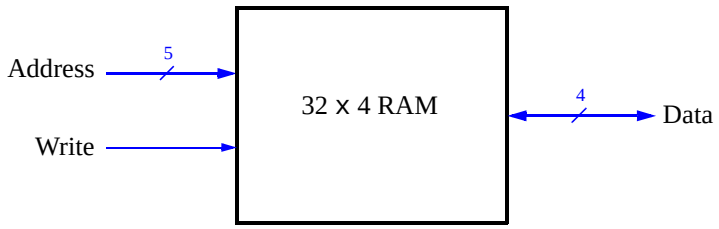
Laboratory Exercise 8

Memory Blocks

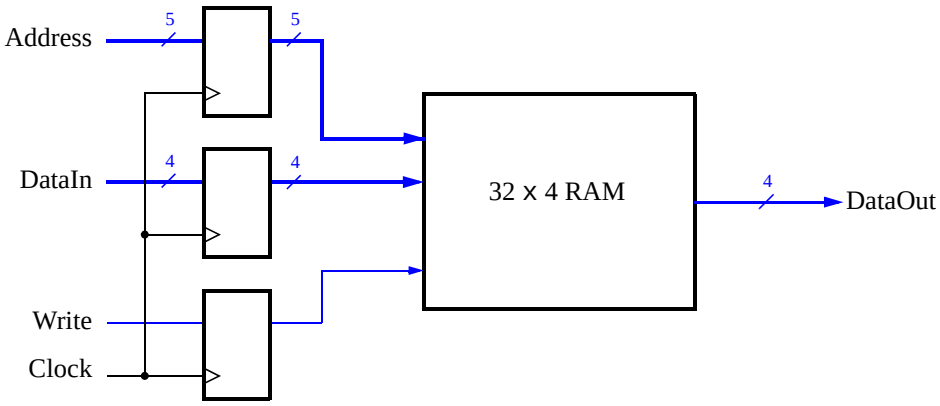
In computer systems it is necessary to provide a substantial amount of memory. If a system is implemented using FPGA technology it is possible to provide some amount of memory by using the memory resources that exist in the FPGA device. In this exercise we will examine the general issues involved in implementing such memory.

A diagram of the random access memory (RAM) module that we will implement is shown in Figure 1*a*. It contains 32 four-bit words (rows), which are accessed using a five-bit *address* port, a four-bit *data* port, and a *write* control input.

The FPGAs that are included on the Intel FPGA DE10-Lite, DE0-CV, DE1-SoC, and DE2-115 boards provide dedicated memory resources. The MAX 10 FPGA on the DE10-Lite, and Cyclone IV FPGA on the DE2-115 contain dedicated memory resources called *M9K blocks*. The Cyclone V FPGA on the DE0-CV and DE1-SoC boards have *M10K blocks*. Each M9K block contains 9216 memory bits, while each M10K block contains 10240 memory bits. Both M9K and M10k blocks can be configured to implement memories of various sizes. A common term used to specify the size of a memory is its *aspect ratio*, which gives the *depth* in words and the *width* in bits (depth x width). In this exercise we will use an aspect ratio that is four bits wide, and we will use only the first 32 words in the memory. Although the M9K and M10K blocks support many other modes of operation, we will not discuss them here.



(*a*) RAM organization



(b) RAM implementation

Figure 1: A 32 x 4 RAM module.

There are two important features of the memory blocks that have to be mentioned. First, they includes registers that can be used to synchronize all of the input and output signals to a clock input. The registers on the input ports must always be used, and the registers on the output ports are optional. Second, the blocks have separate ports for data being written to the memory and data being read from the memory. Given these requirements, we will implement the modified 32 x 4 RAM module shown in Figure 1*b*. It includes registers for the *address*, *data input*, and *write* ports, and uses a separate unregistered *data output* port.

Part I

Commonly used logic structures, such as adders, registers, counters and memories, can be implemented in an FPGA chip by using prebuilt modules that are provided in *libraries*. In this exercise we will use such a module to implement the memory shown in Figure 1*b*.

1. Create a new Vivado project to implement the memory module.
2. To open the IP Catalog in the Vivado software click on Project Manager > IP Catalog. In the IP Catalog window choose the Memories & Storage elements, which is found under the Vivado Repository.
3. Click to RAMs & ROMs & BRAMs and Block Memory Generator.
4. Choose Single Port Ram and specify the Port A options as requested.
5. Instantiate this subcircuit in a top-level Verilog file that includes appropriate input and output signals for the memory ports given in Figure 1b. Compile the circuit.
6. Simulate the behavior of your circuit and ensure that you can read and write data in the memory. An example simulation output is given in Figure 4.

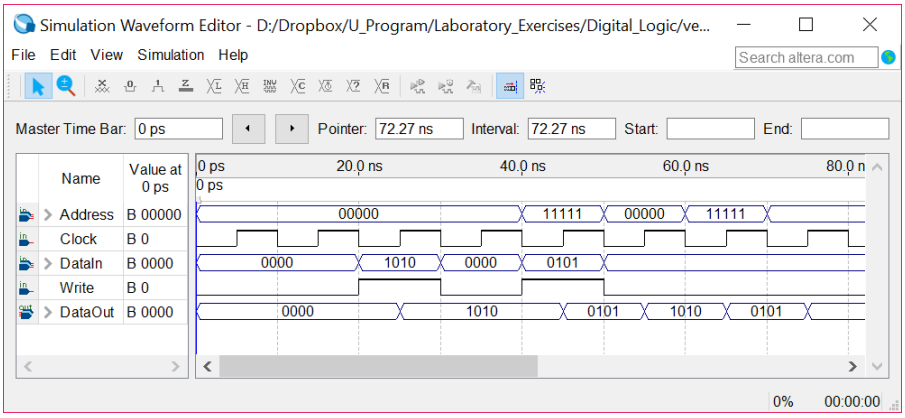


Figure 4: An example of simulation output.

Part II

Now, we want to realize the memory circuit in the FPGA on your board, and use slide switches to load some data into the created memory. We also want to display the contents of the RAM on the 7-segment displays.

1. Make a new Vivado project which will be used to implement the desired circuit on your board.
2. Create another Verilog file that instantiates the *ram32x4* module and that includes the required input and output pins on your board. Use slide switches SW_{3-0} to provide input data for the RAM, and use switches switches SW_{8-4} to specify the address. Use SW_9 as the *Write* signal and use KEY_0 as the *Clock* input. Show the address value on the 7-segment displays *HEX*$5-4$, show the data being input to the memory on *HEX*2, and show the data read out of the memory on *HEX*0.
3. Test your circuit and make sure that data can be stored into the memory at various locations.

Part III

Instead of creating a memory module subcircuit by using the IP Catalog, we can implement the required memory by specifying its structure in Verilog code. In a Verilog-specified design it is possible to define the memory as a multidimensional array. A 32 x 4 array, which has 32 words with 4 bits per word, can be declared by the statement

reg [3:0] memory_array [31:0];

In the Cyclone series of FPGAs, such an array can be implemented either by using the flip-flops that each logic element contains or, more efficiently, by using the built-in memory blocks.

Perform the following steps:

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