EELE 367 - Introduction to Logic Circuits

Lab 10(a) – Memory Systems: Reading from ROM

Objective

This lab will give experience with modeling memory systems and also how to use a finite state machine to control the data coming out of a ROM. You will create a 16x4, synchronous read only memory device with an enable and initialize it with a pre-determined set of data. You will then create a FSM that will generate the necessary control signals to read the data out of the ROM. You will display the address and data output on the HEX character displays of the DE0-CV board.

Outcomes

After completing this lab you should be able to:

- Design a 16x4, synchronous read only memory device with enable.
- Design an address counter that can be synchronously enabled and cleared.
- Design a finite state machine to generate the necessary control signals to read the contents of the ROM.

Deliverables

The deliverable(s) for this lab are as follows:

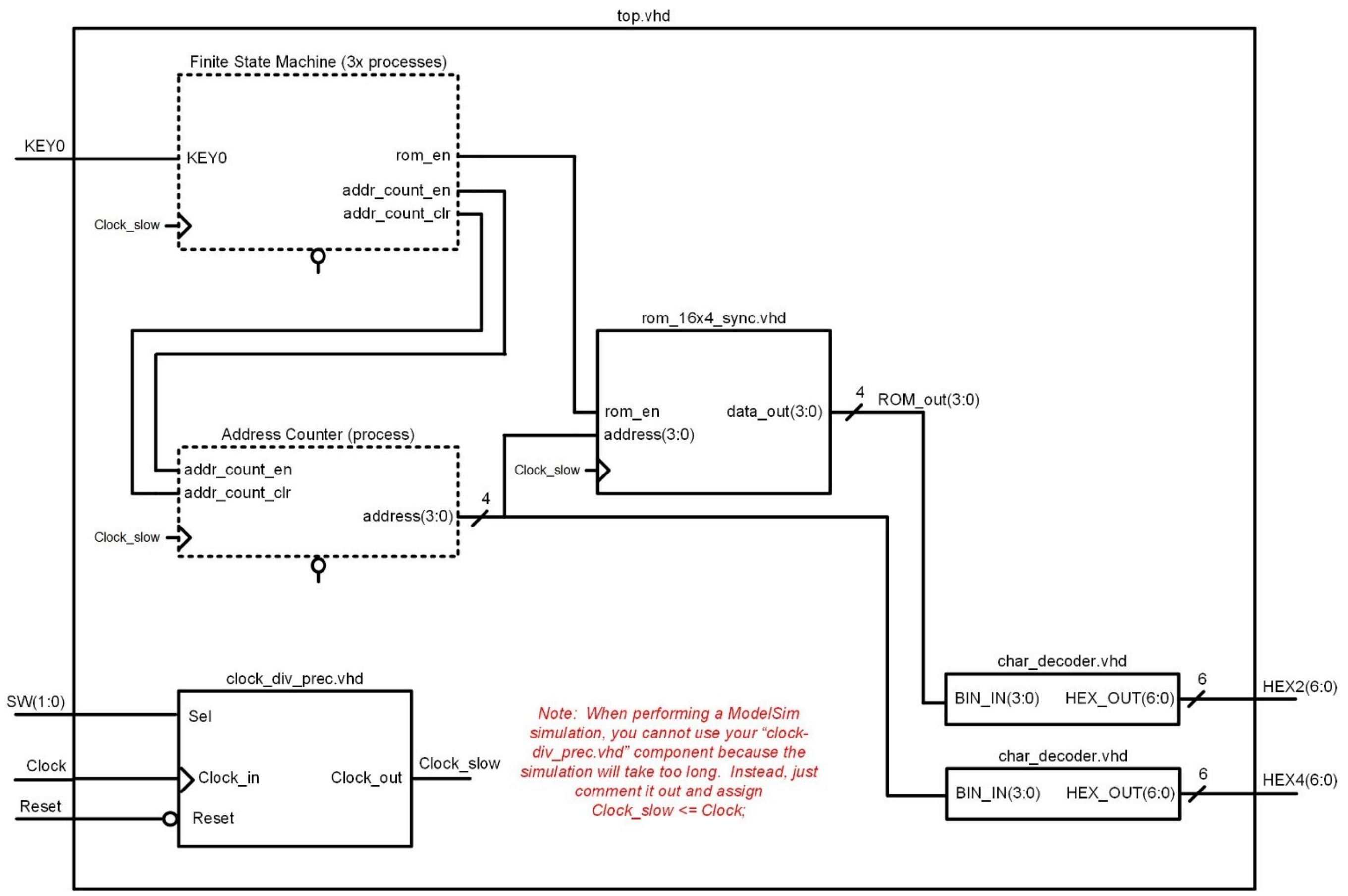
- A ModelSim simulation of your memory system with waveforms uploaded to the lab DropBox (45%).
- Demonstration of your system implemented on the DE0-CV board (45%).
- Upload your top.vhd file to the lab DropBox (10%).

Lab Work & Demonstration

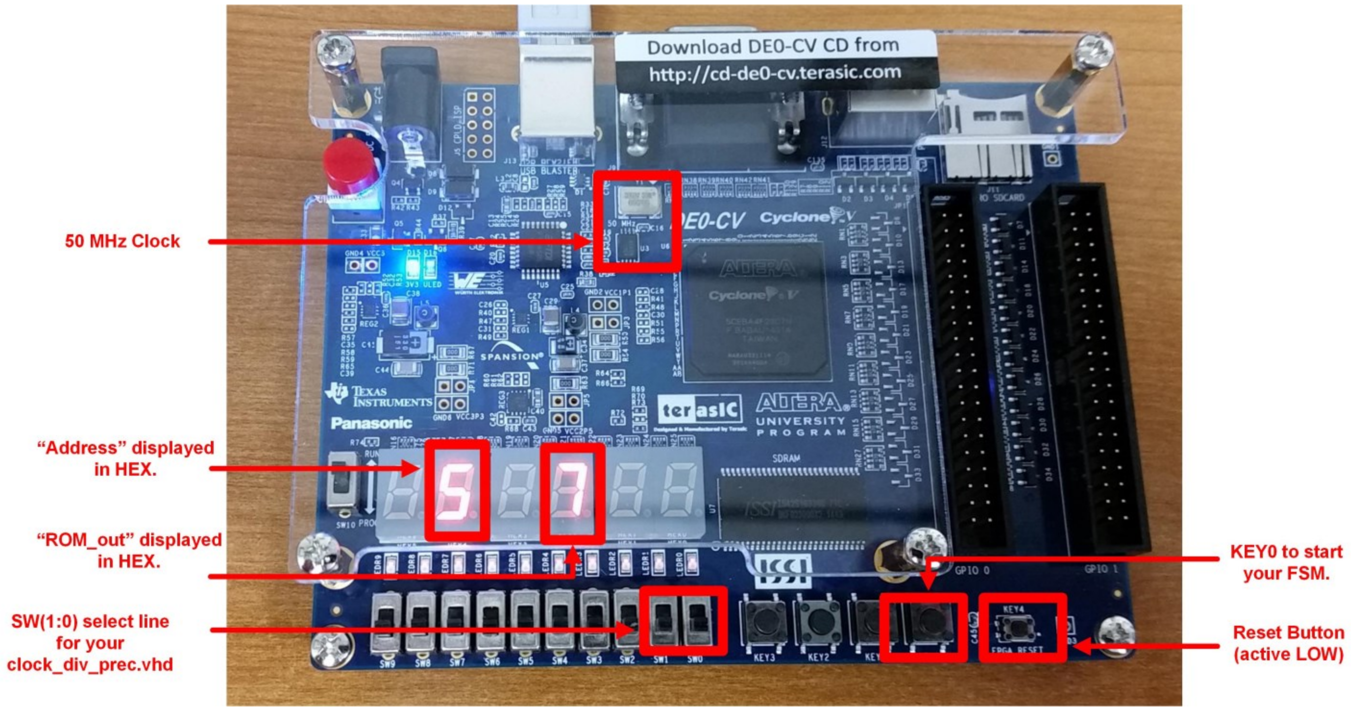
Part 1 – Design and Simulation of your Memory System

You are going to create a system that will automatically read all of the contents of the ROM and display them one-by-one on the HEX2 character display on the DE0-CV board. The corresponding address of each location being read will be displayed on the HEX4 display. The reading will only begin when the KEY0 button is pressed. Once all of the ROM values have been displayed, your system will stop and wait forever. This system is complex enough that you will first need to simulate it with ModelSim before trying to implement it on the FPGA. You are provided with a test bench to verify your system. Note that when you *implement* your system on the FPGA, you will need to use your precision clock divider (clock_div_prec.vhd) so that the address and ROM data can be displayed at a rate that is visible by the human eye. However, when you *simulate* your design, the clock divider can't be used because the simulation will take too long to complete. This is because it will take tens of thousands of clock edges to produce the divided down clock that will actually run your system. To accommodate this, you will simply comment out your clock_div_prec.vhd in your simulation and do a signal assignment of: Clock_slow <= Clock;

The following figure shows the block diagram for the system that you will implement. Notice that this system contains numerous components, a 3-process FSM, and a 1-process counter.



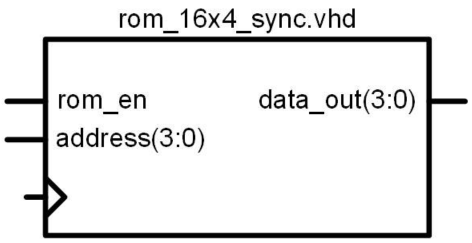
The following shows the DE0-CV I/O that will be used in this lab.


50 MHz Clock
"Address" displayed in HEX.
"ROM_out" displayed in HEX.
SW(1:0) select line for your clock_div_prec.vhd
KEY0 to start your FSM.
Reset Button (active LOW)

16x4 Synchronous ROM with Enable

You are going to create a VHDL model for a 16x4, synchronous read only memory device with enable. The following figure shows the port definition and contents that your ROM should hold. When the enable is asserted, the data_output will produce the contents being held at the given address on the rising edge of clock. When the enable is not asserted, the output will simply hold its last value.

Address	Data
0	x”E”
1	x”E”
2	x”3”
3	x”6”
4	x”7”
5	x”1”
6	x”0”
7	x”6”
8	x”1”
9	x”c”
10	x”2”
11	x”0”
12	x”1”
13	x”6”
14	x”E”
15	x”E”

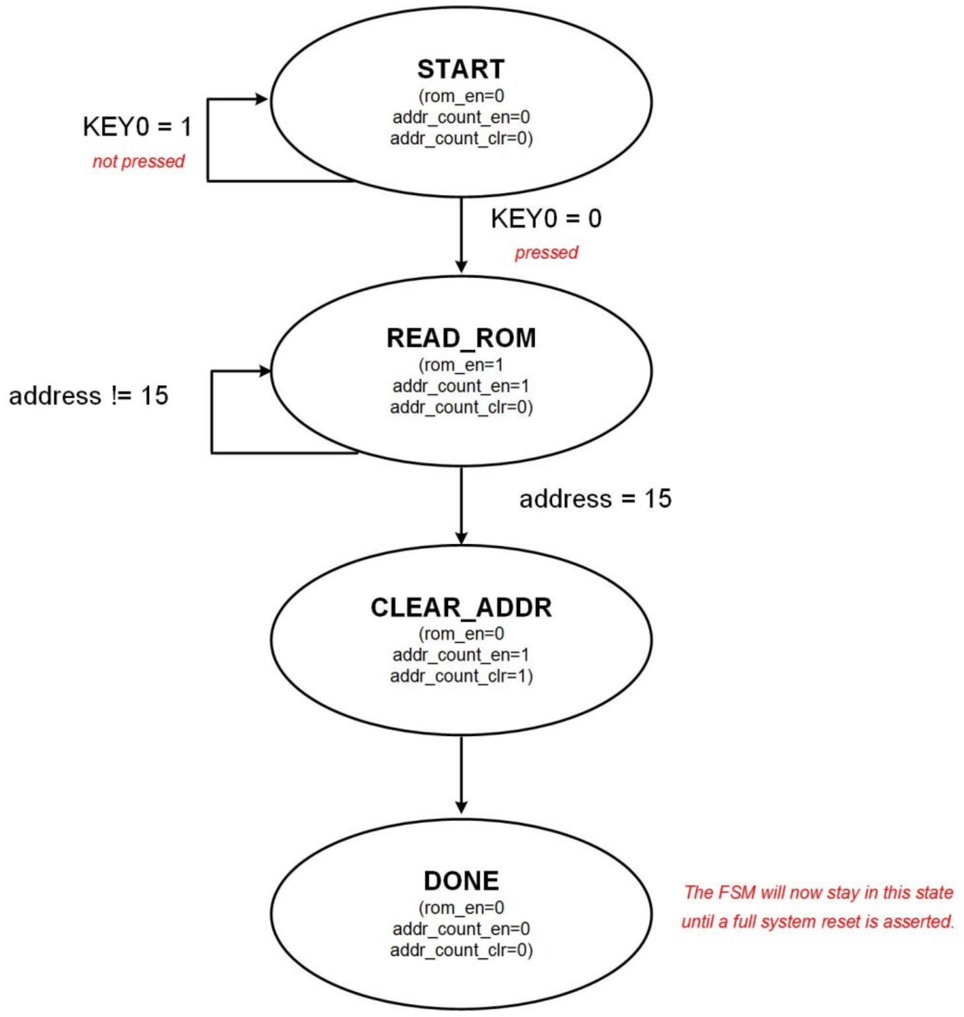


Address Counter

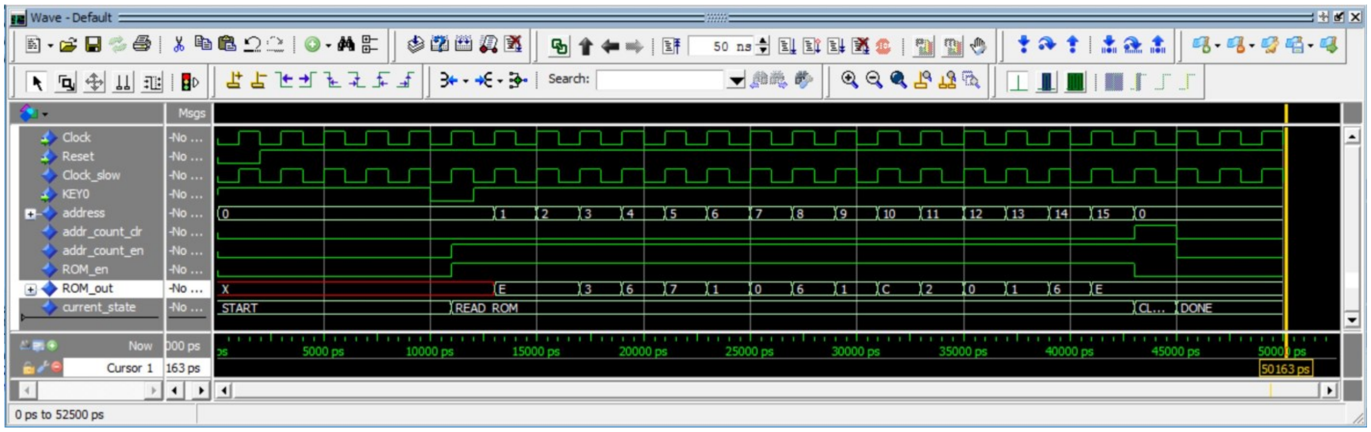
In order to produce the addresses to read from the ROM (i.e., 0,1,2, …, 15), you will need to create an address counter. This can be accomplished with a single process. Your address counter process should have a synchronous enable (addr_count_en) and a synchronous clear (addr_count_clr). When the address counter is disabled, its output will not change. When it is enabled, it will increment by one on each rising edge of the clock. When it is enabled and the clear line is asserted, it will set its counter value back to zero. Note that the clear is synchronous and is only used when the counter is enabled. You will need to create an *unsigned* version of the address so that you can use the “+” defined in the NUMERIC_STD package (for example, *address_uns*). You can use this signal within your address counter process. After the process, you'll need to type cast this unsigned vector to your actual address vector of type *std_logic_vector*.

Control Finite State Machine

In order to control the address counter and enable the ROM at the appropriate time, you will need to have a finite state machine that will generate the control signals for the system. The finite state machine will have three outputs: rom_en, addr_count_en, and addr_count_clr. The following state diagram describes the behavior of your control FSM. The machine will begin in a START state and wait there until the KEY0 is pressed. Once KEY0 is pressed, the FSM will go into a READ_ROM state. In this state it will enable the ROM and address counter. It will reside in READ_ROM until the address counter increments to 15. Once at 15, it will enter a CLEAR_ADDR state in which it will clear the address counter by asserting the synchronous *addr_count_clr* signal. Once cleared, it will enter a DONE state where it will remain forever. The only way to get the system out of the DONE state is to reset the entire system.



A) Design and simulate your memory system. A test bench (top_TB.vhd) is provided. Match your top entity to match what is expected in the top_TB component call. **Remember that you cannot use your clock_div_prec.vhd subsystem in the simulation**. Your simulation should look like this:



B) Once complete, upload your simulation waveform to the lab DropBox.

Part 2 – Implement the Memory System on the DE0-CV FPGA Board

A) Now implement your top.vhd on the DE0-CV board using Quartus. Remember to re-insert your clock_div_prec.vhd so that you can observe the address and ROM data out values in real time.

NOTE: We have not used KEY0 before so you'll need to assign its pin location.

B) Compile your design. Fix any Errors.

C) Download and test your design. Fix any Errors.

D) Demonstrate your design.

DEMO – Show the lab instructor your system. Once checked off, upload your top.vhd file to the lab DropBox.

Lab Grading

Simulation – Review of your simulation waveforms	____________	**/ 45** (uploaded to DropBox)
Demo 2 – Proper operation of your memory system	____________	**/ 45**
Review of your top.vhd file	____________	**/ 10** (uploaded to DropBox)
Total	____________	**/ 100**

Post Lab Survey (Just for your own knowledge. This is not graded)

1) Can you create a synchronous ROM with enable in VHDL?
2) Can you create a counter with synchronous enable and clear lines?
3) Can you create a FSM to generate control signals to read the contents of a ROM?