**California State University, Sacramento**
**The College of Engineering and Computer Science**

# CPE 166 Advanced Logic Design

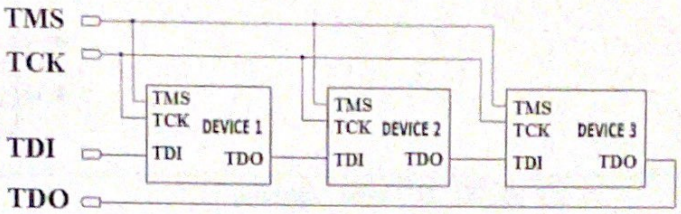
Final Exam

Fall 2020

Student Name: Xavier Howell

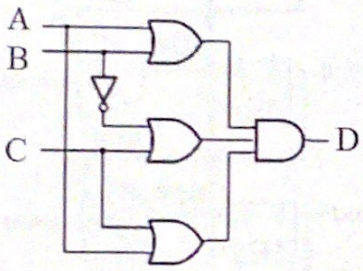
1. [21 points] Mark the correct answer or fill out blanks for the following questions. Every question has only one correct answer.

1). [ √ True] [ ___ False] The diagram below shows parallel JTAG boundary scan.




2). [ ___ True] [ √ False]
Consider the binary message 111001, and a generating polynomial $P(x)= x^3 + x + 1$. Then for error correction and detection, the CRC bits attached at the end of the above 6-bit message should be 101.

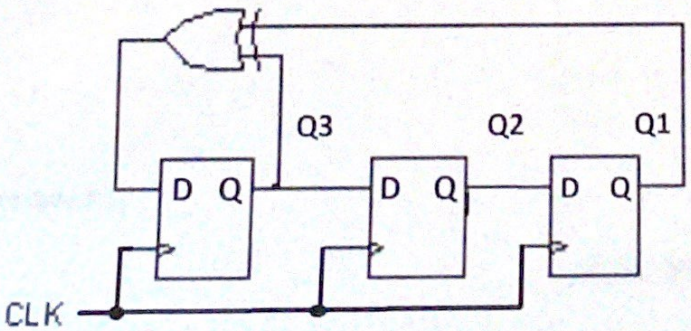
3). [ √ True] [ ___ False] The circuit below has static-0 hazard.



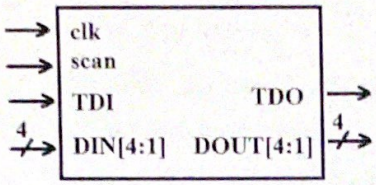

4). [ √ True] [ __ False] Reducing the clock rate can increase the mean time between failure (MTBF) value.

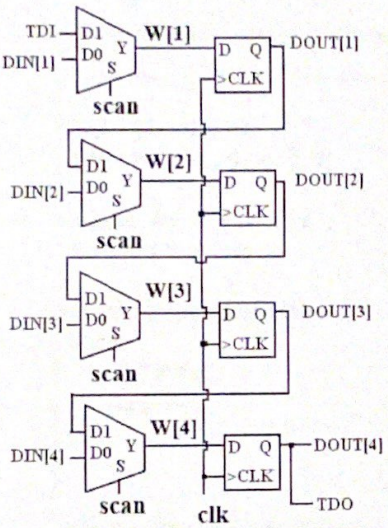
5). [ √ True] [ __ False] Xilinx CLB is used to implement the input and output functions of an FPGA device.

6). In the follower linear feedback shift register circuit, suppose the currently loaded binary values are Q3Q2Q1 = "001", after two clock cycles, the register binary values will be Q3 Q2 Q1 = 010 .



3. [ 24 points ] Use hierarchical design strategy to implement the circuit shown below in **Verilog**. When scan is true, the system should shift the testing data TDI through all scannable registers and move out through signal TDO. When scan is false, the system should work in the normal mode.




module dff (D, CLK, Q);

```
input D, CLK;
output Q;

always @ (posedge clk)
Q <= D
```

endmodule

```
module mux (D1, D0, S, Y);
input D1, D0, S;
output Y;
assign Y = S? D1: D0;
endmodule
```

endmodule

module topcir ( TDI, DIN, clk, scan, TDO, DOUT );

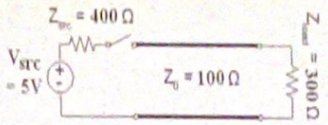
```
input TDI, clk, scan;
input [4:1] DIN;
output TDO;
output [4:1] DOUT;
wire [4:1] W;
assign TDO = DOUT[4]
mux m0 (.D1(TDI), .D0(DIN[1]), .S(scan), .Y(W[1]));
mux m1 (.D1(DOUT[1]), .D0(DIN[2]), .S(scan), .Y(W[2]));
mux m2 (.D1(DOUT[2]), .D0(DIN[3]), .S(scan), .Y(W[3]));
mux m3 (.D1(DOUT[3]), .D0(DIN[4]), .S(scan), .Y(W[4]));
```

endmodule

```
dff  d0(.D(W[0]), .CLK(clk), .Q(DOUT[0]));
dff  d1(.D(W[1]), .CLK(clk), .Q(DOUT[1]));
dff  d2(.D(W[2]), .CLK(clk), .Q(DOUT[2]));
dff  d3(.D(W[3]), .CLK(clk), .Q(DOUT[3]));

endmodule
```

7). The transmission line circuit is shown below.




Calculate reflection coefficient at the load ρload. Your answer: .5Ω

Calculate the load voltage after the first reflection. Your answer: .6Ω

Calculate the steady state load voltage. Your answer: 2.14Ω

2. [15 points ] Draw the synthesized schematics for the following Verilog codes.

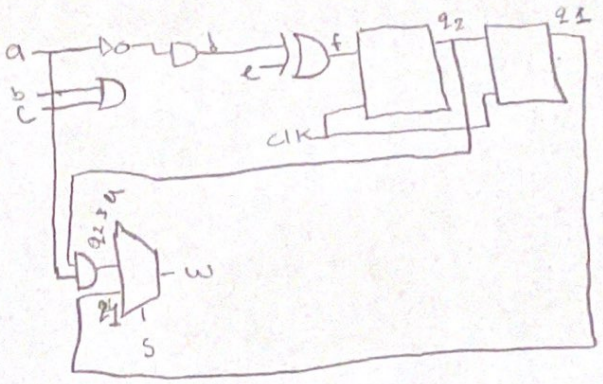
```
reg   q1,  q2,  w;
wire  a,  b, c,   d,  e,  f, s;

assign  d =  ~a &  b | c  ;
assign  f =  d ^ e;

always@(posedge clk)
begin
      q1 <= q2;
      q2  <= f;
end


always @( s or q1 or q2 or a )
begin
    if(s)
        w = q1;
    else
        w = q2  & a;
end
```

**Your Schematic**:



4. [15 points ] An SRAM has 8-bit databus and 6-bit address bus. The SRAM function table is shown below:

| WE | CS | OE | I/O | Function |
|---|---|---|---|---|
| X | L | X | High-Z | Standby |
| L | H | L | High-Z | Output Disabled |
| L | H | H | Dout | Read Data |
| H | H | X | Din | Write Data |

CS, WE, and OE signals in the above function table are high active. Design the above SRAM in **VHDL**.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity ram is
port (      address : IN      STD-LOGIC-VECTOR(6 downto 1);
            data    : INOUT   STD-LOGIC-VECTOR(8 DOWNTO 1),
            cs      : IN      STD-LOGIC                    ;
            we      : IN      STD-LOGIC                    ;
            oe      : IN      STD-LOGIC                    );
end ram;
architecture beh_ram of ram is
type memory is array (0 to 63) of std-logic-vector(7 downto 0);
signal mem: memory;
begin
      Mem_write:
          process (address, data, cs, we)
          begin
             if (cs='1' and we='0') then
              mem (conv_integer(address)) <= data;
             end if;
          end process

      Mem_read:
          process (address, cs, we, oe, mem)
          begin
             if (cs = '1' and we = '0' and oe ='1') then
                data <= mem (conv_integer (address));
             else
                data <= (others => 'z');
             end if;
          end process;
      end beh_ram;
```

5. [25 points] Implement a **VHDL** design, which writes "11100111" to all of the address locations of the SRAM you designed in question 4. Use SRAM you designed in question 4 as one component in your question 5 design.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity fsm is:
    port (clk, reset: in std_logic;
    data: inout std_logic_vector (7 downto 0);
    address: out std_logic_vector (6 downto 0)
    cs, we, oe: out std_logic);
end fsm;

architecture beh of fsm is:
    type state_type is (s0, s1, s2);
    signal state: state_type
    signal cnt: std_logic_vector (6 downto 0);
begin
    cs <= '1'
    address <= cnt;
process (clk, reset)
begin
    if (reset = '1') then
        state <= s0
        cnt <= "0000000";
    else if (clk = '1' and clk event) then
        case state is
            when s0 =>
            state <= s1;
            cnt <= "0000000";
            when s1 =>
            state <= s2;
            cnt <= "0000000";
            when s2 =>
            cnt <= cnt + 1;
            if (cnt < 64) then
            state <= s2;
    else
        state <= s3;
    else if;
    when others =>
    cnt <= "0000000"
    state <= s1;
    end case;
    end if;
end process;

process (state);
    begin
        case state is
        when s0 =>
            we <= '0'; oe <= '0';
            data <= "ZZZZZZZZ";
        when s1 =>
            we <= '1'
            data <= "11100111"
        when s2 =>
            we <= '1'
            data <= "11100111"
        when others =>
            we <= '0'; oe <= '0';
            data <= "ZZZZZZZZ";
    end case;
end process;
end beh;
```