M A S S A C H U S E T T S I N S T I T U T E O F T E C H N O L O G Y
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

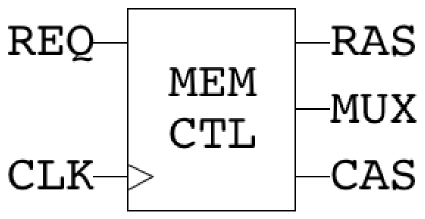
6.111 Introductory Digital Systems Laboratory

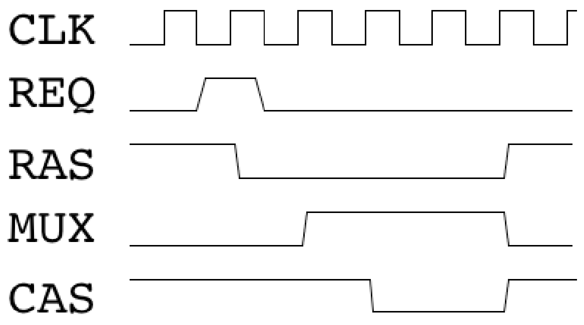
Fall 2017

Lecture PSet #5
Upload as PDF by 14:30 Thu, 09/28/17

This Lecture PSet requires ISE. You can use the lab systems or download ISE WebPACK Design Software from Xilinx. The download is huge but is useful for writing Verilog and running simulation outside of the lab. The Lecture PSet must be submitted online as one pdf. Handwritten solutions will not be accepted except as noted.

Problem 1. FSMs are often used to generate sequences of waveforms necessary to communicate with a component. Memory chips (DRAMs) required several control signals to be asserted in a particular sequence to perform a memory read. Memory is arranged in a grid of columns and rows. Column and row addresses are shared on one address bus to reduce pin count. Row address is presented on the address bus and RAS (Row Address Strobe) is used to strobe in the row address. Some time later, row address is removed and column address is present and CAS (Column Address Strobe) is used to strobe in the column address. The following figure shows a control module and the waveforms it must generate in response to a read request:





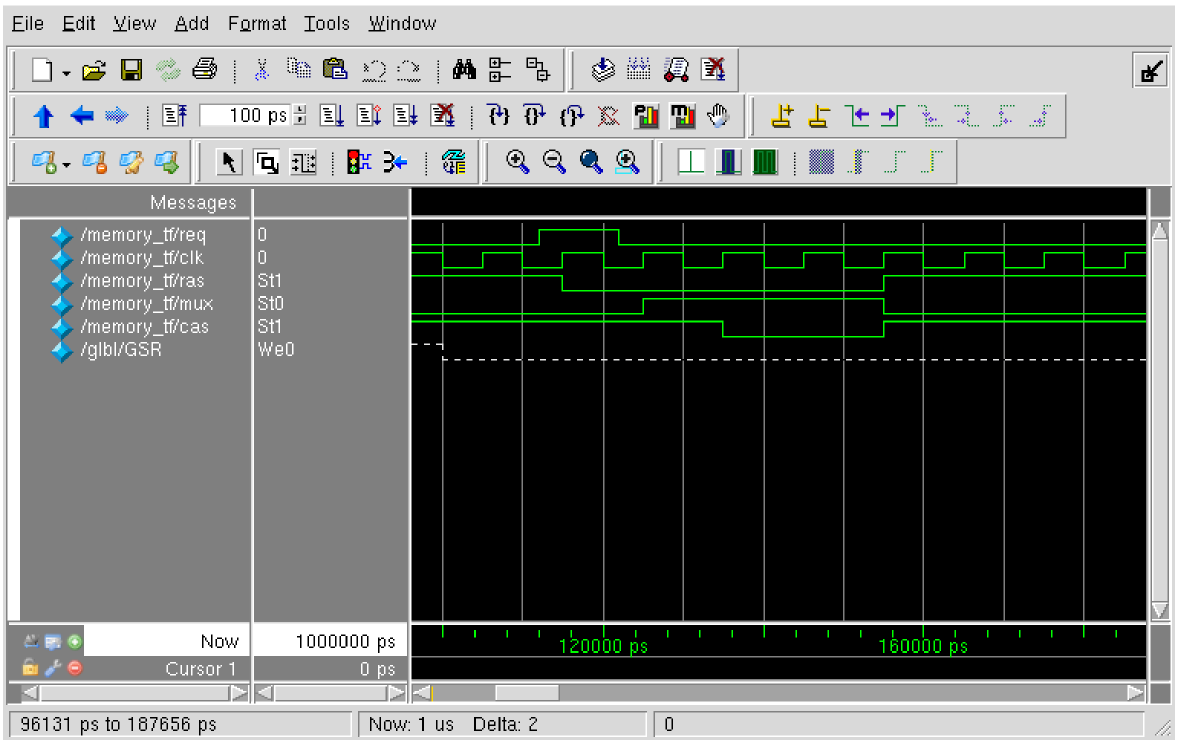
The module sits idle with RAS=1, MUX=0 and CAS=1 until it detects REQ=1 on the rising edge of CLK. In the first cycle of the request it should assert RAS=0, in the second cycle MUX=1 and in the third cycle CAS=0. These signals are held during the fourth cycle and return to their idle values in the fifth cycle. The module then waits for a new request; it ignores requests made while in the middle of processing the last request.

(A) Draw a state transition diagram for a FSM that will generate the appropriate sequence of signals. (Use may draw the diagram using software or import a photo of a hand drawn diagram). [2 points]

(B) Write the Verilog (using ISE) for the module, choosing a state encoding and providing the appropriate combinational logic for generating the next state and output signals. But there's a hitch (this is the real world)! Using combinatorial logic to create RAS, MUX and CAS control signals could result in glitches . This is undesirable for memories. Select a state encoding that will generate glitch free RAS, MUX and CAS signals. There are at least two solutions to this problem. Attach your **Verilog and test bench.** [4 points]

(C) Explain how RAS, MUX and CAS are glitch free. [2 points]

(C) Verify your design by running a simulation using the process outlined in Lab 2 exercise 1(b). (With ISE Webpack, the simulator is ISim.) Note that `req` is asserted some time before the rising edge of `clk.` Attach your test bench and a screenshot of the simulation. Your screenshot should be similar this one. [2 points]



Messages	
/memory_tf/req	0
/memory_tf/clk	0
/memory_tf/ras	St1
/memory_tf/mux	St0
/memory_tf/cas	St1
/glbl/GSR	We0
Now	1000000 ps
Cursor 1	0 ps

96131 ps to 187656 ps   Now: 1 us   Delta: 2   0