IC150: Applied Digital Logic Design

September 15, 2021
10-11:30AM

Third Tierce Examination

Max Marks: 50

Notes:

1. Answer all questions. The exam is only conducted in the offline mode.
2. Write your name on the question paper as well as on the answer sheets.
3. Provide all intermediate steps of deriving your answers. Merely writing the final answer will not result in any marks to be awarded even if the answer is correct.
4. Mobile phones and calculators are not permitted. A person in possession of these will be treated as using unfair practices in the examination.
5. You can see your graded answer scripts on 16th Sep at 6PM.

Q1 (10 Marks): In various applications, one often needs multi-bit shifters. In this problem you are required to design a Verilog module that takes as input one 8-bit number (`in`) and a four-bit signed number (using 2's complement representation) that provides a value between -8 to +8 (`shiftcount`); and produces one 8-bit output (`out`) that is obtained by arithmetic shift of input `in` by `shiftcount`. For negative values of `shiftcount` the output is obtained by arithmetic left shifts of inputs by abs(`shiftcount`). For positive values of `shiftcount` the output is obtained by arithmetic right shift of input by count given in `shiftcount`.

```
module multishift(input [7:0] in, input [3:0] shiftcount,
                  output [7:0] out);
......
endmodule
```

Q2 (10 Marks) The following is a combinatorial shift-and-add multiplier code with some fill-in-the-blanks.

```
module combinatiorial_mult(?? 1 ?? [31:0] product,
          input [15:0] A, input [15:0] B);
   integer i;
   always @(?? 2 ?? )
     begin
        product = 0;
        for (i=0; ?? 3 ??; i=i+1)
          if(A[i] == ?? 4 ??) product = ?? 5 ??;
    end
endmodule
```

Q3 (10 Marks) You are given the following module of J-K flip flop.

```
module JK_FF (input j, input k, input clk, output q);
   reg q;

   always @ (posedge clk)
      case ({j,k})
         2'b00:  q = q;
         2'b01:  q = 0;
         2'b10:  q = 1;
         2'b11:  q = ~q;
      endcase
endmodule
```

Use this module to make a 3-bit counter by filling in the lines of code. The module has been instantiated for your convenience in this code.

```
module ThreeBitCtr(input por, input clk, output [2:0] count);
   reg [2:0] count;
   wire [2:0]J; wire [2:0]K;
   JK_FF JK1(J[0], K[0], clk, count[0]),
         JK2(J[1], K[1], count[0], count[1]),
         JK3(J[2], K[2], count[1], count[2]);
   assign J = ......;
   assign K = ......;
endmodule;
```

Q4 (10 Marks). You are given the following module.

```
module bufz (input A, input EN, output Y);
    assign Y = (EN == 1'b1)? A: 1'bz;
endmodule
```

Use this module **bufz** to implement a module **selector** whose definition without the use of module **bufz** is given below.

```
module selector (input a, input b, input sel, output y);
   assign y = (sel == 1'b1)? a: b;
endmodule
```

bufz z1(.A(a), .EN(sel), .Y(y))

bufz z2(.A(a), .EN(~sel), .Y(y))

Name: Department: ID Number: