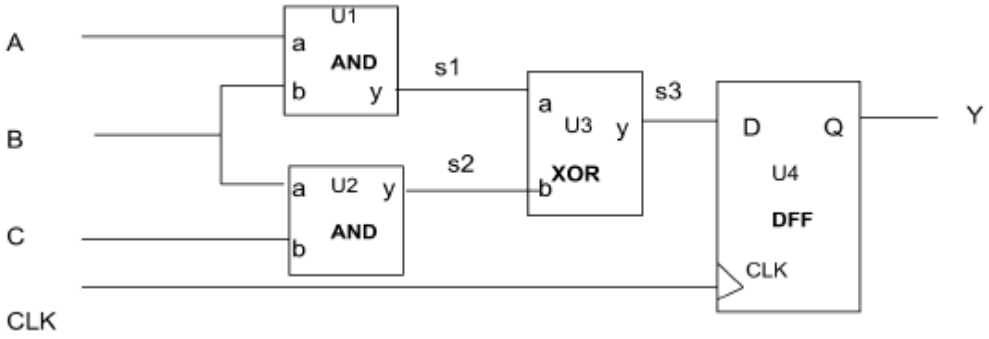
EE4620/EE6620 Name: (last)_____first)_____
Exam#1 (100 min) Summer 2024

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Q1 | 30 pts | | Q3 | 20 pts | | Q5 | 30 pts | | Total | 170 pts | |
| Q2 | 20 pts | | Q4 | 30 pts | | Q6 | 40 pts (incl. Bonus 20 pts) | | | | |

1. [30 pts] Complete the entity and architecture pair of VHDL structural description for a structural representation of a "top_level" circuit shown below. You need to declare the components (i.e., **AND2** for 2-input AND gate, **XOR2** for 2-input XOR gate, and **DFF1** for D flip-flop) used in the schematic diagram. Assume these basic components have been compiled. Ensure all signals and ports of **STD_LOGIC** type and labeled in the schematic diagram match your VHDL code. Use **BY-NAME** method of port mapping.




```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
ENTITY top_level IS
        PORT (A, B, C, clk : in std_logic;
                Y : out std_logic);

END ENTITY top_level;
        architecture structure of top_level is
                -- Component and signal declarations

                component AND2
                port (a, b: in std_logic;
                        y: out std_logic);
                end component;
                component XOR2
                        port (a,b: in std_logic;
                            y: out std_logic);
                end component;
                component DFF1
                        port (d, clk: in std_logic;
                            q: out std_logic);
                end component;

                signal s1, s2, s3: std_logic

        begin
        -- Component instantiations statements

        u1: AND2
                port map (a=>A, b=>B, y=>s1);
        u2: AND2
                port map (a=>B, b=>C, y=>s2);
        u3: XOR2
                port map (a=>s1, b=>s2, y=>s3);
        u4: DFF1
                port map (d=>s3, clk=>clk, q=>Y);

        end structure;
```

2. [20 pts] The VHDL compiler has found multiple errors with the following code. **Identify at least 10 errors**. Identifying non errors will result in a loss of 2 points. **Write your corrected line # and VHDL code on the right column.**

```
1.  library IEEE;
2.  use IEEE.STD_LOGIC_1164.all;
3.  entity decoder1
4.       port(a : in BIT(1 downto 0);
5.            EN: in BIT;
6.            b : out BIT(3 downto 0));
7.  end decoder1;
8.
9.  architecture bhv of decoder1
10. begin
11. process(a  EN)
12. begin
13.      if (EN = 0) then
14.           if (a="00") then
15.                b = "0001";
16.           else (a="01") then
17.                b = "0010";
18.           else (a="10") then
19.                b = "0100";
20.           else
21.                b = "1000";
22.           end if;
23.     else
24.           if (a="00") then
25.                b = "1011";
26.           else (a="01") then
27.                b = "1111";
28.           else (a="10") then
29.                b = "0100";
30.           else
31.                b = "1100";
32.           end if;
33.     end if;
34. end;
35. end bhv;
```

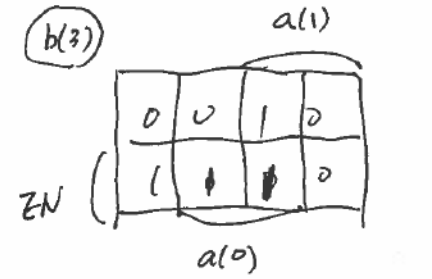
```
1.  library IEEE;
2.  use IEEE.STD_LOGIC_1164.all;
3.  entity decoder1 is
4.       port(a : in BIT_VECTOR (1 downto 0);
5.            EN: in BIT;
6.            b : out BIT_VECTOR (1 downto 0));
7.  end decoder1;
8.
9.  architecture bhv of decoder1 is
10. begin
11. process(a, EN)
12. begin
13.      if (EN = '0') then
14.           if (a="00") then
15.                b <= "0001";
16.           elsif (a="01") then
17.                b <= "0010";
18.           elsif (a="10") then
19.                b <= "0100";
20.           else
21.                b <= "1000";
22.           end if;
23.     else
24.           if (a="00") then
25.                b <= "1011";
26.           elsif (a="01") then
27.                b <= "1111";
28.           elsif (a="10") then
29.                b <= "0100";
30.           else
31.                b <= "1100";
32.           end if;
33.     end if;
34. end process;
35. end bhv;
```

3. [20 pts] Continue Prob. 2

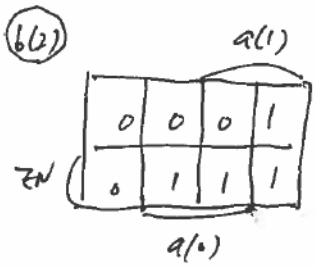
a) [10 pts] After fixing errors of the entity decoder, construct the truth table in the following format.

| **Input** | | | **Output** | | | |
|---|---|---|---|---|---|---|
| EN | a(1) | a(0) | b(3) | b(2) | b(1) | b(0) |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

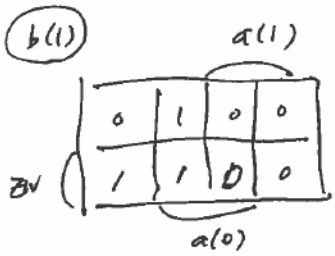
b) [10 pts] *Use K-maps to derive the optimal Boolean expression* for b(3), b(2), b(1), and b(0).



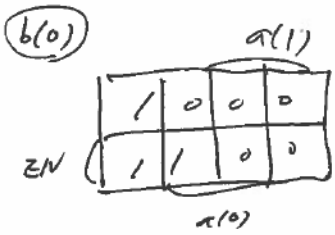

$b(3) = EN \cdot a(1)' + a(1) \cdot a(0)$




$b(2) = EN \cdot a(0) + a(1)a(0)'$




$b(1) = EN \cdot a(1)' + a(1)'a(0)$




$b(0) = EN \cdot a(1)' + a(1)' \cdot a(0)'$

4. [30 pts] Complete VHDL that implements two functions: F(A,B,C) = B + A'C' and G(A,B,C) = AB' + C

a) [10 pts] Use ***concurrent VHDL code*** (*use the exact Boolean expression* F(A,B,C) = B + A'C' and G(A,B,C) = AB' + C

```
entity function is
        port ( A,B,C : in std_logic;
                F,G : out std_logic);
end function;

architecture concurrent of function is
begin
            F <= B OR (NOT A AND NOT C);

            G <= (A AND NOT B) OR C;
end concurrent;
```

b) [10 pts] Use ***if*** statement (*use the exact Boolean expression* F(A,B,C) = B + A'C' and G(A,B,C) = AB' + C)

```
entity function is
        port ( A,B,C : in std_logic;
                F,G : out std_logic);
end function;

architecture behavior_1 of function is
begin
    proc1: process(A,B,C)

    begin

        if (B = '1') then
                F <= '1';
        elsif (A = '0' and C = '0') then
                F <= '1';
        else
                F <= '0';
        end if;

        if (A = '1' and B = '0') then
                G <= '1';
        elsif (C = '1') then
                G <= '1';
        else
                G <= '0';
        end if;
  end process proc1;
end behavior_1;
```

c) [10 pts] Complete VHDL that implements two functions: F(A,B,C) = B + A'C' and G(A,B,C) = AB' + C

Use ***case*** statement. Note: *Use two case statements; one for F and the other for G.*

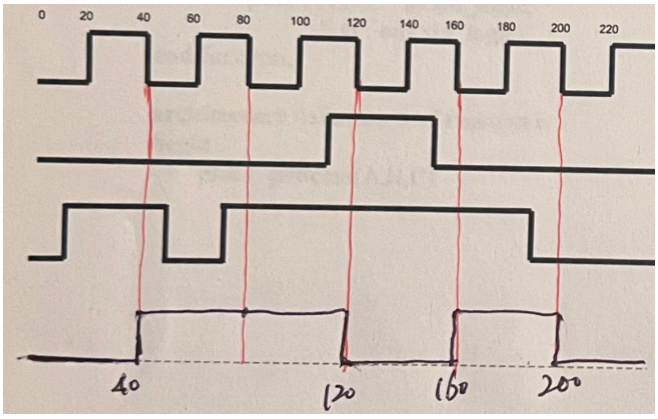
```
entity function is
        port ( A,B,C : in std_logic;
               F,G : out std_logic);
end function;

architecture behavior_2 of function is

signal ABC: std_logic_vector(2 downto 0);
 begin
      ABC <= (A,B,C);
        my_proc: process (ABC)

        begin

                case (ABC) is
                when "000" => F <= '1';
                when "010" => F <= '1';
                when "011" => F <= '1';
                when "110" => F <= '1';
                when "111" => F <= '1';
                when others => F <= '0';
                end case;

                case (ABC) is
                when "001" => G <= '1';
                when "011" => G <= '1';
                when "100" => G <= '1';
                when "101" => G <= '1';
                when "111" => G <= '1';
                when others => G <= '0';
                end case;
        end process my_proc;
end behavior_2;
```

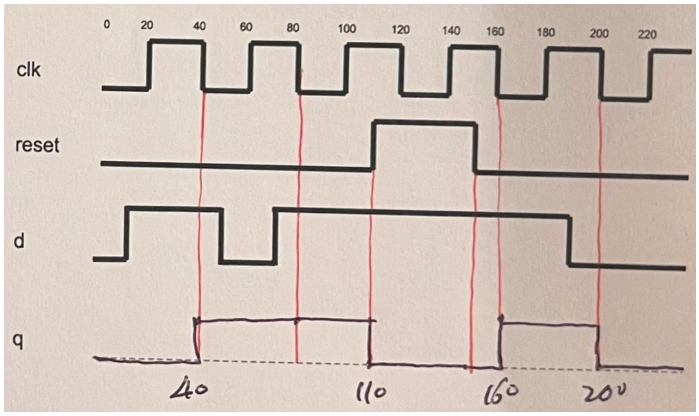
5. [30 pts] Complete the waveform for the output **q** for the following different architectures of D flip-flop.

```
ENTITY dff_reset IS
   PORT (reset, clk: IN BIT;
         d: IN STD_LOGIC;
         q: OUT STD_LOGIC);
END ENTITY dff_reset;
```

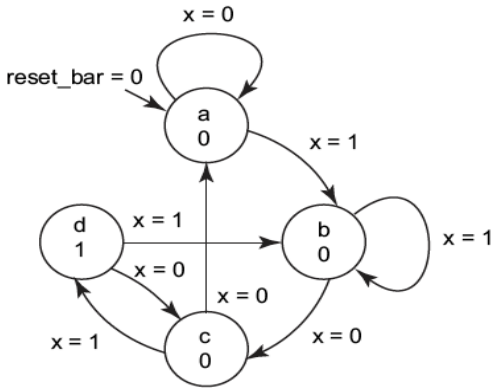
a) [15 pts]
```
ARCHITECTURE a_behavior OF dff_reset IS
 BEGIN
     PROCESS (clk)
     BEGIN
           IF (clk'EVENT AND clk = '0') THEN
            IF (reset = '1') THEN
                    q <=  '0';
            ELSE
                    q <= d;
            END IF;
           END IF;
    END PROCESS;
END ARCHITECTURE a_behavior;
```

b) [15 pts]
```
ARCHITECTURE b_behavior OF dff_reset IS
 BEGIN
   PROCESS (clk, reset)
   BEGIN
        IF (reset = '1') THEN
          q <=  '0';
        ELSIF (clk'EVENT AND clk = '0') THEN
          q <= d;
        END IF;
   END PROCESS;
END ARCHITECTURE b_behavior;
```



6. [40 pts] Below is a state diagram for a Moore FSM implementation. The Output of the State machine depends only on present state. *The FSM uses a synchronous* **negative** reset input, *reset_bar and* ***clock has a higher priority than negative reset***.

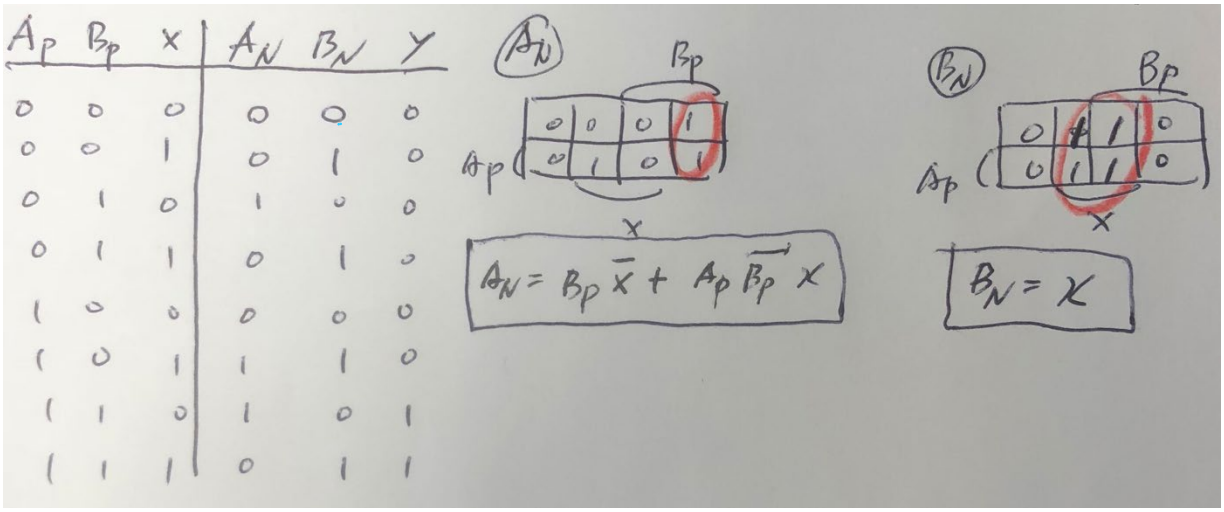


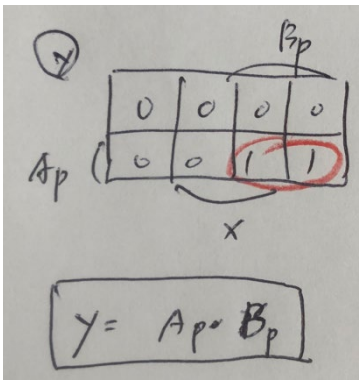

a) [20 pts] Complete the architecture of VHDL **behavioral** description below.

```
library ieee;
use ieee.std_logic_1164.all;
entity seq_detect is
        port (clk, reset_bar, x : in std_logic;
                y : out std_logic);
end;

architecture enum of seq_detect is
type state is (state_a, state_b, state_c, state_d);

signal present_state, next_state : state;

begin

nx_state: process (present_state, x) -- next state process
                begin
                -- use CASE and then IF statement
                case present_state is
                        when state_a =>
                                if x = '1' then
                                        next_state <= state_b;
                                else
                                        next_state <= state_a;
                                end if;
                        when state_b =>
                                if x = '0' then
                                        next_state <= state_c;
                                else
                                        next_state <= state_b;
                                end if;
                        when state_c =>
                                if x = '1' then
                                        next_state <= state_d;
                                else
                                        next_state <= state_a;
                                end if;
```

```
                                    when state_d =>
                                            if x = '0' then
                                                    next_state <= state_c;
                                            else
                                                    next_state <= state_b;
                                            end if;
                            end case;
                    end process nx_state;

            state_reg: process (clk, reset_bar)       -- state register process
                            begin
            -- Synchronous reset; clock has a higher priority than negative reset
                    if (clk'EVENT AND clk = '1') then
                            if (reset_bar = '0') then
                                    present_state <= state_a;
                            else
                                    present_state <= next_state;
                            end if;
                    end if;
              end process state_reg;

            -- Concurrent VHDL for output assignment

            y <= '1' WHEN present_state = state_d ELSE '0';

end enum;
```

b) [Bonus 20 pts] Construct state table for the Moore machine. There are 4 states, a, b, c and d. You need 2 flip-flops, A and B for implementation. Use state assignment (A B) = (0 0) for state **a**, (0 1) for state **b**, (1 0) for state **c**, and (1 1) for state **d**. Note: In your state table, use $A_P$ for the present state A, $B_P$ for the present state B, $A_N$ for the next state A, $B_N$ for the next state B, x for the input, and y for the output. Derive Boolean equation for $A_N$, $B_N$, and y.




Bp
0 0 0 0
Ap 0 0 1 1
x
Y= Ap· Bp