UNIVERSITY OF BRITISH COLUMBIA
ELECTRICAL AND COMPUTER ENGINEERING
CPEN 311: Digital Systems Design

Practice Assignment 1: Basic Verilog/VHDL

This assignment will NOT be handed in, however, answers will be posted.

1. Write a Verilog/VHDL description of a 8-to-1 two bit multiplexer (so each of the 8 inputs is two bits, and the output is 2 bits). You can assume the 8 two-bit inputs are named a,b,c,d,e,f,g, and h, the select lines (3 bits) are labeled sel, and the output is labeled outm. You can either write it by hand, or type it into ModelSim and simulate.

2. Write a Verilog/VHDL description of a 3-input 8-output decoder. Recall that a 3-input 8-output decoder operates as follows: denote the 3-bit input as x The decoder asserts (sets to 1) bit x of the output, and deasserts (sets to 0) all other outputs. So, when the input is "000", the output is "00000001" (bit 0 is a 1). When the input is "100", the output is "00010000" (bit 4 is a 1). You can either write it by hand, or type it into ModelSim and simulate.

3. Write a Verilog/VHDL description of the decoder described in the following table:

En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	×	×	0	0	0	0

   This is the same as the above, except that there is an enable input. When the enable is 0, the outputs are all 0, otherwise, it operates as in Question 2. Remember, this is still a combinational circuit (in class we saw a flip-flop with an "enable" input; the meaning of that input is quite different than the meaning of "enable" in this question). You can either write it by hand, or type it into Modelsim and simulate.

4. What is wrong with the following VHDL code (it is supposed to describe a combinational demultiplexer):

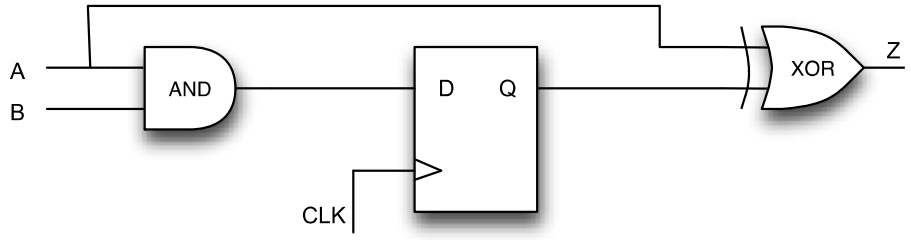
```
entity demux is
      port( a   : in std_logic;
            sel : in std_logic_vector( 2 downto 0 );
            q   : out std_logic_vector( 7 downto 0 ) );
end demux;
architecture cpen311 of demux is
begin
      process(sel,a)
      begin
            case(sel) is
                  when "000" => q(0) <= a;
                  when "001" => q(1) <= a;
                  ...
                  when others => q(7) <= a;
            end case;
      end process;
end cpen311;
```

5. We didn't talk about a JK-Flip Flop in class. A JK flip-flop has three inputs: J, K, and CLK. It has one output called Q. On each rising edge of CLK, the output of the flip-flop is assigned a value, depending on the current value of J and K, according to the following table:

J	K	Q
0	0	previous value of Q
0	1	0
1	0	1
1	1	inverse of the previous value of Q (that is, Q')

In other words, on a rising clock edge, if J and K are both 0, then Q holds its value. If J is 0 and K is 1, then Q becomes 0. If J is 1 and K is 0, then Q becomes 1. Finally, if J and K are both 1, then the value on Q is inverted. Write a Verilog/VHDL description of a JK-Flip Flop.

6. Write synthesizable Verilog/VHDL code that has the same behaviour as the following circuit. Your description should consist of one entity with two processes (no more than two). If you think you can do it in one synthesizable process, you have probably made a mistake.



7. Consider the following VHDL. (a) Describe the function (in words) of the specified circuit. (b) Draw a schematic of a circuit that has the same behaviour as the code. You can use flip-flops, basic gates, multiplexers, and/or tri-state buffers in your diagram.

```
entity mycirc is
        port ( A, B, C, CLK : in  bit;
               F            : out bit );
end mycirc;

architecture defn1 of mycirc is
  signal  W : bit;
begin
        process(CLK)
        begin
                if (CLK='1') then
                    W <= A and B;
                end if;
        end process;

        F <= W and C;

end defn1;
```

8. Draw a schematic of a circuit that has the same behaviour as the following VHDL code. You can use flip-flops, basic gates, multiplexers, and/or tri-state buffers in your diagram. *On the surface, this looks almost the same as the previous question, however, it is a bit more tricky. Be sure you understand it.*

```
entity mycirc is
        port ( A, B, D, E : in  bit;
               F          : out bit );
end mycirc;

architecture defn1 of mycirc is
  signal  W,  Y : bit;
begin
        process(E)
        begin
                if (E='1') then
                        case A is
                                when '0' => W <= B;
                                when '1' => W <= not B;
                        end case;
                        F <= A and Y;
                end if;
        end process;
        process (W,D)
        begin
                Y <= D and W;
        end process;
end defn1;
```

---

9) Write synthesizable Verilog/VHDL code that describes a 8-bit BCD (Binary Coded Decimal) counter. The counter counts from 0 to 99 (each clock cycle, the count increments by 1) and then rolls over to 0. So, the counter would count 00, 01, 02, 03, … 08, 09, 10, 11, 12 … 98, 99, 00, 01, … The counter has an eight bit output. The four least-significant bits (bits 3 to 0) encode the "one's" digit, while the four most-significant bits (bits 7 to 4) encode the "ten's" digit. So, for example, the number 54 would be encoded "0101 0100". Similarly, the number 23 would be encoded "0010 0011". Your circuit should have a *synchronous* reset signal (called RESET). Your code should consist of one entity containing as few processes as possible.