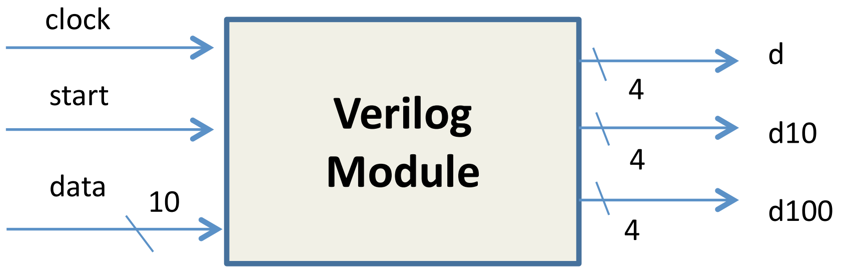
M A S S A C H U S E T T S I N S T I T U T E O F T E C H N O L O G Y
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.111 Introductory Digital Systems Laboratory

Fall 2016

Lecture PSet #8 Last one!
Due: Fri 10/21/2016 23:59
Upload solution as one pdf

Problem 1 (3 point). The double-dabble algorithm also known as *the shift and add 3* is an efficient algorithm that converts binary numbers to decimal in binary coded decimal (BCD) format by a series of shifts and addition. For example, FF becomes 255. Without using the double-dabble algorithm, implement another algorithm that converts a 10 bit binary number to a 3 digit BCD. Write a Verilog module that converts binary to BCD (and is not the double-dabble/ shift add 3 algorithm). You may assume that the maximum value for your input is 999 or 10'h3E7. *(Hint: you are trading off clock cycles for simplicity.) Verilog must have correct syntax and be able to synthesize. Divide by 10 does not synthesize!*



Start pulses high for one cycle when data is available for conversion. Data = 10'h7B (decimal 123) results in d100=1, d10=2, d=3.

Include the Verilog module, test bench and screenshot of the simulation showing conversion of 10'h7B to 123.

```
module convert_bcd(
    input clock,
    input start,
    input [9:0] data,
    output reg [3:0] d,
    output reg [3:0] d10,
    output reg [3:0] d100
    );
// your Verilog

endmodule
```

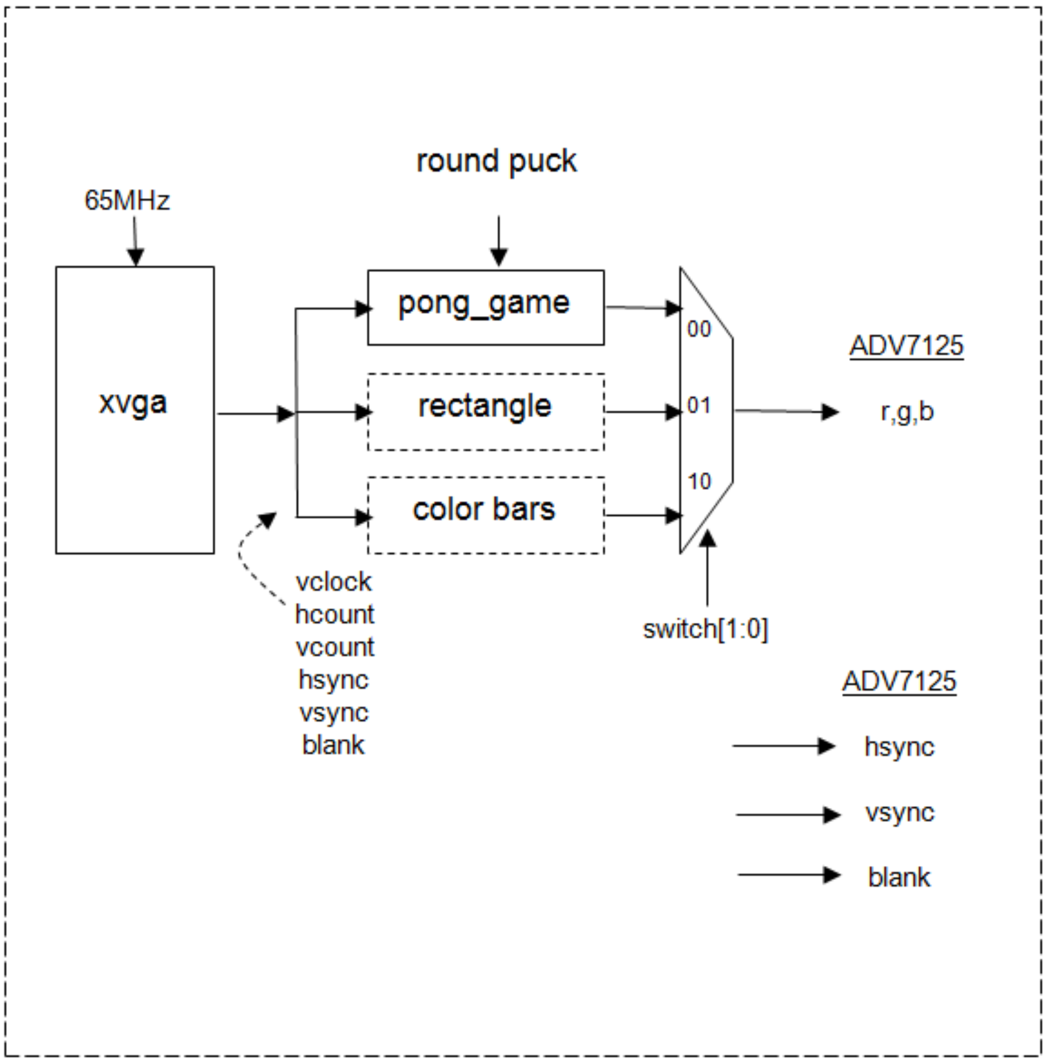
Grading	Problem 1
1	Easy to read & formatted Verilog (See "Verilog Editors" tab for help.)
2	Verilog module meeting all the specs
1	Functional test bench
1	Screenshot showing BCD value
5	**Grade**

Problem 2 (5 points total) In the pong lab, pixel, vsync, hsync, and blank are signals set to the video DAC which generates the VGA signals. The system clock is 65mhz with a 15ns period. Assume the combinatorial logic generating the rectangular blob has a 9ns t_{pd}. Generating a round ball requires additional logic using a multiply with an 8ns t_{pd}. As a result this additional block added to the other logic does not meet the timing requirements (9ns + 8ns > 15ns clock):

```
always @ *  begin   // generate round puck
     // compute x-xcenter and y-ycenter
     radiussquared = RADIUS*RADIUS;  // RADIUS is a paramater
    deltax = (hcount > (x+RADIUS)) ? (hcount-(x+RADIUS)) : ((x+RADIUS)-hcount);
     deltay = (vcount > (y+RADIUS)) ? (vcount-(y+RADIUS)) : ((y+RADIUS)-vcount);
     // check if distance is less than radius squared
     if(deltax*deltax+deltay*deltay <= radiussquared)
            pixel = COLOR;
     else pixel = 0;
   end
```

(A) Pipeline the above Verilog noting that the multiplies have the longest t_{pd}. (Logic with the longest tpd should be pipelined separately.) Use correct blocking/non-blocking assignments. Assume hcount, vcount, x and y are already registered glitch free inputs clocked by the system clock: clock_65mhz. **[2 point]**

(B) Indicate any additional register(s) required in this block diagram for correct VGA timing. The round puck Verilog is part of the pong game block. Indicate the number of registers per pipelined signal. **[2 point]** Note – you can copy and paste the image below for your solution.



C) Explain why additional register(s) are or not required. **[1 point]**

Upload solutions to Problems 1 and 2 as one pdf file.