FPGA Family	Spartan-6
FPGA Part	xc6slx25-csg324
Speed Grade	-2
Synthesis Tool	ISE
Design Entry	VHDL

If any of these options are incorrect, please click on "Cancel", change the CORE Generator Project Options,

Click next to proceed to the next screen and type in a component name if necessary. Leaving the default name should work fine as well. For the sake of clarity, I'll call my component "s6_lpddr". Please replace this with the component name you chose when "s6_lpddr" appears later in this tutorial.



MIG Output Options

● Create Design

Select this option to generate a new memory controller. Generating a memory controller will c
simulation files.

○ Xilinx Reference Boards

Select this option for information on specific designs for Xilinx reference boards.

○ Update Design and UCF

Select this option to update an existing design to the design provided by this version of MIG.
screen. Refer to the User Guide for more information.

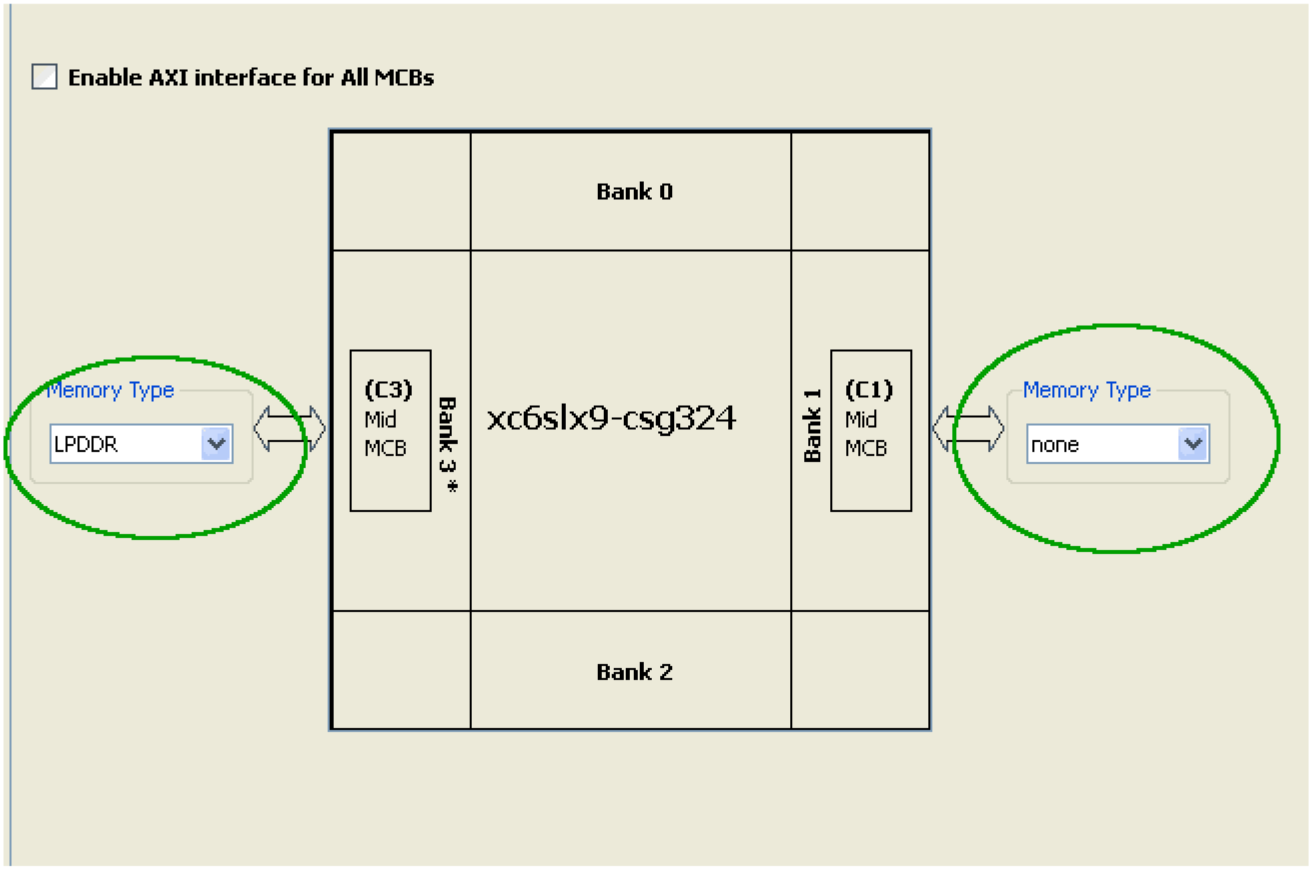
Component Name

Please specify the component name for the memory interface. The design directories will be g
directories will be created "example_design", "user_design" and "docs". The user_design will c
example_design adds a simple example application connected to the generated memory inter

Component Name: s6_lpddr

Click next to go to screen three of the wizard. Leave all options unchanged and proceed to screen 4. This screen is where we select the type of DDR memory and tell the wizard where it is connected. Spartan 6 LX device has two memory controllers available. On this FPGA module, the LPDDR device is connected to Bank 3 of the

FPGA. Select LPDDR from the combo box corresponding to Bank 3. Leave Bank 1 settings unchanged. Settings on this page should look like as in the image below.

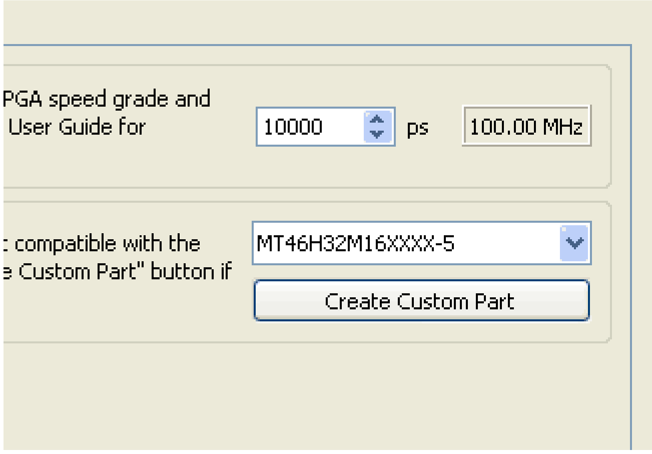


□ Enable AXI interface for All MCBs

Memory Type: LPDDR

Memory Type: none

Click next to go to the next screen. This is the place where we select the DDR memory device and its operating frequency. This board has onboard LPDDR memory which is Micron MT46H32M16 or equivalent. This device supports DDR clock up to 166MHz. Select MT46H32M16 memory device and set the clock period to 10,000. Clock period 10,000 corresponds to 100MHz DDR clock frequency. Though the DDR device supports up to 166MHz clock, we will use 100MHz to avoid the complication of messing with the PLL settings later. This board has a 100MHz clock source and by using the same frequency for DDR clock, we can leave the PLL settings generated by MIG as is. Below is the image with correct memory part and frequency selected.



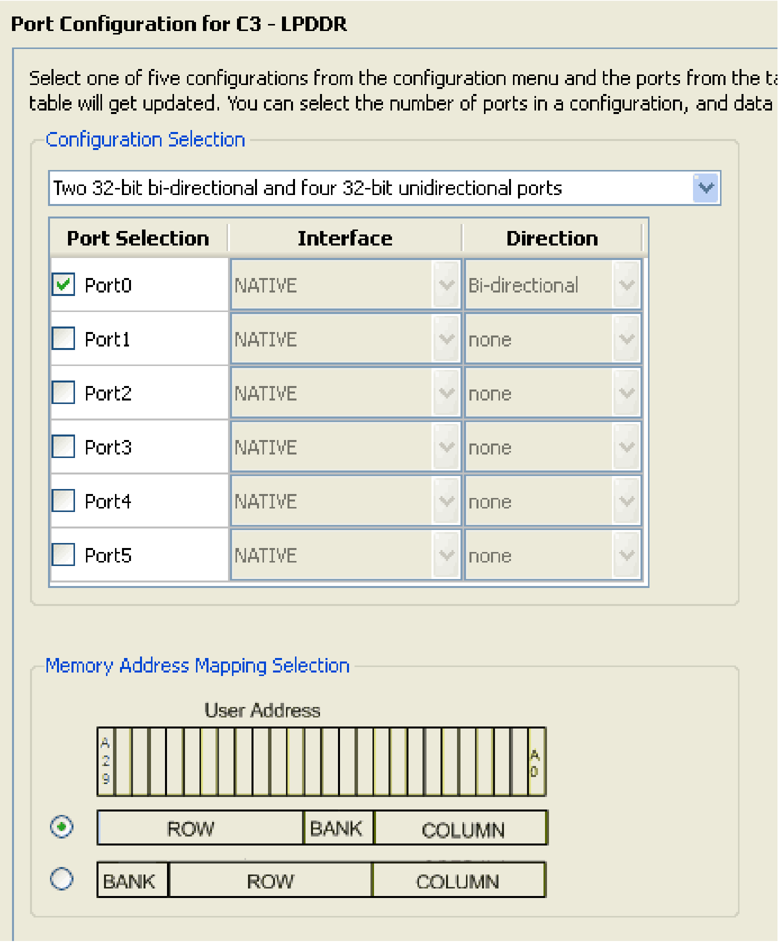
PGA speed grade and
User Guide for

10000 ps 100.00 MHz

: compatible with the
e Custom Part" button if

MT46H32M16XXXX-5

Click next to proceed to the next screen and leave all settings to its defaults. Click next again to proceed to the port configuration screen. Select Port 0 and leave rest of the ports unchecked as shown below.



Port Configuration for C3 - LPDDR

Select one of five configurations from the configuration menu and the ports from the ta
table will get updated. You can select the number of ports in a configuration, and data

Configuration Selection

Two 32-bit bi-directional and four 32-bit unidirectional ports

Port Selection	Interface	Direction
☑ Port0	NATIVE	Bi-directional
□ Port1	NATIVE	none
□ Port2	NATIVE	none
□ Port3	NATIVE	none
□ Port4	NATIVE	none
□ Port5	NATIVE	none

Memory Address Mapping Selection

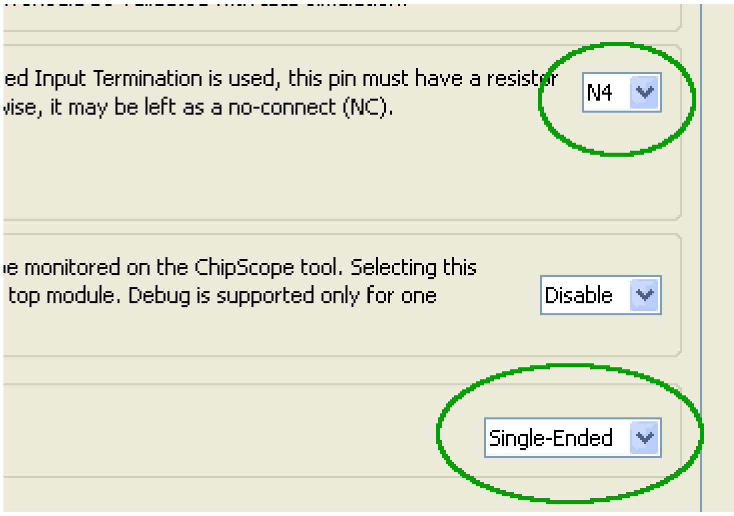
User Address

A29 … A0

● ROW | BANK | COLUMN

○ BANK | ROW | COLUMN

Click next to proceed to the arbitration configuration page. Since we are using only one port, there are no parameters to change on this screen. Click next again to proceed to FPGA Options screen. Select N4 as RZQ pin location and select Single Ended as system clock input. See image below with correct settings.



ed Input Termination is used, this pin must have a resistor
wise, it may be left as a no-connect (NC).

N4

e monitored on the ChipScope tool. Selecting this
top module. Debug is supported only for one

Disable

Single-Ended

Click the next button a few more times and finish. Core Generator will generate a bunch of files.These files can be found under the directory \ipcore_dir\s6_lpddr (Assuming you used the name “s6_lpddr” for the auto generated component). You will see three folders here, docs, example_design and user_design. docs folder has some very important documentation that can be used to learn more about Spartan 6 Memory Controller and the IP generated by MIG. Keep them for a later read.

Editing user constraints

Right now we are going to use the example design generated by MIG. The example design can be found (unsurprisingly) under the folder example_design. There are a few folders and files inside the example_design folder. rtl folder has all the verilog files generated by MIG. par folder contains some batch files and scripts to build the example design. The user constraints seems to be autogenerated to match with Xilinx’s own development boards. Some changes are necessary to make the autogenerated code to work. Before building the project, we need to do the following.

1. Edit the ucf file to make it usable
2. Configure the build environment to generate binary configuration file

Go to par folder and find example_top.ucf. Open example_top.ucf using any text editor of your choice. Make the following changes.

1. Change the line CONFIG VCCAUX=2.5; to CONFIG VCCAUX=3.3; . This change is necessary because this board uses 3.3V for VCCAUX.
2. Change the following lines

```
NET  "error"                      IOSTANDARD = MOBILE_DDR;
NET  "calib_done"                 IOSTANDARD = MOBILE_DDR;
NET  "calib_done"                 LOC = "B2";
NET  "error"                      LOC = "A2";
```

To

```
NET  "error"                      IOSTANDARD = LVCMOS33;
NET  "calib_done"                 IOSTANDARD = LVCMOS33;
NET  "calib_done"                 LOC = "C2";
NET  "error"                      LOC = "L6";
```

Above changes will make the “error” and “calib_done” pins are connected to RGB LED

Error pin is connected to red color of LED1 & calib_done pin is connected to green color of LED2

3. Change the lines

```
  4. NET  "c3_sys_clk"                      IOSTANDARD = LVCMOS25;
  5. NET  "c3_sys_rst"                      IOSTANDARD = LVCMOS18;
```

TO

```
NET  "c3_sys_clk"                          IOSTANDARD = LVCMOS33;
NET  "c3_sys_rst"                          IOSTANDARD = LVCMOS33;
```

Above change will set the IO standards for clock input and reset input to LVCMOS33. This is again because the bank that these IO belongs is powered by 3.3V rail.

4. This will kep the DDR IP out of reset without having to use any external components. Despite the name “c3_sys_rst”, MIG seems to be configuring reset input as active high.

5. Change “c3_sys_clk” and “c3_sys_rst” pin assignments as below.

```
NET  "c3_sys_clk"                          LOC = "V10";
NET  "c3_sys_rst_n"                       LOC = "U16";
```

This code change will assign correct IO pads for clock input and reset input.

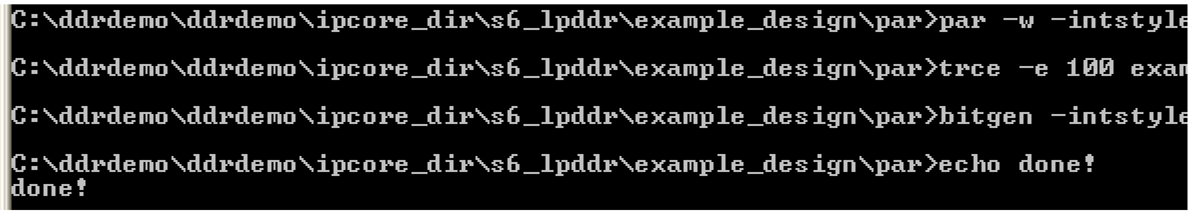
We are done with the changes in ucf file now. This may seem difficult but easy enough if done carefully. It is a good idea to backup your original ucf file before saving the changes just in case if you want to go back and restart again.

Building the code

Next step is to modify the build environment to generate binary configuration file. This is a very easy step to do. Find the file mem_interface_top.ut and open it in a text editor. Find the line “-g Binary:no” and change it to “-g Binary:yes” and save.

Now we are ready to build the project. Before building the project, make sure that the path to Xilinx build tools is added to PATH environment variable. Usually the path is C:\Xilinx\<ISE Version>\ISE_DS\ISE\bin\nt assuming ISE is installed on C: drive.

Now run the batch file ise_flow.bat by double clicking the file or by using command prompt. If everything went fine so far, the batch file will run the necessary tools to build the project and you will end up with a “Done” message and a bunch of new files in the par folder. The message should look like in the image below.



```
C:\ddrdemo\ddrdemo\ipcore_dir\s6_lpddr\example_design\par>par -w -intstyle

C:\ddrdemo\ddrdemo\ipcore_dir\s6_lpddr\example_design\par>trce -e 100 exam

C:\ddrdemo\ddrdemo\ipcore_dir\s6_lpddr\example_design\par>bitgen -intstyle

C:\ddrdemo\ddrdemo\ipcore_dir\s6_lpddr\example_design\par>echo done!
done!
```

If the build process fails, refer to ise_flow_results.txt for more details on the causes of failure. You should see the file “example_top.bit” in par folder if build succeeded. This is the file we are going to program in Spartan 6 Module

Testing if the example program we built, is very easy to do. You may remember we changed a few lines in the ucf file. Below are two of those lines.

```
NET  "error"                    IOSTANDARD = MOBILE_DDR;
NET  "calib_done"               IOSTANDARD = MOBILE_DDR;
NET  "calib_done"               LOC = "C2";
NET  "error"                    LOC = "L6";
```

We have connected Error pin to red color of LED1 & calib_done pin to green color of LED2

The first line assigns the FPGA pin C2 to the net “calib_done” net and the second line assigns FPGA pin L6 to “error” net. To verify proper functioning of the example program, all that we need to do is to check and make sure that “calib_done” goes high after power up, indicating successful completion of initial calibration and “error” stays low. This check can be done by just viewing color. Referring to the sp6_lpddr_v1module user guide, it is easy to find out that C2 (LED2_G) and L6 (LED1_R) .

All sources code can be download from following link

https://github.com/fpgatechsolution/Spartan6_LPDDR