Laboratory Exercise 8

In computer systems it is necessary to provide a substantial amout of memory. If a system is implemented using the FPGA technology it is possible to provide some amount of memory by using the memory resources that exist in the FPGA device. If additional memory is needed, it has to be implemented by connecting external memory chips to the FPGA. In this exercise we will examine the general issues involved in implementing such memory. First, we will investgate how memory can be implemented by using the resources in the FPGA device. Next, we will make use of the memory chips on the Altera DE2 board.

Part I

Commonly used logic structures, such as adders, registers, counters and memories, can be implemented in an FPGA chip by using LPM modules from the Quartus II Library of Parameterized Modules. Altera recommends that a random access memory (RAM) be implemented by using the *altsyncram* LPM. Use this LPM to implement a memory that has 32 words with 8 bits per word, as follows:

1. Create a new project, called *ramlpm*, which will be used to implement the memory module.
2. Use the MegaWizard Plug-in Manager to generate the desired RAM. You can learn how a specific LPM module is generated by reading the tutorial *Using Library Modules in Verilog Designs*. Give the Verilog file generated by the MegaWizard the name *ramlpm.v* to match the name of the project.
3. Compile the circuit. Observe in the Compilation Report that Quartus II Compiler used 256 bits in one of the memory blocks available in the Cyclone II chip to implement the RAM circuit.
4. Simulate the behavior of your circuit.

Part II

Now, we want to use toggle switches on the DE2 board to load some data into the created RAM. We also want to display the contents of the RAM on the 7-segment displays.

1. Create a new project, called *fpgaram*, which will be used to implement the desired circuit on the DE2 board.
2. Write a Verilog file, called *fpgaram.v*, which provides the necessary functionality. Include the *ramlpm* module in your file. Use toggle switches SW_{7-0} to input a byte of data into the RAM location identified by a 5-bit address specified with toggle switches SW_{15-11}. Make sure that all 32 locations can be loaded in this manner. To see the RAM contents, add to your design a capability to display the contents of each byte (in hexadecimal format) on the 7-segment displays $HEX1$ and $HEX0$. Scroll through the bytes by displaying each byte for two seconds. As each byte is being displayed, show its address (in hex format) on the 7-segment displays $HEX5$ and $HEX4$.
3. Include the Verilog file in your project and compile the circuit.
4. Simulate the behavior of your circuit.
5. Assign the pins on the FPGA to connect to the switches and the 7-segment displays, as indicated in the User Manual for the DE2 board.
6. Recompile the circuit and download it into the FPGA chip.
7. Test the functionality of your design by loading and displaying at least 8 bytes of data.

Part III

Instead of using the LPM module, we can implement the required memory by specifying its structure in the Verilog code. In a Verilog-specified design it is possible to define the memory as a multidimensional array. A 32-by-8 array, which has 32 words with 8 bits per word, can be declared by the statement

$$\text{reg } [7:0] \text{ memory\_array } [31:0];$$

In the Cyclone II FPGA, such an array can be implemented either by using the flip-flops that each logic element contains or, more efficiently, by using the specialized memory blocks available in the chip. There are two ways of ensuring that the memory blocks will be used. One is to use an LPM module from the Library of Parameterized Modules, as we saw in Part I. The other is to define the memory requirement by using a suitable style of Verilog code from which the Quartus II compiler can infer that a memory block should be used. Quartus II Help shows how this may be done.

1. Create a new project which will be used to implement the desired circuit on the DE2 board.
2. Write a Verilog file that provides the necessary functionality, including the ability to load the RAM and read its contents as done in Part II.
3. Include the Verilog file in your project and compile the circuit.
4. Simulate the behavior of your circuit.
5. Assign the pins on the FPGA to connect to the switches and the 7-segment displays.
6. Recompile the circuit and download it into the FPGA chip.
7. Test the functionality of your design by applying some inputs and observing the output.

Part IV

The DE2 board includes an SRAM chip, which is a static RAM having a capacity of 256K 16-bit words. The SRAM interface consists of an address bus and a data bus. It also has the following active-low control signals:

- *Chip Enable* ($\overline{CE}$) is asserted (low) during all SRAM operations.
- *Write Enable* ($\overline{WE}$) is asserted during a write opration.
- *Upper Byte* ($\overline{UB}$) and *Lower Byte* ($\overline{LB}$) signals are asserted during a read or write operation when there is valid data on the bus.

For a read operation, valid data is presented on the data bus one clock cycle after the appropriate signals are asserted. A write operation is completed in one cycle.

1. Create a new project, called *sram*, which will be used to implement the SRAM controller on the DE2 board.
2. Write a Verilog file, called *sram.v*, which provides the necessary functionality. Use toggle switches SW_{7-0} to input a byte of data into the RAM location identified by an 8-bit address specified with SW_{15-8} (this is only a subset of the SRAM capacity). Use KEY_0 to initiate a read operation and KEY_1 to initiate a write operation. Display the value read from the SRAM on the 7-segment displays $HEX1$ and $HEX0$.
3. Assign the pins on the FPGA to connect to the switches, 7-segment displays, and SRAM.
4. Compile the circuit and download it into the FPGA chip.
5. Test the functionality of your design by reading and writing values to several different memory locations.