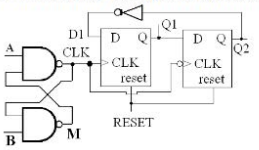
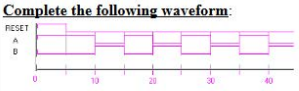
1.The following reset signal is asynchronous



Complete the following waveform:




```
Library ieee;
Use ieee.std_logic_1164.all;
entity prob1  (a, b, reset: in std_logic;
               q1, q2:  out  std_logic );
end prob1;
architecture circuit of prob1 is
signal      clk,  m, d1, t1, t2: std_logic;
begin
   clk <= a  nand m;
   m  <=  b  nand clk;
   d1 <= not t2;
process ( clk , reset)
begin
   if (reset= '1') then
      t1 <= '0';
   elsif (rising_edge(clk)) then
      t1 <= d1;
   end if;
end process;
process (  clk , reset)
begin
   if (reset= '1') then
        t2 <= '0';
   elsif (falling_edge(clk)) then
        t2 <= t1;
   end if;
end process;
q1 <= t1;
q2 <= t2;
end circuit;
```

TESTBENCH

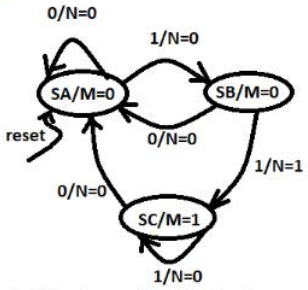
```
Library ieee;
Use ieee.std_logic_1164.all;
entity prob1_tb
end prob1_tb;
architecture beh of prob1_tb is
signal a, b, reset:   std_logic;
signal   q1, q2:    std_logic;
component prob1  (a, b, reset: in std_logic;
                  q1, q2:  out  std_logic );
end component;
begin
uut: prob1  port map(a, b, reset, q1, q2);
-- Method 2:
-- uut: prob1 port map ( a=> a, b=>b,  reset=>reset, q1=>q1, q2=>q2 );
--     port mapping rule: lower level entity  pin name => top level
testing signal name
process
begin
   reset <= '1';
   wait for 5 ns;
   reset <= '0';
   wait for 100 ns;
   wait;
end process;
b <= not a;
 process
 begin
   a <= '1';
   wait for 10 ns;
   a <= '0';
   wait for 5 ns;
   a <= '1';
   wait for 5 ns;
   a <= '0';
   wait for 5 ns;
   a <= '1';
   wait for 5 ns;
   a <= '0';
   wait for 5 ns;
   a <= '1';
   wait for 5 ns;
   a <= '0';
   wait for 5 ns;
   wait;
end process;
end beh;
```

2.Write testbench for the finite state machine and complete the waveform

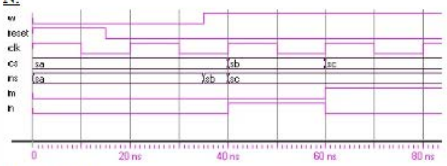
```
Library ieee;
Use ieee.std_logic_1164.all;
entity prob2_fsm ( w, Reset, Clk: in std_logic;
          M, N: out std_logic );
End prob2_fsm;
Architecture beh of prob2_fsm is
type  st is ( SA, SB, SC );
signal  cs, ns:  st;
begin
    process ( w ,  cs )
    begin
    case (cs)
          when SA=>  if (w = '0') then
                        ns <= SA ;
                     else
                        ns  <= SB ;
                     end if;
          when SB=>  if(w = '0') then
                        ns  <= SA ;
                     else
                        ns <= SC ;
                     end if;

          when SC=>  if (w = '0') then
                        ns <= SA ;
                     else
                        ns <= SC ;
                     end if;
          when others=> ns <= SA;
    end case;
    end process;
process (  Clk ,    Reset)
begin
    if( Reset = '1') then
          cs <= SA ;
    elsif (rising_edge(Clk)) then
          cs <= ns ;
    end if;
 end process;
  M <= '1' when (cs=SC)  else  '0';
  N <= w when (cs=SB)  else '0';
end beh;
```

Draw state diagram for the finite state machine.

Solution:



Complete the following waveform for signals cs, ns, M, and N.



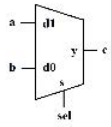

```
Library ieee;
Use ieee.std_logic_1164.all;
entity prob2_fsm_tb
end prob2_fsm_tb;
architecture beh of prob2_fsm_tb is
signal   w, Reset, Clk:  std_logic;
signal              M, N::     std_logic;
component prob2_fsm ( w, Reset, Clk: in std_logic;
                  M, N: out std_logic );
end component;
begin
uut: prob2_fsm port map ( w, Reset, Clk,  M, N);
reset <= '1', '0' after 15 ns;
w     <= '0', '1' after 35 ns;
process
begin
     Clk <= '1'; wait for 10 ns;
     Clk <= '0'; wait for 10 ns;
end process;

end beh;
```

3. Draw the synthesized schematic for the VHDL code

```
process(a, b, sel)
begin
 if (sel =  '1') then
   c <= a;
 else
   c <= b;
end if;
end process;
```

Draw the synthesized schematic for the VHDL code.

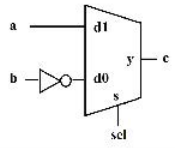
Solution:




```
process (a ,  b ,  sel)
begin
case (sel) is
when '0'=> c <= ~b;
when '1' => c <= a;
end case;
end process;
```

Draw the synthesized schematic for the VHDL code.

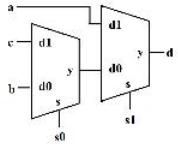
Solution:




```
process ((sl ,  s0 ,  a ,  b ,  c)
begin
if (sl ='1') then
   d <= a;
elsif (s0 = '0') then
   d <= b;
else
  d <= c;
end if;
end process;
```

Draw the synthesized schematic for the VHDL code.

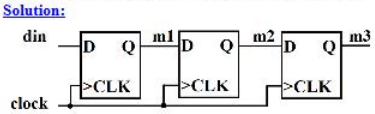
Solution:




```
process (clock )
begin
 if (rising_edge(clock)) then
  m1 <= din;
  m2 <= m1;
  m3 <= m2;
end if;
end process;
```

Draw the synthesized schematic for the VHDL code.

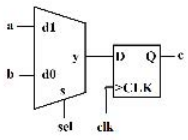
Solution:



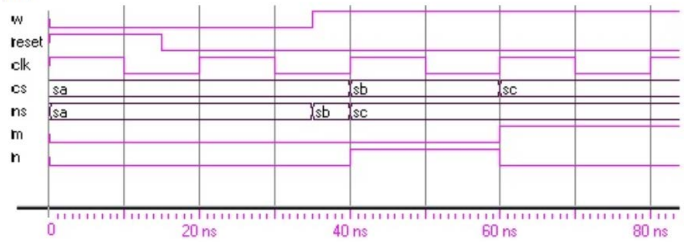

```
process (clock )
begin
  if (rising_edge(clock)) then
    if(sel = '1') then
       c <= a;
    else
       c <= b;
    end if;
end if;
end process;
```

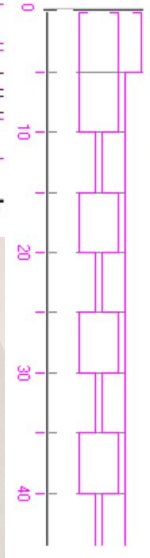
Draw the synthesized schematic for the VHDL code.

Solution:

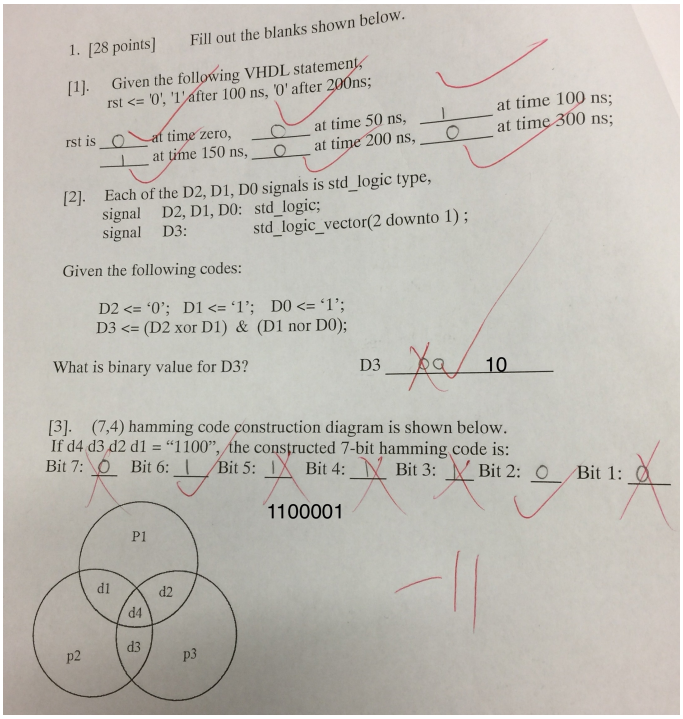


Complete the following waveform for signals cs, ns, M, and N.





Complete the following waveform:

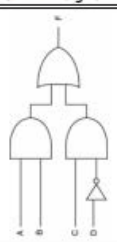


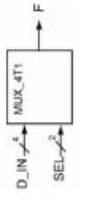

```
end  ocess
```

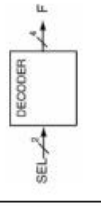
INPUTS		OUTPUTS						
A	B	AND	NAND	OR	NOR	EXOR	EXNOR	
0	0	0	1	0	1	0	1	
0	1	0	1	1	0	1	0	
1	0	0	1	1	0	1	0	
1	1	1	0	1	0	0	1	

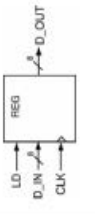
NOT gate	
A	A
0	1
1	0

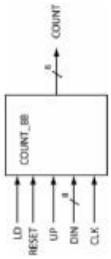
Description	CKT Diagram	VHDL Model

Typical logic circuit		`entity my_ckt is`
		`  Port ( A,B,C,D : in std_logic;`
		`  F : out std_logic);`
		`end my_ckt;`
		
		`architecture ckt1 of my_ckt is`
		`begin`
		`  F <= (A AND B) OR (C AND (NOT D));`
		`end ckt1;`
		
		`architecture ckt2 of my_ckt is`
		`begin`
		`  F <= '1' when (A = '1' AND B = '1') else`
		`  '1' when (C = '1' AND D = '0') else`
		`  '0';`
		`end ckt2;`

4:1 Multiplexor		`entity MUX_4T1 is`
		`  Port ( SEL : in std_logic_vector(1 downto 0);`
		`  D_IN : in std_logic_vector(3 downto 0);`
		`  F : out std_logic);`
		`end MUX_4T1;`
		
		`architecture my_mux of MUX_4T1 is`
		`begin`
		`  F <= D_IN(0) when (SEL = "00") else`
		`  D_IN(1) when (SEL = "01") else`
		`  D_IN(2) when (SEL = "10") else`
		`  D_IN(3) when (SEL = "11") else`
		`  '0';`
		`end my_mux;`

2:4 Decoder		`entity DECODER is`
		`  Port ( SEL : in std_logic_vector(1 downto 0);`
		`  F : out std_logic_vector(3 downto 0));`
		`end DECODER;`
		
		`architecture my_dec of DECODER is`
		`begin`
		`  with SEL select`
		`  F <= "0001" when "00",`
		`  "0010" when "01",`
		`  "0100" when "10",`
		`  "1000" when "11",`
		`  "0000" when others;`
		`end my_dec;`

8-bit register with chip load enable		`entity REG is`
		`  port ( LD,CLK : in std_logic;`
		`  D_IN : in std_logic_vector (7 downto 0);`
		`  D_OUT : out std_logic_vector (7 downto 0));`
		`end REG;`
		
		`architecture my_reg of REG is`
		`begin`
		`  process (CLK,LD)`
		`  begin`
		`  if (LD = '1' and rising_edge(CLK)) then`
		`  D_OUT <= D_IN;`
		`  end if;`
		`  end process;`
		`end my_reg;`

8-bit up/down counter with asynchronous reset		`entity COUNT_8B is`
		`  port ( RESET,CLK,LD,UP : in std_logic;`
		`  DIN : in std_logic_vector (7 downto 0);`
		`  COUNT : out std_logic_vector (7 downto 0));`
		`end COUNT_8B;`
		
		`architecture my_count of COUNT_8B is`
		`  signal t_cnt : std_logic_vector(7 downto 0);`
		`begin`
		`  process (CLK, RESET)`
		`  begin`
		`  if (RESET = '1') then`
		`  t_cnt <= "00000000";`
		`  elsif (rising_edge(CLK)) then`
		`  if (LD = '1') then t_cnt <= DIN;`
		`  else`
		`  if (UP = '1') then t_cnt <= t_cnt + 1;`
		`  else t_cnt <= t_cnt - 1;`
		`  end if;`
		`  end if;`
		`  end if;`
		`  end process;`
		
		`  COUNT <= t_cnt;`
		`end my_count;`

	Data bits				parity		
	D3	D2	D1	D0	P3	P2	P1
0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	1
2	0	0	1	0	1	0	1
3	0	0	1	1	1	1	0
4	0	1	0	0	1	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	1	0	0	0
8	1	0	0	0	1	1	1
9	1	0	0	1	1	0	0
A	1	0	1	0	0	1	0
B	1	0	1	1	0	0	1
C	1	1	0	0	0	0	1
D	1	1	0	1	0	1	0
E	1	1	1	0	1	0	0
F	1	1	1	1	1	1	1

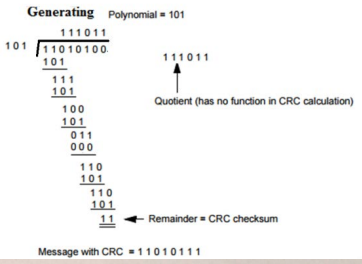
Table 1

Data bits and Hamming code.

5. CRC Error Detection

Message 110101

Generating Polynomial = 101

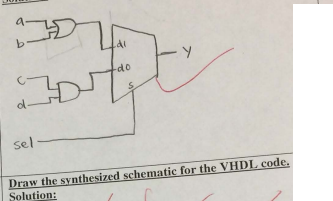


Message with CRC = 11010111

[16 points] Draw the synthesized schematic for the VHDL code.

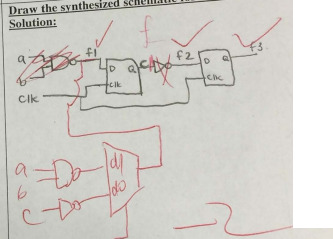
```
signal a, b, c, d, sel: std_logic;
...
...

process (a , b , c, d, sel)
begin
case (sel) is
when '0' => y <= a or b;
when '1' => y <= c and d;
end case;
end process;
```

Draw the synthesized schematic for the VHDL code.
Solution:



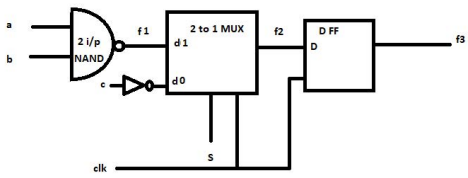

```
signal  a, b, c:  std_logic;
signal  f1, f2, f3;
...
...

f1  <= a  nand  b;

process (clock )
begin
  if (rising_edge(clock)) then
    if( s = '1') then
        f2  <= f1;
    else
        f2 <=  not  c;
    end if;
    f3 <= f2;
  end if;
end process;
```

Draw the synthesized schematic for the VHDL code.
Solution:



a, b & c are 3 input signals. clk is also an input signal. a & b is given to 2 i/p nand gate and its output is f1. f1 & not c is given as input to 2 to 1 MUX. When clk is at rising edge and select line is 1 the output of MUX f2 is f1 else if S =0 then f2 is not of c (c'). Then f2 is given as input to D flip flop and at rising edge of clock it's output f3 is f2.

The RTL Schematic of VHDL code



Count Value	Output Frequency
1	25MHz
25	1MHz
50	500KHz
1000	25KHz
25000000	1Hz

VHDL Code for Clock Divider

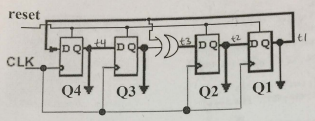
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.numeric_std.ALL;
4
5  entity Clock_Divider is
6  port ( clk,reset: in std_logic;
7  clock_out: out std_logic);
8  end Clock_Divider;
9
10 architecture bhv of Clock_Divider is
11
12 signal count: integer:=1;
13 signal tmp : std_logic := '0';
14
15 begin
16
17 process(clk,reset)
18 begin
19 if(reset='1') then
20 count<=1;
21 tmp<='0';
22 elsif(clk'event and clk='1') then
23 count <=count+1;
24 if (count = 25000) then
25 tmp <= NOT tmp;
26 count <= 1;
27 end if;
28 end if;
29 clock_out <= tmp;
30 end process;
31
32 end bhv;
```

VHDL Testbench code for Clock Divider

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY Tb_clock_divider IS
5  END Tb_clock_divider;
6
7  ARCHITECTURE behavior OF Tb_clock_divider IS
8
9  -- Component Declaration for the Unit Under Test (UUT)
10
11 COMPONENT Clock_Divider
12 PORT(
13 clk : IN std_logic;
14 reset : IN std_logic;
15 clock_out : OUT std_logic
16 );
17 END COMPONENT;
18
19 --Inputs
20 signal clk : std_logic := '0';
21 signal reset : std_logic := '0';
22
23 --Outputs
24 signal clock_out : std_logic;
25
26 -- Clock period definitions
27 constant clk_period : time := 20 ns;
28
29 BEGIN
30
31 -- Instantiate the Unit Under Test (UUT)
32 uut: Clock_Divider PORT MAP (
33 clk => clk,
34 reset => reset,
35 clock_out => clock_out
36 );
37
38 -- Clock process definitions
39 clk_process :process
40 begin
41 clk <= '0';
42 wait for clk_period/2;
43 clk <= '1';
44 wait for clk_period/2;
45 end process;
46
47 -- Stimulus process
48 stim_proc: process
49 begin
50 wait for 100 ns;
51 reset <= '1';
52 wait for 100 ns;
53 reset <= '0';
54 wait;
55 end process;
56
57 END;
```

2. [32 points]

Design the following circuit in VHDL. This circuit is required to be loaded with value Q4 Q3 Q2 Q1 = "1100" if the asynchronous reset control signal is logic '1'.




```
library ieee;
use ieee.std_logic_1164.all;
entity flop is port(
clk, D, reset : in std_logic;
Q4, Q3, Q2, Q1 : out std_logic;
end flop;
Architecture archi of flop is
signal clk, t1, D2, Q3, t2, t3, t4 : std_logic
begin
t1 <= xor Q3;
D2 <= t1;
process (reset, clk)
begin
if (reset = '1') then
Q4 <= '1';
Q3 <= '1';
Q2 <= '0';
Q1 <= '0';
elseif (rising_edge(clk) then
Q4 <= t4;
Q3 <= t3;
Q2 <= t2;
Q1 <= t1;
endif;
end process;
end archi;
```

t2 <= ?
t3 <= ?
t4 <= ?

```
//Answer Question 2
    Q: out std_logic_vector(4downto1);
end flop;

architecture whatever of flop is //MAKE SURE YOU CHANGE THE NAME OF whatever TO SOMETHING DIFFERENT
signal t: std_logic_vector(4downto1);
process (clk,reset)
begin
    if (reset='1') then
    t<="1100";
    elseif (rising_edge(clk))then
    t(1) <= t(2);
    t(2) <= t(3) xor t(1);
    t(3) <= t(4) ;
    t(4) <= t(1) ;
    endif;
    endif;
end process;
Q<=T
end whatever; //MAKE SURE YOU CHANGE THE NAME OF whatever TO SOMETHING DIFFERENT]

//Testbench, flop_tb can be renamed

library ieee;
use ieee.std_logic_1164.all
entity flop_tb
entity flop_tb
architecture beh of flop_tb is
component flop (
    clk, reset: in std_logic;
    Q: out_std_logic_vector(4downto1);
end component;
signal clk, D, reset: std_logic;
signal Q: std_logic_vector(4downto1);
constant clk2: time := 100 ns;

begin
uut: flop port map (clk, reset, Q)


clk_process :process
  begin
  clk <= '0';
  wait for clk2/2;
  clk <= '1';
  wait for clk2/2;
  end process;

 process
 begin
 reset <='1';
 wait 5ns;
 reset <='0'
 wait 5ns;
 wait;
 end process;
 end beh;
```