University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

EECS150, Spring 2010

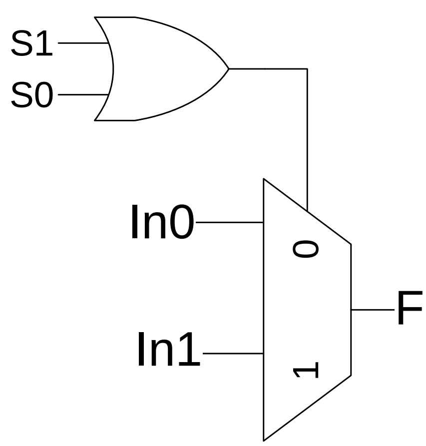
Midterm Review: Sequential Logic, Basic Combinational Logic, Verilog, Video

March 29, 2010

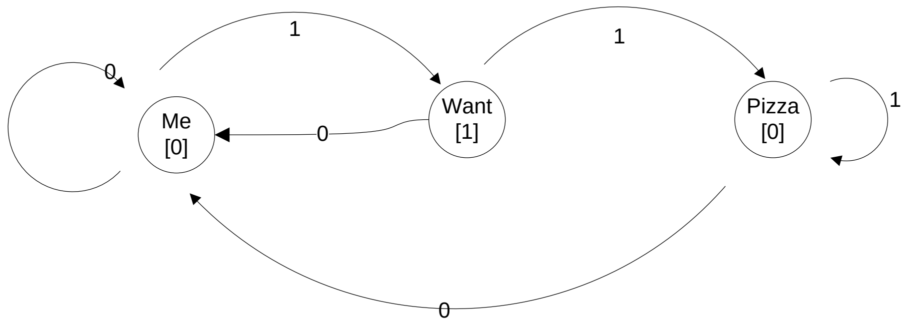
Brandon Myers

This sheet is meant to pack a lot on a little paper. Use a separate piece of paper for your answers.

1. Design a counter with one control input. When the input is high, the counter should sequence through three states: 11, 01, 10 and repeat. When the input is low the counter should sequence through the same states in the opposite order 11, 10, 01 and repeat.
   (a) Implement the counter using D flip flops and whatever gates you like.
   (b) Extra: Draw the state diagram and state transition table
2. Design a 4:16 decoder out of 2:4 decoders and 2-input gates.
3. Adapted from sp07-Mt1-Q3. Suppose we have the basic programmable logic block (instead of LUTs), shown below. All the inputs In0,In1,S0,S1 can be tied to input A, input B, 0, or 1.



   Implement each of the following logic functions using the basic building block: A, A', A NAND B, A NOR B, A AND B, A OR B, A XOR B, A XNOR B.
4. Write a Verilog module for a level sensitive latch, with interface In, Out, and C ("clock"). It must have a parameter Level, which if 1 makes the latch sensitive to high and if 0 makes the latch sensitive to low.
5. Consider the FSM with the state-transition diagram below.



(a) Describe the behavior of the FSM (when does it output 1?).

(b) Write a Verilog module for the FSM.

6. Video. WxH F-fps display (i.e. W pixels/line, H lines, F frames per second). Horizontal blanking interval HB, vertical blanking interval VB, pixel clock frequency PF.
$H * (W/PF + HB) + VB = 1/F$
This equation comes from the fact that the total time to draw a frame (1/F) is equal to the time to draw every line (including HB at end of each line) plus VB.

7. The Bresenham Line Drawing Algorithm (without steepness/direction tests).

```
function line(x0, x1, y0, y1)
int deltax := x1 - x0
int deltay := y1 - y0
int error := deltax / 2
int y := y0
for x from x0 to x1
plot(x,y)
error := error - deltay
if error < 0 then
y := y + 1
error := error + deltax
```

Walk through the algorithm for input (1,1) to (7,4) (i.e. (x0,x1,y0,y1)=(1,7,1,4)). Listing the pixel locations that will be drawn.