Lab Report No 13



### Digital System Design Lab

Submitted by: **Naveed Ahmad** Registration No.: **22PWCSE2165**

Class Section: **B**

Submitted to:

### Engr. Shah Zada Fahim Jan

Sunday, June 1, 2025

**LAB 13**

**DDR SDRAM**

**Objectives:**

The objective is to understand how **DDR SDRAM** works and how it is used in memory systems. It helps us learn how data is stored and transferred faster in modern computers by using both edges of the clock signal

### **Working :**

* **DDR SDRAM** transfers data **twice per clock cycle** – once on the **rising edge** and once on the **falling edge** of the clock.
* This makes it **faster** than regular SDRAM.
* It is **synchronous**, meaning it works with a clock signal.
* Data is stored in **rows and columns** inside memory banks.
* When the CPU requests data, DDR SDRAM **activates a row**, reads or writes data quickly, then moves to the next operation.

**DDR SDRAM**: High-speed, volatile memory used for storing data in FPGA-based applications.

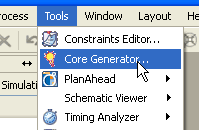
* **Memory Controller**: A logic block (hard or soft) that manages communication between FPGA and DDR.
* **Wrapper Logic**: Bridges user logic to memory controller with a simplified interface.
* **User Logic**: Your custom logic that reads/writes to memory.

Spartan-6 LX9 (CSG324) includes **two built-in DDR memory controllers**; one is connected to onboard LPDDR on Mimas V2. Availability of sufficient data storage is a very important factor when selecting an FPGA development board. Various boards offer different memory options such as SRAM, QDR, SDRAM, FLASH etc. DDR SDRAM is most popular of all since it offers a reasonably large amount of volatile storage that can be accessed at a reasonable speed. An onboard volatile memory is key for a lot of applications such as image processing, data logging etc. since the Block RAM available in the FPGA logic fabric is very limited. Start Xilinx ISE and select a new project from the 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. Mimas V2 has a Spartan 6 LX9 device (XC6SLX9). Settings, as shown in the image below, should work fine.

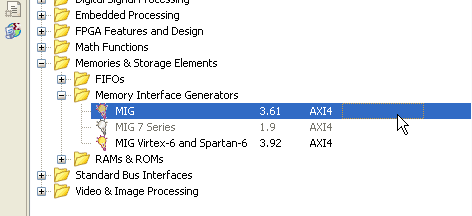
A screenshot of a project wizard

AI-generated content may be incorrect.

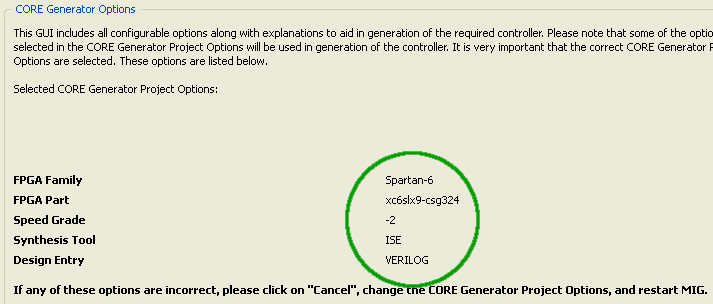
Click next and finish the wizard. Surprisingly we are not going to create or add any source files to the project and our use of ISE’s graphical user interface ends once we generate the required IP and other files using Coregen tool. To start Coregen tool, go to the Tools menu and select “Core Generator”.

****

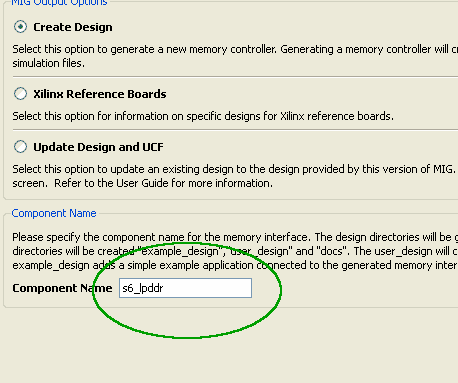
If Core Generator does not create a project automatically, create a project by selecting File>New Project. You will need to select the FPGA and its package when creating the project. Select Verilog as a design entry method. This is to make sure that Core Generator generates code in Verilog. In the IP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

****

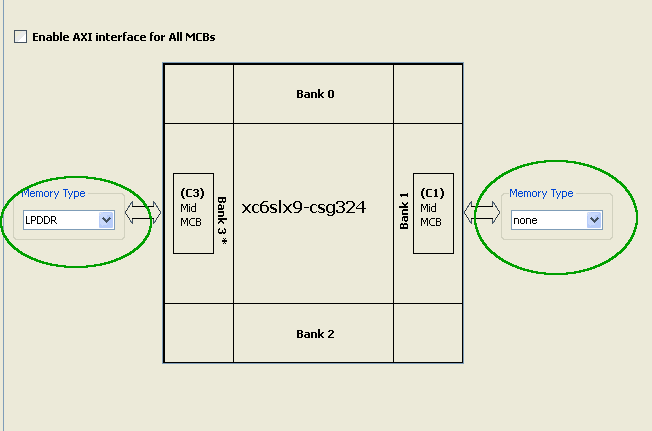
Double click to run Memory Interface Generator wizard. On the first screen, make sure that the selected FPGA device and other settings are correct. The settings should look like in the image below.

****

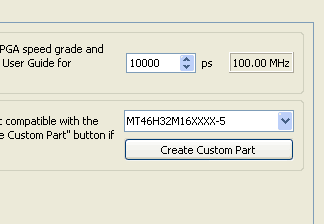
Click next to proceed to the next screen and type in a component name if necessary. Leaving the default name should work fine as well. For the sake of clarity, we will call our component “s6\_lpddr”. Please replace this with the component name you chose when “s6\_lpddr” appears later in this tutorial.

****

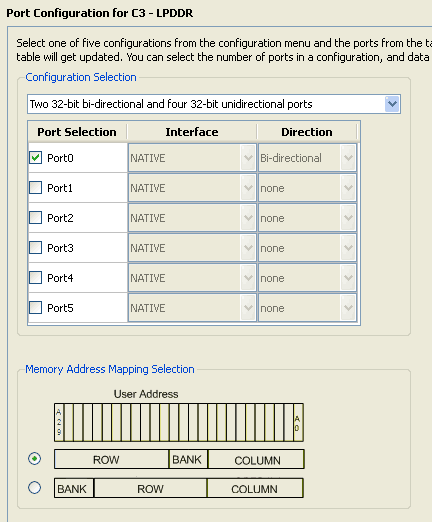
Click next to go to screen three of the wizard. Leave all options unchanged and proceed to screen 4. This screen is where we select the type of DDR memory and tell the MIG wizard where it is connected. Spartan 6 LX9 device has two memory controllers available. On Mimas V2 FPGA Development Board, the 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.

****

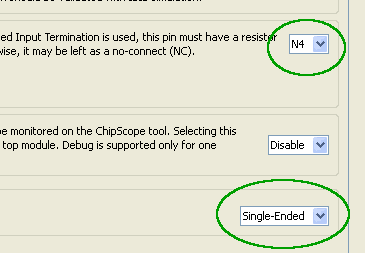
Click next to go to the next screen. This is the place where we select the DDR memory device and its operating frequency. Mimas V2 has onboard LPDDR memory which is Micron MT46H32M16 or equivalent. This device supports DDR clock up to 166MHz. Select MT46H32M16 memory device and set the clock period to 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. Mimas V2 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.

****

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.

****

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.

****

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 seem to be auto-generated to match with Xilinx’s own development boards. Some changes are necessary to make the auto-generated code work with Mimas V2. Before building the project, we need to do the following.

1. Edit the ucf file to make it usable with Mimas V2
2. Configure the build environment to generate a 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 Mimas V2 uses 3.3V for VCCAUX.
2. Change the following lines  
   NET “error” IOSTANDARD = LVCMOS18;  
   NET “calib\_done” IOSTANDARD = LVCMOS18;  
   NET “calib\_done” LOC = “B2” ;  
   NET “error” LOC = “A2” ;

To

1. NET “error” IOSTANDARD = LVCMOS33;  
   NET “calib\_done” IOSTANDARD = LVCMOS33;  
   NET “calib\_done” LOC = “T18” ; #LED1  
   NET “error” LOC = “T17” ; #LED2

Above changes will make the “error” and “calib\_done” pins operate at LVCMOS33 IO standard and assign the nets to T18 and T17 of FPGA where LED1 and LED2 are connected. This will cause LED1 to turn on when calibration is done and LED2 to turn on if memory test failed

1. Change the lines

NET “c3\_sys\_clk” IOSTANDARD = LVCMOS25;  
NET “c3\_sys\_rst\_n” IOSTANDARD = LVCMOS18;

To

NET “c3\_sys\_clk” IOSTANDARD = LVCMOS33;  
NET “c3\_sys\_rst\_n” 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.

1. Add a line NET “c3\_sys\_rst\_n” PULLDOWN;

This will enable pull down on the reset pin and will keep the memory controller our of reset without having to use any external components. Despite the name “c3\_sys\_rst\_n”, MIG seems to be configuring reset input as active high.

1. Change “c3\_sys\_clk” and “c3\_sys\_rst\_n” pin assignments as below.  
   NET “c3\_sys\_clk” LOC = “V10” ;  
   NET “c3\_sys\_rst\_n” LOC = “M16” **;**

This will assign correct IO pads for clock input and assign switch SW3 as reset input.

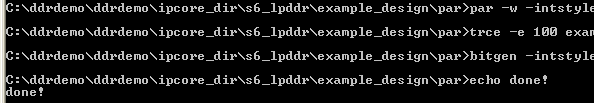
We are done with the changes in ucf file now. This may seem difficult but easy enough if done carefully. It is a good idea to back up your original ucf file before saving the changes just in case if you want to go back and restart again.

**Building the code**

Next step is to modify the build environment to generate a 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\_DS\ISE\bin\nt assuming ISE is installed on C: drive.

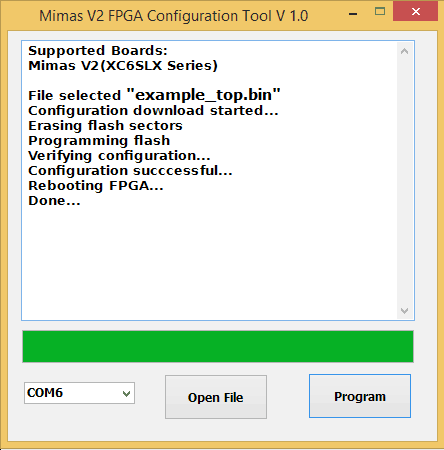
Now run the batch file ise\_flow.bat by double-clicking the file or by using the command prompt. If everything went fine so far, the batch file will run the necessary tools to build the project and you will end up with a “Done” message and a bunch of new files in the par folder. The message should look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

****

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.bin” in par folder if build succeeded. This is the file we are going to program Mimas V2 Spartan 6 FPGA development board with.

**Configuring Mimas V2 and testing**

Programming Mimas V2 Spartan 6 FPGA development board is very easy. Download the latest version of the Configuration Downloader Application from the product page. Run the executable, no installation required. Load the binary file and program the flash as illustrated in the image below.

****

**CONCLUSION:**

The DDR SDRAM successfully stored and transferred data at high speed by using both the rising and falling edges of the clock signal. The system demonstrated faster data transfer compared to regular SDRAM, showing how DDR technology improves memory performance. The output confirmed correct read and write operations from memory banks with proper synchronization.