EXPERIMENT 5 PRELIMINARY WORK

Q1: a) 1. Analytical Solution

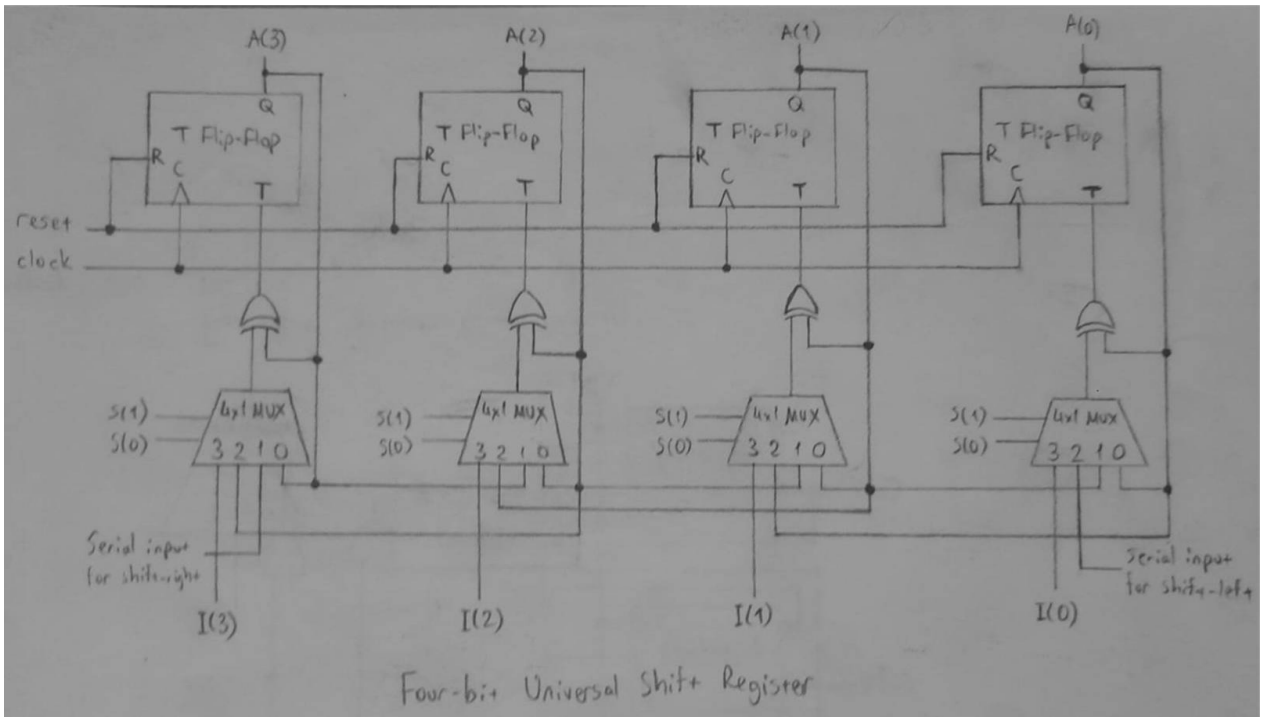
Refer to the figure in the textbook. To convert a D Flip-Flop to T Flip-Flop, equate the next state expressions. The reset input is now active high; i.e., the bubbles are absent compared to the figure in the textbook.

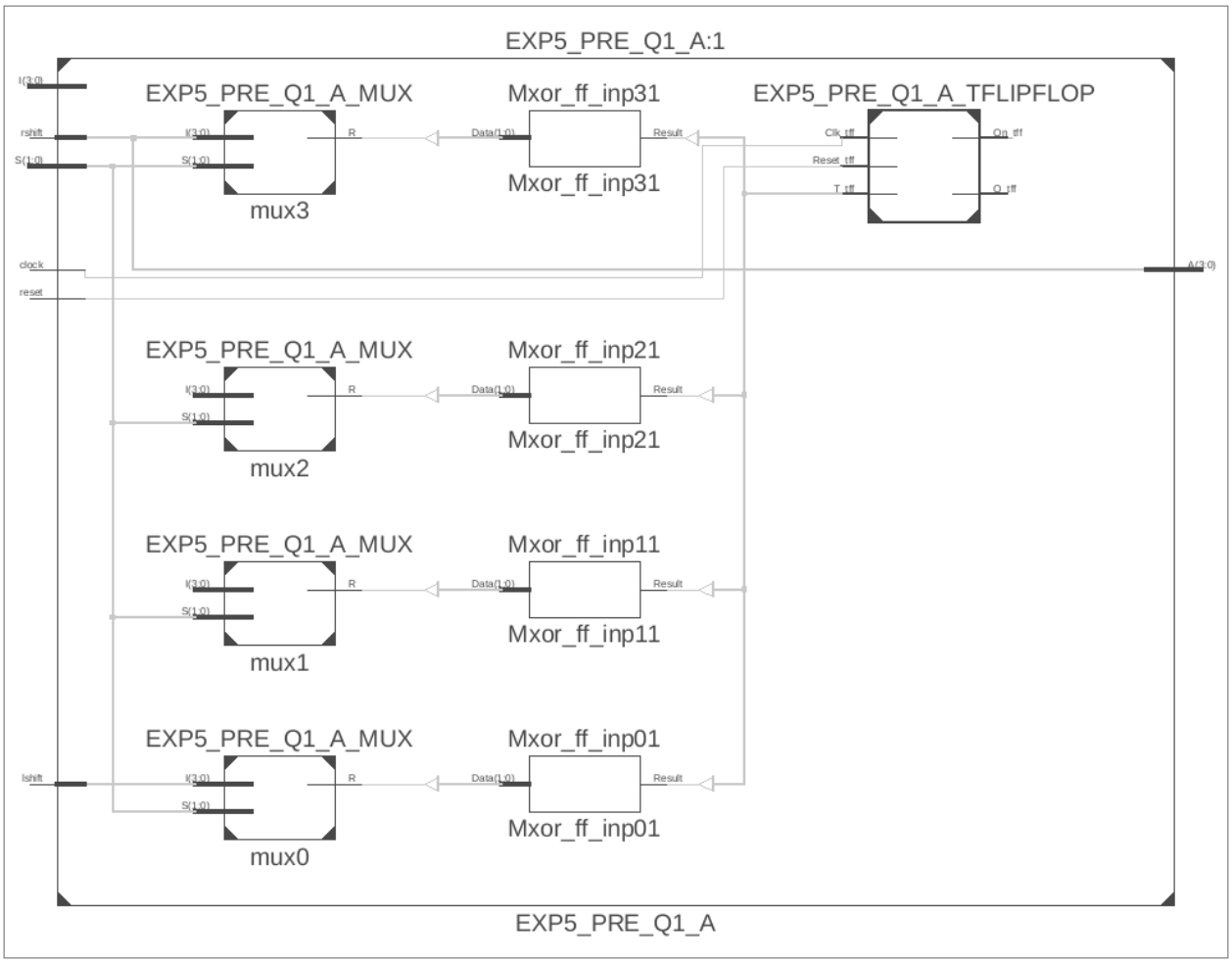
$$\mathrm{Q_T}(t+1) = \mathrm{TQ'} + \mathrm{T'Q}, \qquad \mathrm{Q_D}(t+1) = \mathrm{D} \implies \mathrm{D} = \mathrm{T} \oplus \mathrm{Q}$$

Solve for T.

$$\mathrm{D} = \mathrm{T} \oplus \mathrm{Q} \implies \mathrm{D} \oplus \mathrm{Q} = \mathrm{T} \oplus \underbrace{(\mathrm{Q} \oplus \mathrm{Q})}_{0} \implies \mathrm{T} = \mathrm{D} \oplus \mathrm{Q}$$

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - MUX

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q1_A_MUX is
    port ( I : in  STD_LOGIC_VECTOR (3 downto 0);
           S : in  STD_LOGIC_VECTOR (1 downto 0);
           R : out  STD_LOGIC);
end EXP5_PRE_Q1_A_MUX;

architecture Behavioral of EXP5_PRE_Q1_A_MUX is
begin
    R <= (not S(1) and not S(0) and I(0)) or
         (not S(1) and S(0) and I(1)) or
         (S(1) and not S(0) and I(2)) or
         (S(1) and S(0) and I(3));
end Behavioral;
```

VHDL - T Flip-Flop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q1_A_TFLIPFLOP is
    port ( T_tff, Reset_tff, Clk_tff : in std_logic;
           Q_tff, Qn_tff : out std_logic);
end EXP5_PRE_Q1_A_TFLIPFLOP;
architecture Behavioral of EXP5_PRE_Q1_A_TFLIPFLOP is
signal tempQ : std_logic;
begin
    process (Reset_tff, Clk_tff)
    begin
        if Clk_tff'event and Clk_tff = '1' then
            if Reset_tff = '1' then tempQ <= '0';
            elsif T_tff = '1' then tempQ <= not tempQ;
            end if;
        end if;
    end process;
    Q_tff <= tempQ; Qn_tff <= not tempQ;
end Behavioral;
```

VHDL - Four-Bit Universal Shift Register

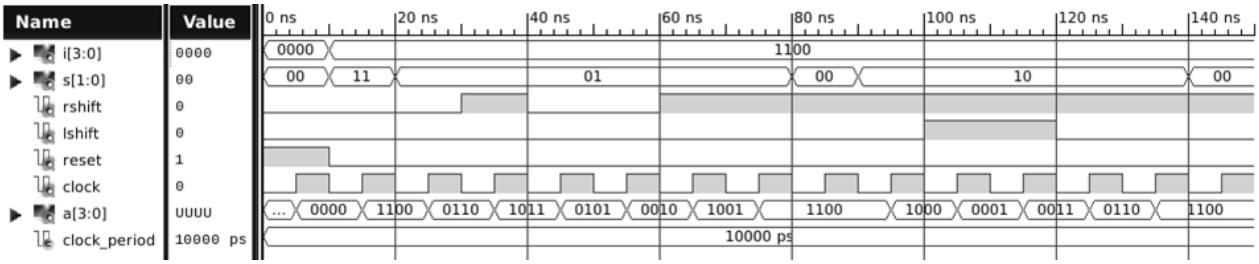
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q1_A is
    port ( I : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (1 downto 0);
           rshift, lshift, reset, clock : in STD_LOGIC;
           A : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q1_A;
architecture Behavioral of EXP5_PRE_Q1_A is
component EXP5_PRE_Q1_A_MUX
    port ( I : in  STD_LOGIC_VECTOR (3 downto 0);
           S : in  STD_LOGIC_VECTOR (1 downto 0);
           R : out  STD_LOGIC);
end component;
component EXP5_PRE_Q1_A_TFLIPFLOP
    port ( T_tff, Reset_tff, Clk_tff : in std_logic;
           Q_tff, Qn_tff : out std_logic);
end component;
signal mux_inp3, mux_inp2 : STD_LOGIC_VECTOR (3 downto 0);
signal mux_inp1, mux_inp0 : STD_LOGIC_VECTOR (3 downto 0);
signal mux_outp3, mux_outp2, mux_outp1, mux_outp0 : STD_LOGIC;
signal ff_inp3, ff_inp2, ff_inp1, ff_inp0 : STD_LOGIC;
signal ff_outp3, ff_outp2, ff_outp1, ff_outp0 : STD_LOGIC;
```

```
signal ff_outpn3, ff_outpn2, ff_outpn1, ff_outpn0 : STD_LOGIC;
begin
    mux_inp3 <= I(3) & ff_outp2 & rshift & ff_outp3;
    mux_inp2 <= I(2) & ff_outp1 & ff_outp3 & ff_outp2;
    mux_inp1 <= I(1) & ff_outp0 & ff_outp2 & ff_outp1;
    mux_inp0 <= I(0) & lshift & ff_outp1 & ff_outp0;
    mux3 : EXP5_PRE_Q1_A_MUX port map (mux_inp3, S, mux_outp3);
    mux2 : EXP5_PRE_Q1_A_MUX port map (mux_inp2, S, mux_outp2);
    mux1 : EXP5_PRE_Q1_A_MUX port map (mux_inp1, S, mux_outp1);
    mux0 : EXP5_PRE_Q1_A_MUX port map (mux_inp0, S, mux_outp0);
    ff_inp3 <= mux_outp3 xor ff_outp3; ff_inp2 <= mux_outp2 xor ff_outp2;
    ff_inp1 <= mux_outp1 xor ff_outp1; ff_inp0 <= mux_outp0 xor ff_outp0;
    ff3 : EXP5_PRE_Q1_A_TFLIPFLOP
          port map (ff_inp3, reset, clock, ff_outp3, ff_outpn3);
    ff2 : EXP5_PRE_Q1_A_TFLIPFLOP
          port map (ff_inp2, reset, clock, ff_outp2, ff_outpn2);
    ff1 : EXP5_PRE_Q1_A_TFLIPFLOP
          port map (ff_inp1, reset, clock, ff_outp1, ff_outpn1);
    ff0 : EXP5_PRE_Q1_A_TFLIPFLOP
          port map (ff_inp0, reset, clock, ff_outp0, ff_outpn0);
    A <= ff_outp3 & ff_outp2 & ff_outp1 & ff_outp0;
end Behavioral;
```

3. Results

Test bench

```
stim_proc: process     -- S=00: no change,  S=01: shift right,
begin                  -- S=10: shift left, S=11: parallel load
    reset <= '1'; wait for clock_period;
    I <= "1100"; S <= "11"; reset <= '0'; wait for clock_period;
    S <= "01"; rshift <= '0'; wait for clock_period;
    rshift <= '1'; wait for clock_period;
    rshift <= '0'; wait for clock_period;
    rshift <= '0'; wait for clock_period;
    rshift <= '1'; wait for clock_period;
    rshift <= '1'; wait for clock_period;
    S <= "00"; wait for clock_period;
    S <= "10"; lshift <= '0'; wait for clock_period;
    lshift <= '1'; wait for clock_period;
    lshift <= '1'; wait for clock_period;
    lshift <= '0'; wait for clock_period;
    lshift <= '0'; wait for clock_period;
    S <= "00"; wait for clock_period;
end process;
```

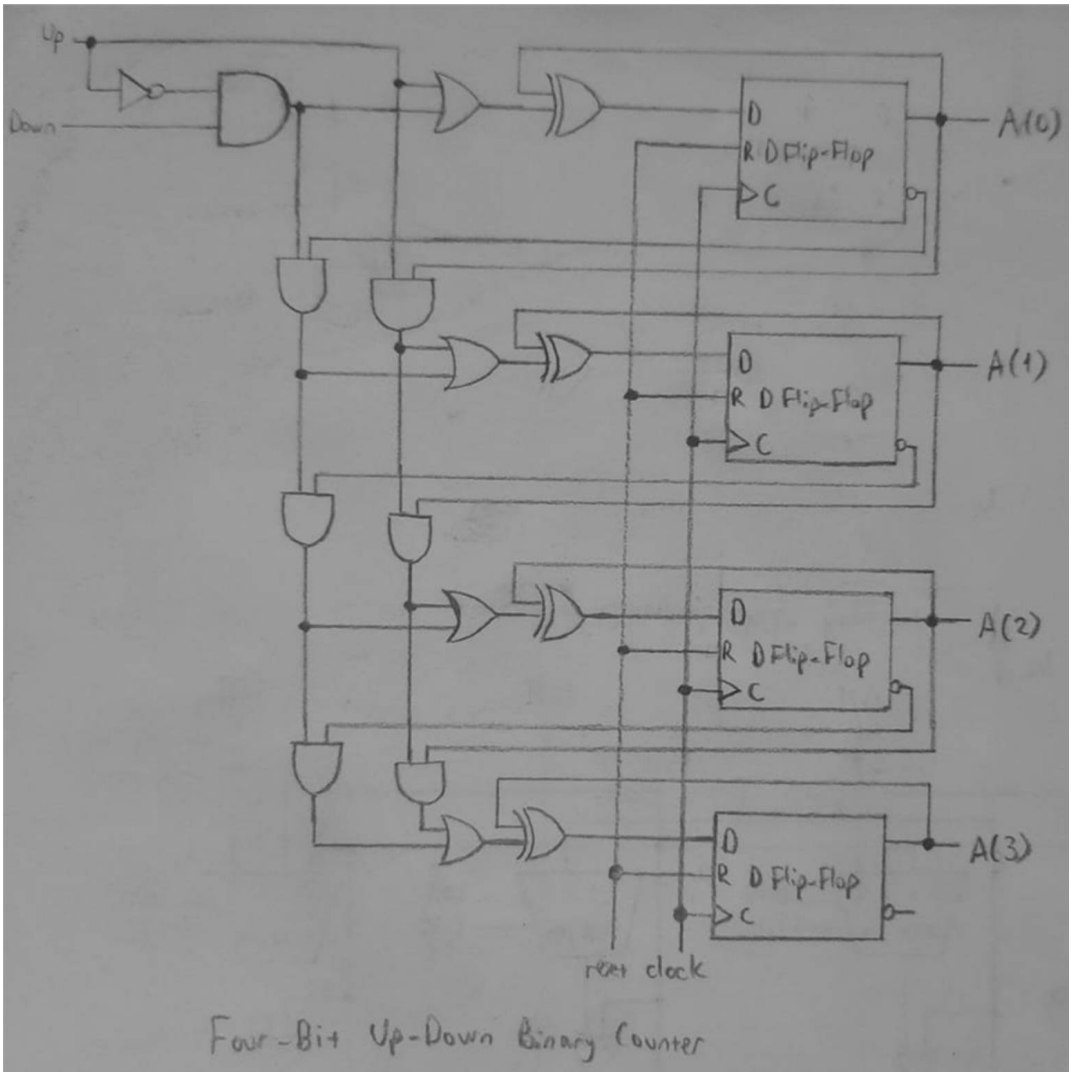


b) 1. Analytical Solution

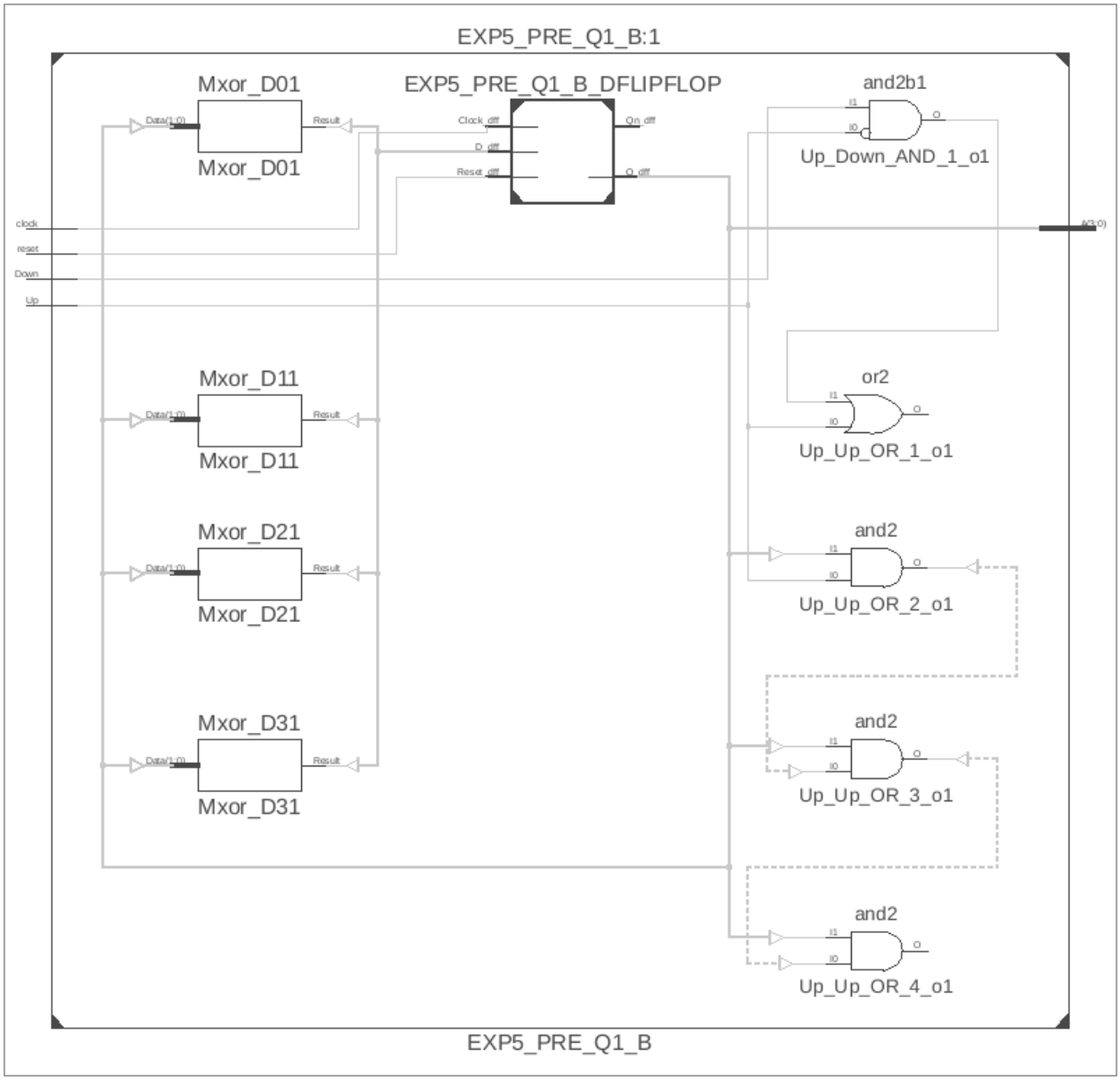
Refer to the figure in the textbook. To convert a T Flip-Flop to D Flip-Flop, equate the next state expressions.

$$\mathrm{Q_T}(t+1) = \mathrm{TQ'} + \mathrm{T'Q}, \qquad \mathrm{Q_D}(t+1) = \mathrm{D} \implies \mathrm{D} = \mathrm{TQ'} + \mathrm{T'Q} = \mathrm{T} \oplus \mathrm{Q}$$

Hand-drawn circuit and RTL schematic



The reset input is now active high, in contrast to the figure in the textbook.



2. Codes

VHDL - D Flip-Flop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q1_B_DFLIPFLOP is
    Port ( D_dff, Reset_dff, Clock_dff : in STD_LOGIC;
           Q_dff, Qn_dff : out STD_LOGIC);
end EXP5_PRE_Q1_B_DFLIPFLOP;

architecture Behavioral of EXP5_PRE_Q1_B_DFLIPFLOP is
signal tempQ : STD_LOGIC;
```

```
begin
    process (Reset_dff, Clock_dff)
    begin
        if Clock_dff'event and Clock_dff = '1' then
            if Reset_dff = '1' then tempQ <= '0';
            else tempQ <= D_dff;
            end if;
        end if;
    end process;
    Q_dff <= tempQ;
    Qn_dff <= not tempQ;
end Behavioral;
```

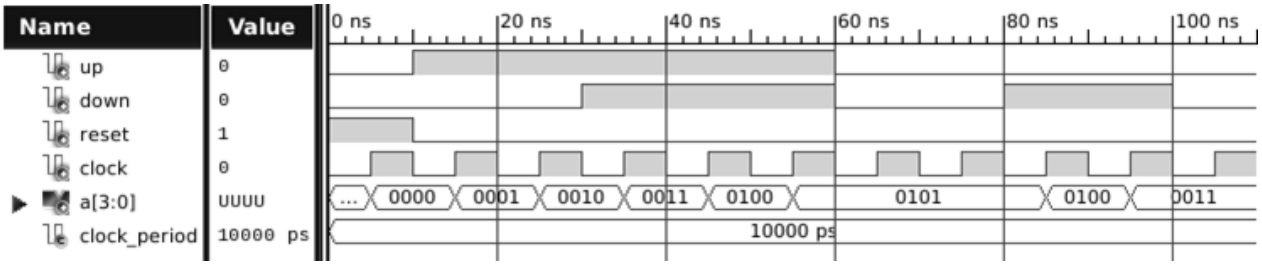
VHDL - Four-Bit Up-Down Binary Counter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q1_B is
    port ( Up, Down, reset, clock : in STD_LOGIC;
           A : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q1_B;
architecture Behavioral of EXP5_PRE_Q1_B is
component EXP5_PRE_Q1_B_DFLIPFLOP
    Port ( D_dff, Reset_dff, Clock_dff : in STD_LOGIC;
           Q_dff, Qn_dff : out STD_LOGIC);
end component;
signal tempA, tempAn : STD_LOGIC_VECTOR (3 downto 0);
signal D0, D1, D2, D3, D0n, D1n, D2n, D3n : STD_LOGIC;
begin
    D0 <= (Up or (not Up and Down)) xor tempA(0);
    D1 <= (((not Up and Down) and D0n) or (Up and tempA(0))) xor tempA(1);
    D2 <= ((((not Up and Down) and D0n) and D1n)
          or ((Up and tempA(0)) and tempA(1))) xor tempA(2);
    D3 <= (((((not Up and Down) and D0n) and D1n)
          and D2n) or (((Up and tempA(0)) and tempA(1)) and tempA(2)))
          xor tempA(3);
    dff0 : EXP5_PRE_Q1_B_DFLIPFLOP port map (D0, reset, clock, tempA(0),
                                             tempAn(0));
    dff1 : EXP5_PRE_Q1_B_DFLIPFLOP port map (D1, reset, clock, tempA(1),
                                             tempAn(1));
    dff2 : EXP5_PRE_Q1_B_DFLIPFLOP port map (D2, reset, clock, tempA(2),
                                             tempAn(2));
    dff3 : EXP5_PRE_Q1_B_DFLIPFLOP port map (D3, reset, clock, tempA(3),
                                             tempAn(3));
    A <= tempA(3) & tempA(2) & tempA(1) & tempA(0);
end Behavioral;
```

3. Results

Test bench

```
stim_proc: process
begin
    reset <= '1'; wait for clock_period;
    reset <= '0'; Up <= '1'; Down <= '0'; wait for clock_period;
    Up <= '1'; Down <= '0'; wait for clock_period;
    Up <= '1'; Down <= '1'; wait for clock_period;
    Up <= '1'; Down <= '1'; wait for clock_period;
    Up <= '1'; Down <= '1'; wait for clock_period;
    Up <= '0'; Down <= '0'; wait for clock_period;
    Up <= '0'; Down <= '0'; wait for clock_period;
    Up <= '0'; Down <= '1'; wait for clock_period;
    Up <= '0'; Down <= '1'; wait for clock_period;
    Up <= '0'; Down <= '0'; wait for clock_period;
end process;
```



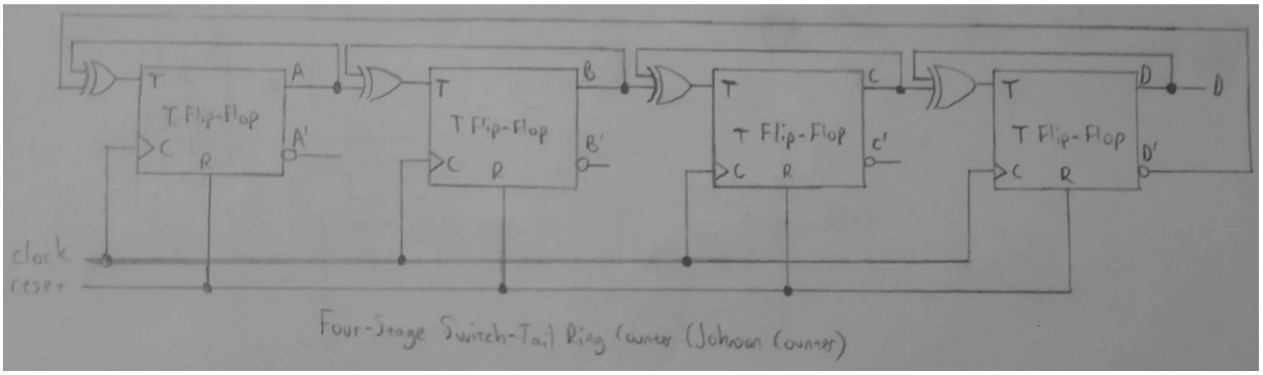
c) 1. Analytical Solution

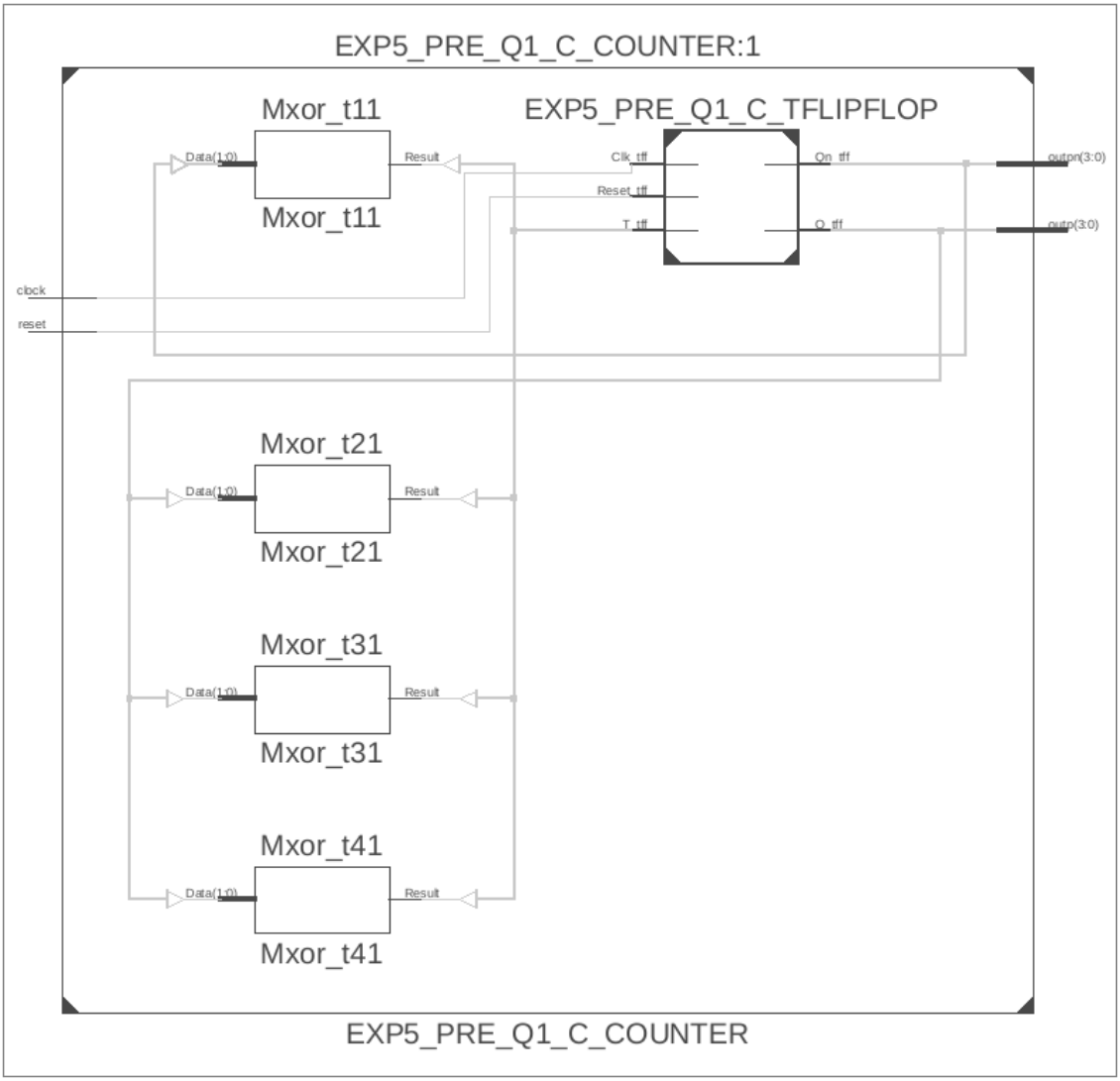
Refer to the figure in the textbook. To convert a D Flip-Flop to T Flip-Flop, equate the next state expressions. and solve for T. The reset input is now active high, in contrast to the figure in the textbook.

$$Q_T(t+1) = TQ' + T'Q, \qquad Q_D(t+1) = D \implies D = T \oplus Q$$

$$D = T \oplus Q \implies D \oplus Q = T \oplus \underbrace{(Q \oplus Q)}_{0} \implies T = D \oplus Q$$

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - Four-Stage Switch-Tail Ring Counter (Johnson Counter)

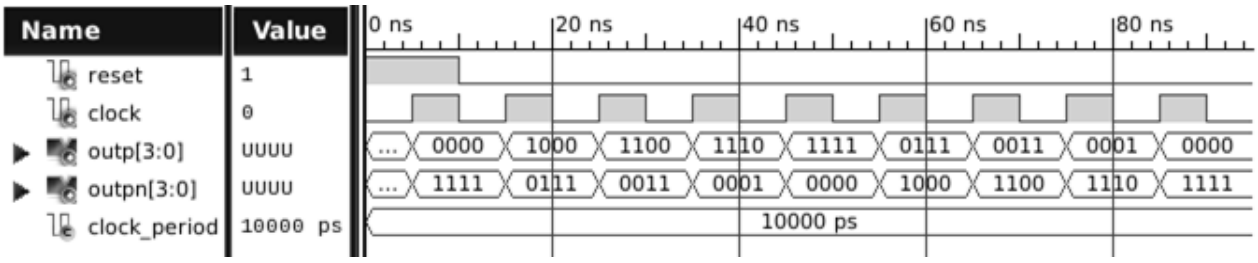
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q1_C_COUNTER is
    port (reset, clock : in STD_LOGIC;
        outp, outpn : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q1_C_COUNTER;
architecture Behavioral of EXP5_PRE_Q1_C_COUNTER is
component EXP5_PRE_Q1_A_TFLIPFLOP
    port ( T_tff, Reset_tff, Clk_tff : in std_logic;
           Q_tff, Qn_tff : out std_logic);
end component;
signal temp_outp, temp_outpn : STD_LOGIC_VECTOR (3 downto 0);
signal t3, t2, t1, t0 : STD_LOGIC;
```

```
begin
    t3<=temp_outp(3) xor temp_outpn(0); t2<=temp_outp(2) xor temp_outp(3);
    t1<=temp_outp(1) xor temp_outp(2);  t0<=temp_outp(0) xor temp_outp(1);
    tff3 : EXP5_PRE_Q1_A_TFLIPFLOP port map (t3, reset, clock,
                                             temp_outp(3), temp_outpn(3));
    tff2 : EXP5_PRE_Q1_A_TFLIPFLOP port map (t2, reset, clock,
                                             temp_outp(2), temp_outpn(2));
    tff1 : EXP5_PRE_Q1_A_TFLIPFLOP port map (t1, reset, clock,
                                             temp_outp(1), temp_outpn(1));
    tff0 : EXP5_PRE_Q1_A_TFLIPFLOP port map (t0, reset, clock,
                                             temp_outp(0), temp_outpn(0));
    outp <= temp_outp; outpn <= temp_outpn;
end Behavioral;
```

3. Results

Test bench

```
stim_proc: process
begin
  reset <= '1'; wait for clock_period; reset <= '0'; wait for clock_period;
  wait for clock_period; wait for clock_period; wait for clock_period;
  wait for clock_period; wait for clock_period; wait for clock_period;
  wait for clock_period; wait for clock_period;
end process;
```



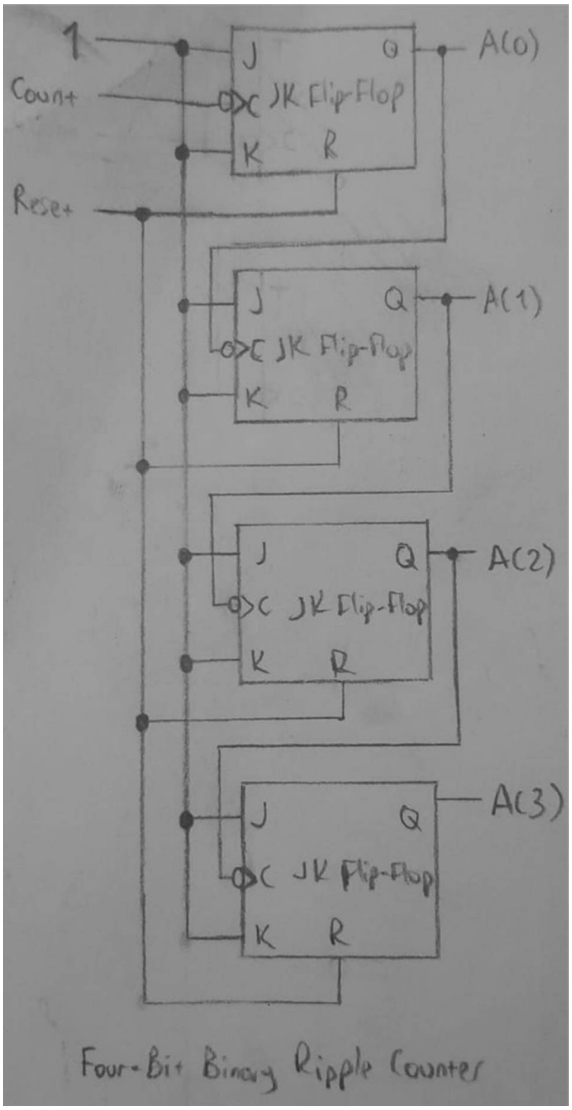
d) 1. Analytical Solution

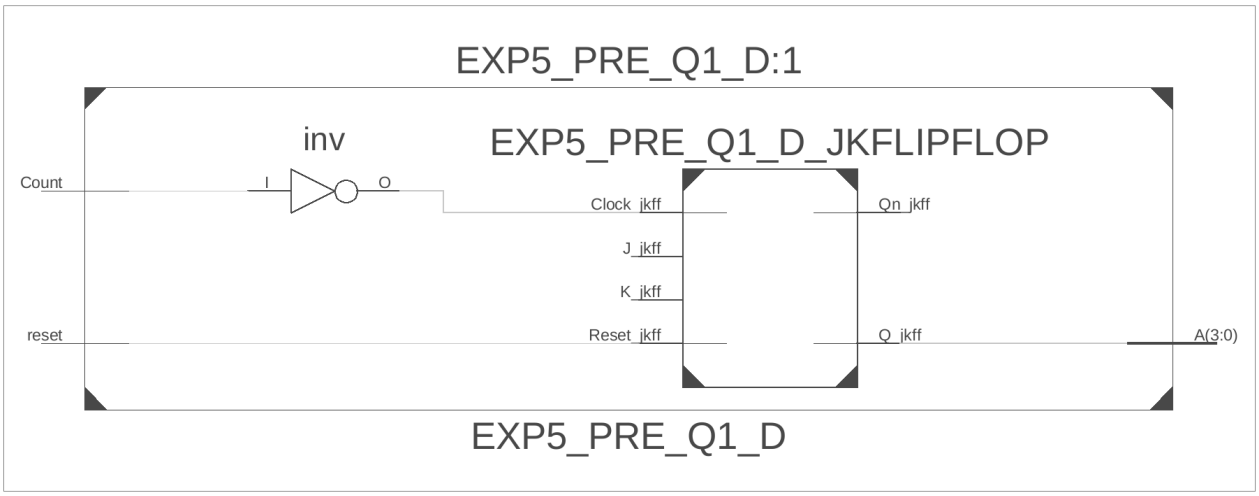
Refer to the figure in the textbook. To convert a T Flip-Flop to JK Flip-Flop, we just use the J and K inputs instead of T.

$$J = T,\ K = T$$

The reset input is now active high, in contrast to the figure in the textbook.

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - JK Flip-Flop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q1_D_JKFLIPFLOP is
    Port ( J_jkff, K_jkff, Reset_jkff, Clock_jkff : in STD_LOGIC;
           Q_jkff, Qn_jkff : out STD_LOGIC);
end EXP5_PRE_Q1_D_JKFLIPFLOP;
architecture Behavioral of EXP5_PRE_Q1_D_JKFLIPFLOP is
signal tempQ: std_logic;
begin
    process (Reset_jkff, Clock_jkff)
    begin
      if Clock_jkff'event and Clock_jkff = '1' then
            if Reset_jkff = '1' then tempQ <= '0';
                elsif (J_jkff='0' and K_jkff='0') then tempQ <= tempQ;
                elsif (J_jkff='0' and K_jkff='1') then tempQ <= '0';
                elsif (J_jkff='1' and K_jkff='0') then tempQ <= '1';
                elsif (J_jkff='1' and K_jkff='1') then tempQ <= not (tempQ);
            end if;
        end if;
    end process;
    Q_jkff <= tempQ; Qn_jkff <= not tempQ;
end Behavioral;
```

VHDL - Four-Bit Binary Ripple Counter

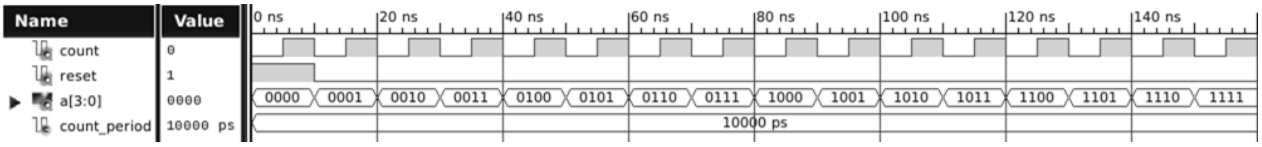
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q1_D is
    port (Count, reset : in STD_LOGIC;
          A : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q1_D;
architecture Behavioral of EXP5_PRE_Q1_D is
component EXP5_PRE_Q1_D_JKFLIPFLOP
    Port ( J_jkff, K_jkff, Reset_jkff, Clock_jkff : in STD_LOGIC;
           Q_jkff, Qn_jkff : out STD_LOGIC);
end component;
signal c_ff, tempA, tempAn : STD_LOGIC_VECTOR (3 downto 0);
begin
    c_ff(0) <= not Count; c_ff(1) <= not tempA(0);
    c_ff(2) <= not tempA(1); c_ff(3) <= not tempA(2);
    jkff0 : EXP5_PRE_Q1_D_JKFLIPFLOP port map ('1', '1', reset, c_ff(0),
                                               tempA(0), tempAn(0));
    jkff1 : EXP5_PRE_Q1_D_JKFLIPFLOP port map ('1', '1', reset, c_ff(1),
                                               tempA(1), tempAn(1));
```

```
    jkff2 : EXP5_PRE_Q1_D_JKFLIPFLOP port map ('1', '1', reset, c_ff(2),
                                                tempA(2), tempAn(2));
    jkff3 : EXP5_PRE_Q1_D_JKFLIPFLOP port map ('1', '1', reset, c_ff(3),
                                                tempA(3), tempAn(3));
    A <= tempA;
end Behavioral;
```

3. Results

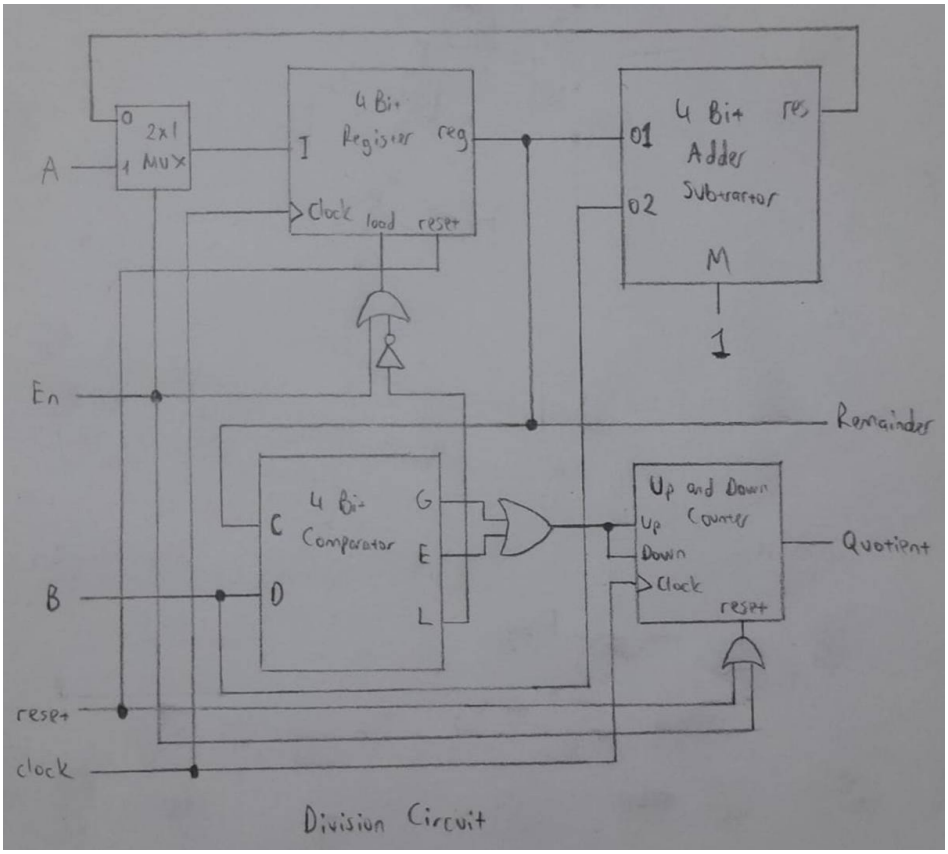
Test bench

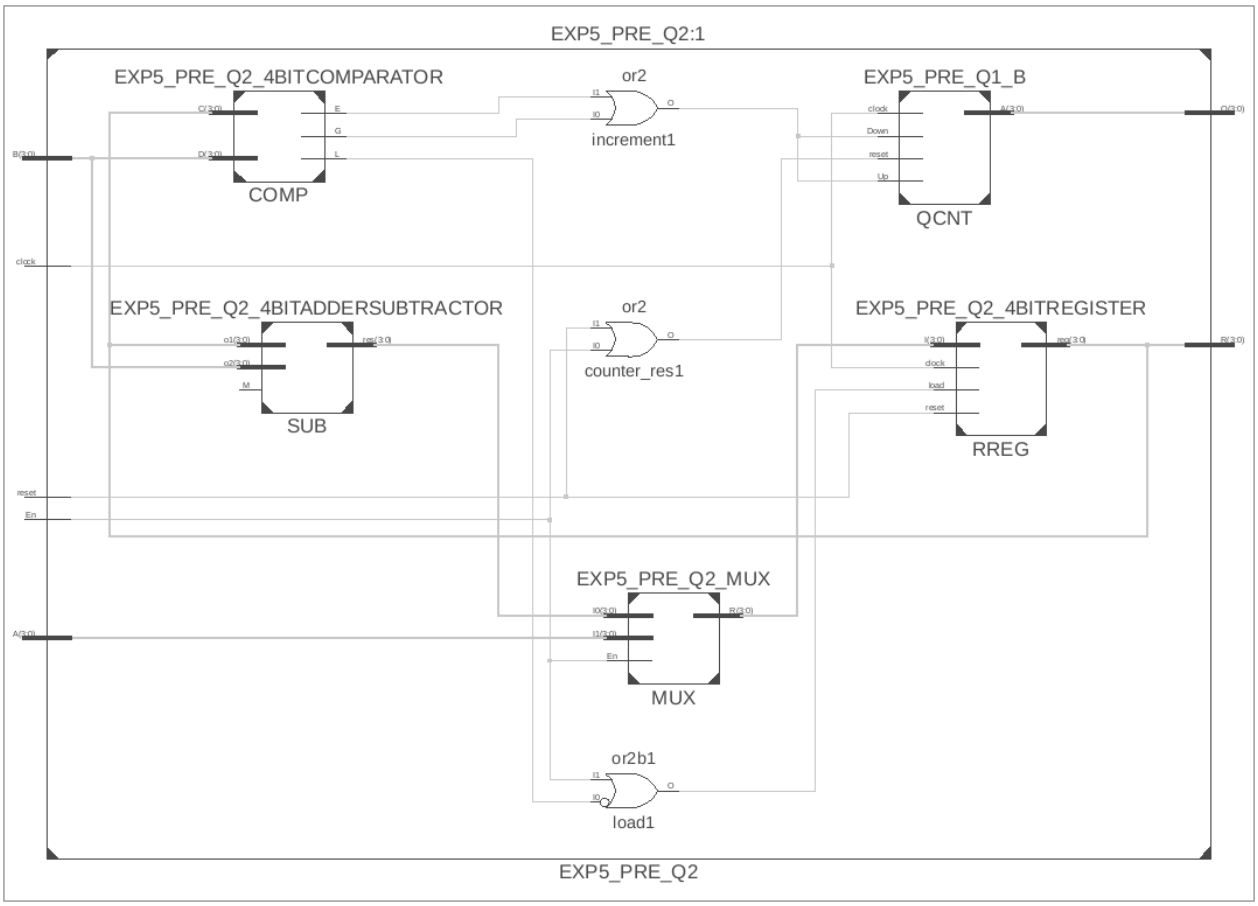
```
stim_proc: process
begin
    Reset <= '1'; wait for Count_period;
    Reset <= '0'; wait for Count_period*15;
end process;
```



Q2: 1. Analytical Solution

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - Four-Bit Register

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q2_4BITREGISTER is
    port ( I : in STD_LOGIC_VECTOR (3 downto 0);
           load, reset, clock : in STD_LOGIC;
           reg : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q2_4BITREGISTER;

architecture Behavioral of EXP5_PRE_Q2_4BITREGISTER is
component EXP5_PRE_Q1_B_DFLIPFLOP
    Port ( D_dff, Reset_dff, Clock_dff : in STD_LOGIC;
           Q_dff, Qn_dff : out STD_LOGIC);
end component;
signal tempreg0, tempreg1, tempreg2, tempreg3 : STD_LOGIC;
signal tempregn0, tempregn1, tempregn2, tempregn3 : STD_LOGIC;
signal d0, d1, d2, d3 : STD_LOGIC;
```

```
begin
  d0 <= (not load and tempreg0) or (I(0) and load);
  d1 <= (not load and tempreg1) or (I(1) and load);
  d2 <= (not load and tempreg2) or (I(2) and load);
  d3 <= (not load and tempreg3) or (I(3) and load);

  dff0:EXP5_PRE_Q1_B_DFLIPFLOP port map(d0,reset,clock,tempreg0,tempregn0);
  dff1:EXP5_PRE_Q1_B_DFLIPFLOP port map(d1,reset,clock,tempreg1,tempregn1);
  dff2:EXP5_PRE_Q1_B_DFLIPFLOP port map(d2,reset,clock,tempreg2,tempregn2);
  dff3:EXP5_PRE_Q1_B_DFLIPFLOP port map(d3,reset,clock,tempreg3,tempregn3);

  reg <= tempreg3 & tempreg2 & tempreg1 & tempreg0;
end Behavioral;
```

VHDL - Four-Bit Adder Subtractor

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q2_4BITADDERSUBTRACTOR is
    Port (o1, o2 : in  STD_LOGIC_VECTOR (3 downto 0);
                M : in  STD_LOGIC;
              res : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q2_4BITADDERSUBTRACTOR;

architecture Behavioral of EXP5_PRE_Q2_4BITADDERSUBTRACTOR is
component EXP5_PRE_Q2_FA
    Port (fa_a, fa_b, fa_c_in : in  STD_LOGIC;
          fa_s, fa_c_out : out STD_LOGIC);
end component;
signal b, c_out : STD_LOGIC_VECTOR (3 downto 0);

begin
    b(0) <= o2(0) xor M; b(1) <= o2(1) xor M;
    b(2) <= o2(2) xor M; b(3) <= o2(3) xor M;

    FA1 : EXP5_PRE_Q2_FA port map (o1(0), b(0), M, res(0), c_out(0));
    FA2 : EXP5_PRE_Q2_FA port map (o1(1), b(1), c_out(0), res(1), c_out(1));
    FA3 : EXP5_PRE_Q2_FA port map (o1(2), b(2), c_out(1), res(2), c_out(2));
  FA4 : EXP5_PRE_Q2_FA port map (o1(3), b(3), c_out(2), res(3), c_out(3));
end Behavioral;
```

VHDL - MUX

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity EXP5_PRE_Q2_MUX is
    port ( I0, I1 : in STD_LOGIC_VECTOR (3 downto 0);
           En : in  STD_LOGIC;
           R : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q2_MUX;

architecture Behavioral of EXP5_PRE_Q2_MUX is
begin
    R(0) <= (I1(0) and En) or (I0(0) and not En);
    R(1) <= (I1(1) and En) or (I0(1) and not En);
    R(2) <= (I1(2) and En) or (I0(2) and not En);
    R(3) <= (I1(3) and En) or (I0(3) and not En);
end Behavioral;
```

VHDL - Four-Bit Comparator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q2_4BITCOMPARATOR is
    port ( C, D : in STD_LOGIC_VECTOR (3 downto 0);
         G, E, L : out STD_LOGIC);
end EXP5_PRE_Q2_4BITCOMPARATOR;

architecture Behavioral of EXP5_PRE_Q2_4BITCOMPARATOR is
component EXP5_PRE_Q2_DECODER
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0); En : in STD_LOGIC;
           Z : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal D1_in, D2_in, D3_in, D4_in : STD_LOGIC_VECTOR(1 downto 0);
signal D1_out, D2_out, D3_out, D4_out : STD_LOGIC_VECTOR (3 downto 0);
signal En2, En3, En4 : STD_LOGIC;
begin
    D1_in <= C(3) & D(3); D2_in <= C(2) & D(2);
    D3_in <= C(1) & D(1); D4_in <= C(0) & D(0);
    decoder1 : EXP5_PRE_Q2_DECODER port map(D1_in, '1', D1_out);
    En2 <= D1_out(0) or D1_out(3);
    decoder2 : EXP5_PRE_Q2_DECODER port map(D2_in, En2, D2_out);
    En3 <= D2_out(0) or D2_out(3);
    decoder3 : EXP5_PRE_Q2_DECODER port map(D3_in, En3, D3_out);
    En4 <= D3_out(0) or D3_out(3);
    decoder4 : EXP5_PRE_Q2_DECODER port map(D4_in, En4, D4_out);
    G <= D1_out(2) or D2_out(2) or D3_out(2) or D4_out(2);
    E <= D4_out(0) or D4_out(3);
    L <= D1_out(1) or D2_out(1) or D3_out(1) or D4_out(1);
end Behavioral;
```

VHDL - Division Circuit

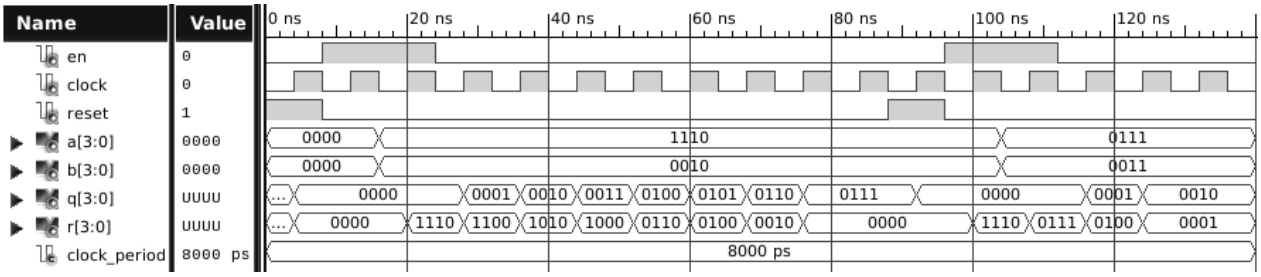
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q2 is
     port ( En, clock, reset : in  STD_LOGIC;
            A, B : in  STD_LOGIC_VECTOR(3 downto 0);
            Q, R : out STD_LOGIC_VECTOR(3 downto 0));
end EXP5_PRE_Q2;
architecture Behavioral of EXP5_PRE_Q2 is
component EXP5_PRE_Q2_4BITREGISTER
    port ( I : in STD_LOGIC_VECTOR (3 downto 0);
           load, reset, clock : in STD_LOGIC;
           reg : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component EXP5_PRE_Q2_4BITADDERSUBTRACTOR
    Port (o1, o2 : in  STD_LOGIC_VECTOR (3 downto 0);
          M : in  STD_LOGIC;
          res : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component EXP5_PRE_Q1_B
    port ( Up, Down, reset, clock : in STD_LOGIC;
           A : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component EXP5_PRE_Q2_4BITCOMPARATOR
    port ( C, D : in STD_LOGIC_VECTOR (3 downto 0);
           G, E, L : out STD_LOGIC);
end component;
component EXP5_PRE_Q2_MUX
    port ( I0, I1 : in STD_LOGIC_VECTOR (3 downto 0);
           En : in  STD_LOGIC;
           R : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal R_reg : STD_LOGIC_VECTOR(3 downto 0);
signal R_next, Q_reg, mux_out : STD_LOGIC_VECTOR(3 downto 0);
signal increment : STD_LOGIC;
signal G, E, L : STD_LOGIC;
signal counter_res, load : STD_LOGIC;
begin
  COMP: EXP5_PRE_Q2_4BITCOMPARATOR port map (R_reg, B, G, E, L);
  MUX : EXP5_PRE_Q2_MUX port map (R_next, A, En, mux_out);
  increment <= G or E;
  counter_res <= En or reset;
  load <= En or not L;
  SUB: EXP5_PRE_Q2_4BITADDERSUBTRACTOR port map (R_reg, B, '1', R_next);
  RREG:EXP5_PRE_Q2_4BITREGISTER port map(mux_out,load,reset,clock,R_reg);
  QCNT:EXP5_PRE_Q1_B port map(increment,increment,counter_res,clock,Q_reg);
```

```
  R <= R_reg;
  Q <= Q_reg;
end Behavioral;
```

3. Results

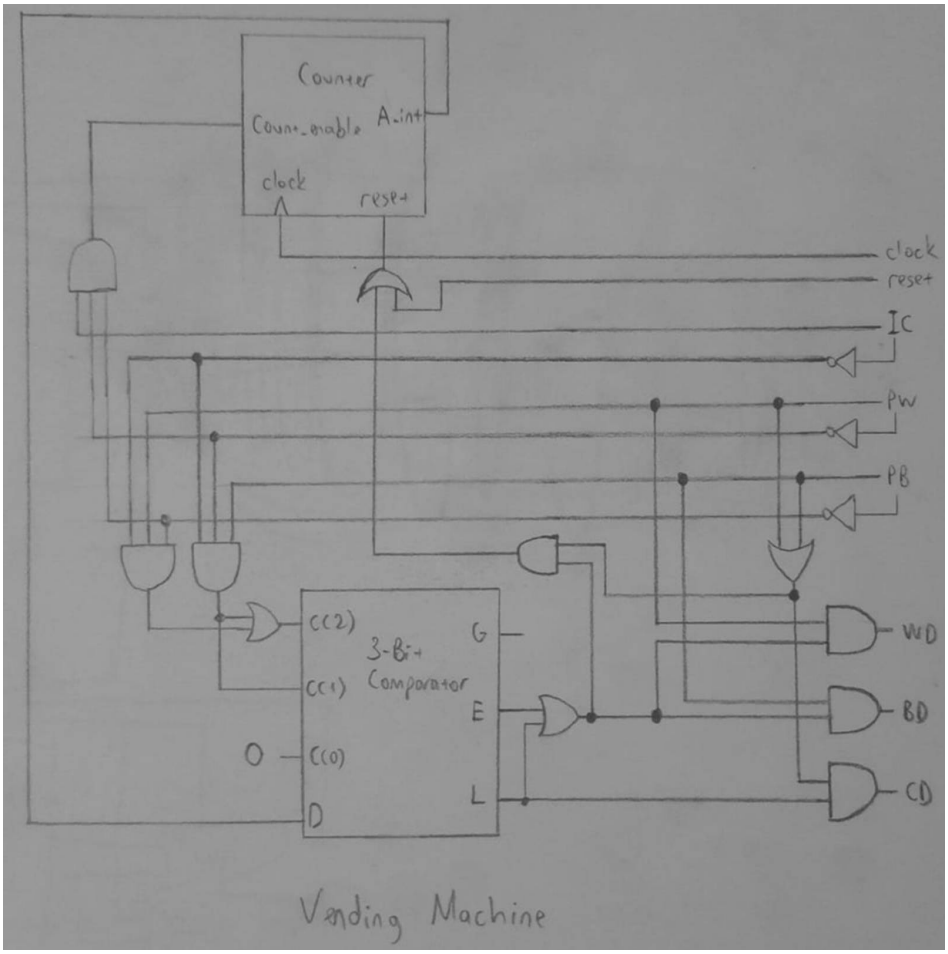
Test bench

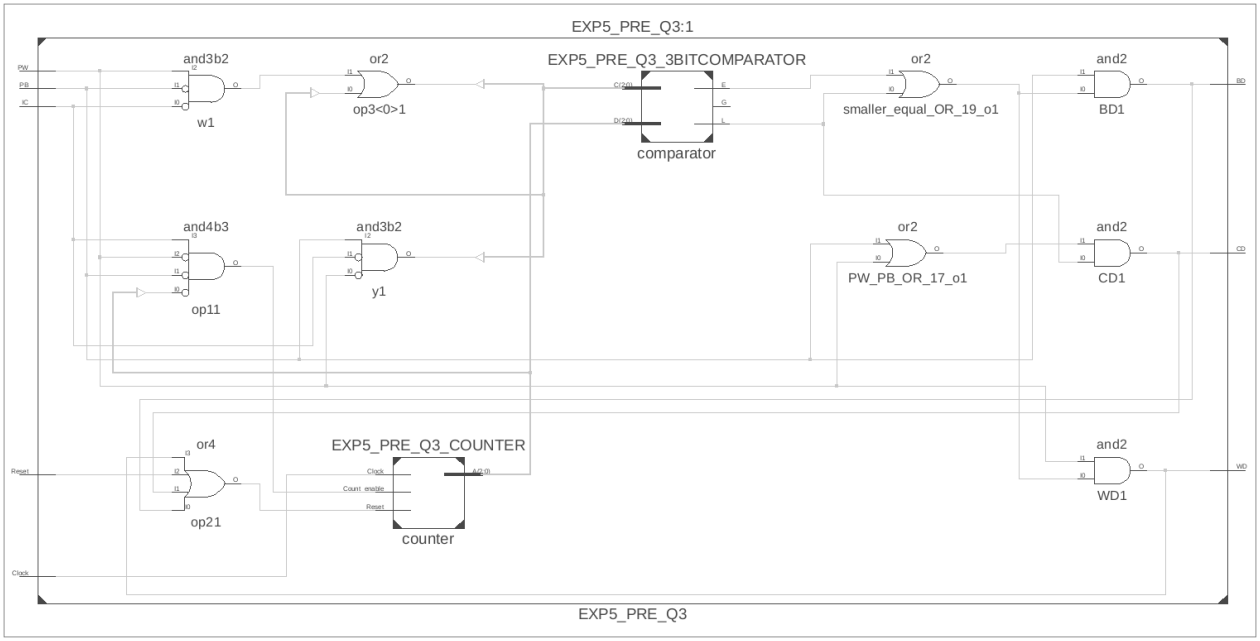
```
stim_proc: process
begin
    reset <= '1'; wait for clock_period;
    reset <= '0'; En <= '1'; wait for clock_period;
    A <= "1110"; B <= "0010" ; En <= '1'; wait for clock_period;
    En <= '0'; wait for clock_period*8;

    reset <= '1'; wait for clock_period;
    reset <= '0'; En <= '1'; wait for clock_period;
    A <= "0111"; B <= "0011" ; En <= '1'; wait for clock_period;
    En <= '0'; wait for clock_period*3.5;
end process;
```



Q3: 1. Analytical Solution

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - Three-Bit Comparator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q3_3BITCOMPARATOR is
    port ( C, D : in STD_LOGIC_VECTOR (2 downto 0);
           G, E, L : out STD_LOGIC);
end EXP5_PRE_Q3_3BITCOMPARATOR;

architecture Behavioral of EXP5_PRE_Q3_3BITCOMPARATOR is
component EXP5_PRE_Q2_DECODER
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0); En : in STD_LOGIC;
           Z : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal D1_in, D2_in, D3_in : STD_LOGIC_VECTOR(1 downto 0);
signal D1_out, D2_out, D3_out : STD_LOGIC_VECTOR (3 downto 0);
signal En2, En3 : STD_LOGIC;
begin
    D1_in <= C(2) & D(2); D2_in <= C(1) & D(1); D3_in <= C(0) & D(0);
    decoder1 : EXP5_PRE_Q2_DECODER port map(D1_in, '1', D1_out);
    En2 <= D1_out(0) or D1_out(3);
    decoder2 : EXP5_PRE_Q2_DECODER port map(D2_in, En2, D2_out);
    En3 <= D2_out(0) or D2_out(3);
    decoder3 : EXP5_PRE_Q2_DECODER port map(D3_in, En3, D3_out);
    G <= D1_out(2) or D2_out(2) or D3_out(2);
    E <= D3_out(0) or D3_out(3);
    L <= D1_out(1) or D2_out(1) or D3_out(1);
end Behavioral;
```

VHDL - Counter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q3_COUNTER is
    Port ( Clock, Reset, Count_enable : in STD_LOGIC;
           A : out STD_LOGIC_VECTOR(2 downto 0));
end EXP5_PRE_Q3_COUNTER;

architecture Behavioral of EXP5_PRE_Q3_COUNTER is
component EXP5_PRE_Q1_D_JKFLIPFLOP
    Port ( J_jkff, K_jkff, Reset_jkff, Clock_jkff : in STD_LOGIC;
           Q_jkff, Qn_jkff : out STD_LOGIC);
end component;
signal A_int, An_int : STD_LOGIC_VECTOR(2 downto 0);
```

```
signal op1, op2 : STD_LOGIC;
begin
  op1 <= Count_enable and A_int(0);
  op2 <= op1 and A_int(1);
  JK_FF0:EXP5_PRE_Q1_D_JKFLIPFLOP port map(Count_enable,Count_enable,Reset,
                                          Clock, A_int(0), An_int(0));
  JK_FF1:EXP5_PRE_Q1_D_JKFLIPFLOP port map(op1, op1, Reset,
                                          Clock, A_int(1), An_int(1));
  JK_FF2:EXP5_PRE_Q1_D_JKFLIPFLOP port map(op2, op2, Reset,
                                          Clock, A_int(2), An_int(2));
  A <= A_int;
end Behavioral;
```

VHDL - Vending Machine

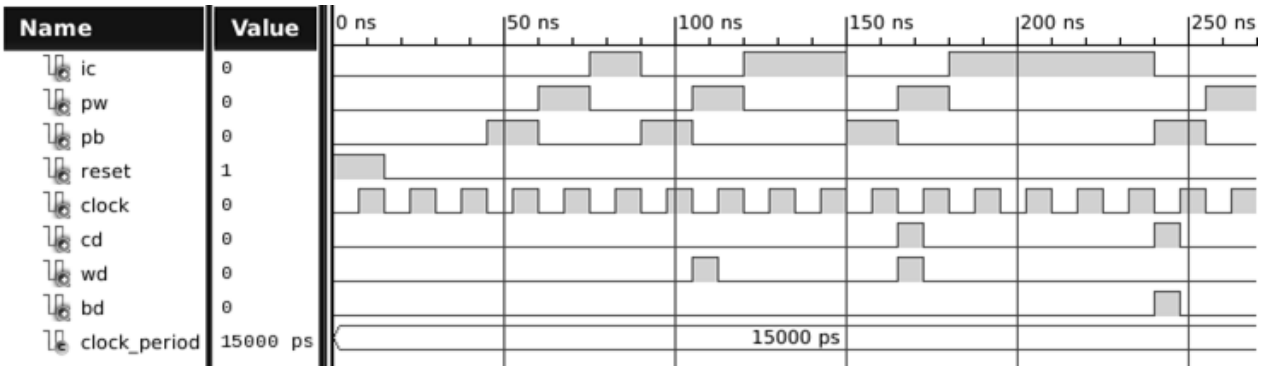
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity EXP5_PRE_Q3 is
    Port ( IC, PW, PB, Reset, Clock: in STD_LOGIC;
           CD, WD, BD: out STD_LOGIC);
end EXP5_PRE_Q3;

architecture Behavioral of EXP5_PRE_Q3 is
component EXP5_PRE_Q3_COUNTER
    Port ( Clock, Reset, Count_enable: in STD_LOGIC;
           A: out STD_LOGIC_VECTOR(2 downto 0));
end component;
component EXP5_PRE_Q3_3BITCOMPARATOR
    port ( C, D : in STD_LOGIC_VECTOR (2 downto 0);
           G, E, L : out STD_LOGIC);
end component;

signal x, w, y, greater, equal, less, output_flag : STD_LOGIC;
signal A_int, w_vector, y_vector : STD_LOGIC_VECTOR(2 downto 0);
signal op1, op2 : STD_LOGIC;
signal op3 : STD_LOGIC_VECTOR (2 downto 0);

begin
    x <= IC and not PW and not PB;
    w <= not IC and PW and not PB;
    y <= not IC and not PW and PB;
    op1 <= x and not A_int(2);
    op2 <= Reset or output_flag;
    op3 <= y_vector or w_vector;

    counter: EXP5_PRE_Q3_COUNTER port map(Clock, op2, op1, A_int);
```

```
    w_vector(0) <= w; w_vector(1) <= '0'; w_vector(2) <= '0';
    y_vector(0) <= y; y_vector(1) <= y; y_vector(2) <= '0';

    comparator : EXP5_PRE_Q3_3BITCOMPARATOR port map(op3, A_int, greater,
                                                     equal, less);
    output_flag <= (PW or PB) and (less or equal);
    CD <= less and (PW or PB);
    BD <= (less or equal) and PB;
    WD <= (less or equal) and PW;
end Behavioral;
```

3. Results

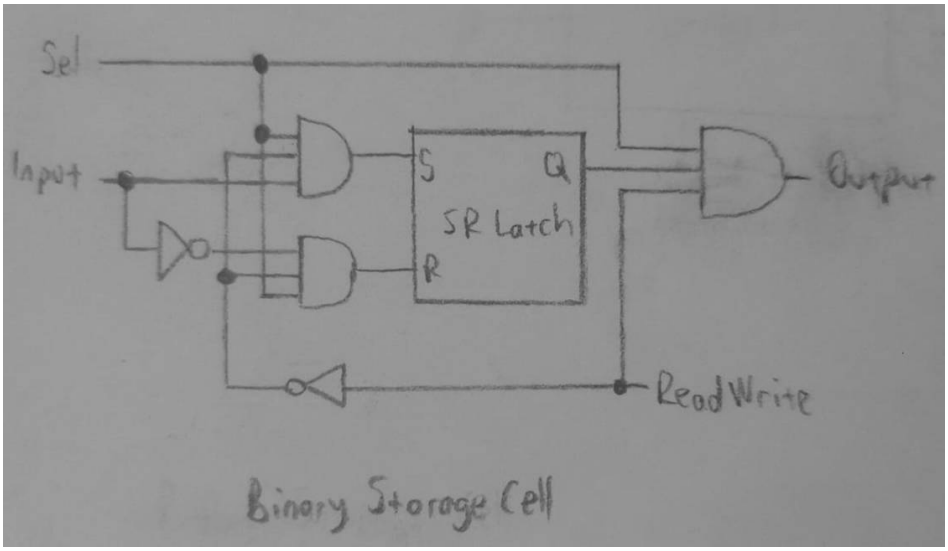
Test bench

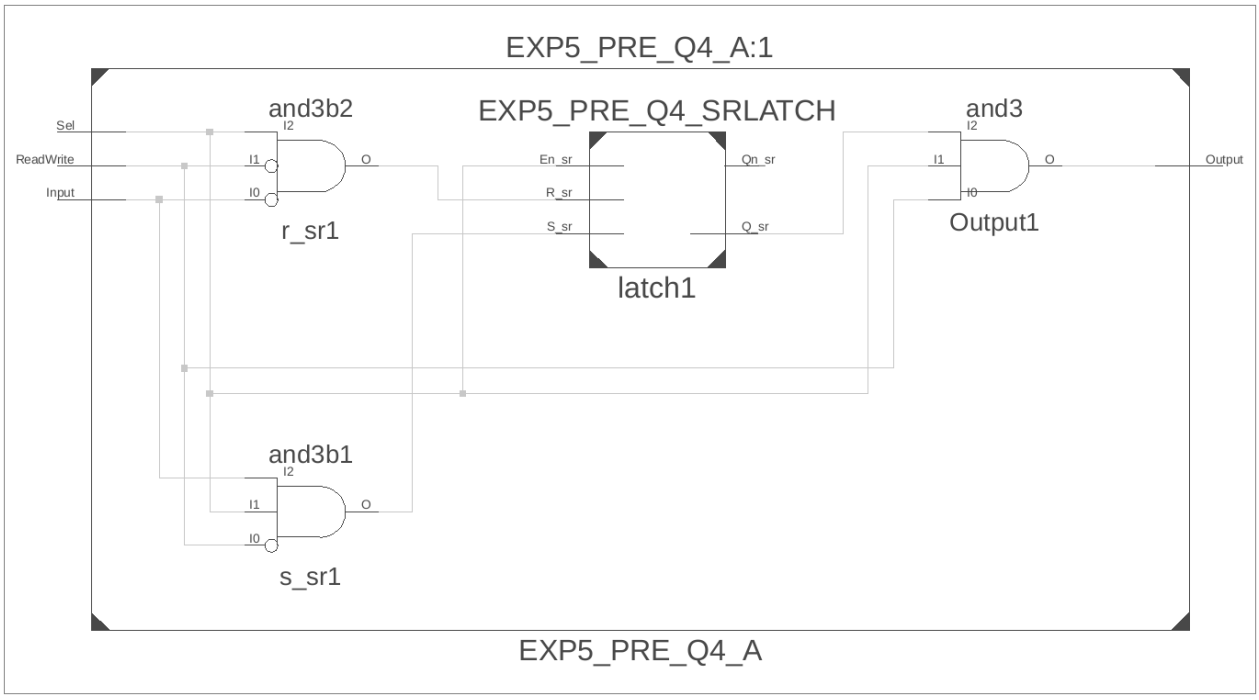
```
stim_proc: process
begin
    reset <= '1'; wait for clock_period;
    reset <= '0'; wait for clock_period;
    IC <= '0'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '0'; PW <= '0'; PB <= '1'; wait for Clock_period;
    IC <= '0'; PW <= '1'; PB <= '0'; wait for Clock_period;
    IC <= '1'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '0'; PW <= '0'; PB <= '1'; wait for Clock_period;
    IC <= '0'; PW <= '1'; PB <= '0'; wait for Clock_period;
    IC <= '1'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '1'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '0'; PW <= '0'; PB <= '1'; wait for Clock_period;
    IC <= '0'; PW <= '1'; PB <= '0'; wait for Clock_period;
    IC <= '1'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '1'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '1'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '1'; PW <= '0'; PB <= '0'; wait for Clock_period;
    IC <= '0'; PW <= '0'; PB <= '1'; wait for Clock_period;
    IC <= '0'; PW <= '1'; PB <= '0'; wait for Clock_period;
end process;
```



Q4: a) 1. Analytical Solution

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - SR Latch

```
library ieee;
use ieee.std_logic_1164.all;
entity EXP5_PRE_Q4_SRLATCH is
    port ( S_sr, R_sr, En_sr : in std_logic; Q_sr, Qn_sr : out std_logic);
end EXP5_PRE_Q4_SRLATCH;
```

```
architecture Behavioral of EXP5_PRE_Q4_SRLATCH is
signal temp_Q, temp_Qn : STD_LOGIC;
begin
    temp_Q <= (S_sr nand En_sr) nand temp_Qn;
    temp_Qn <= (R_sr nand En_sr) nand temp_Q;
    Q_sr <= temp_Q; Qn_sr <= temp_Qn;
end Behavioral;
```

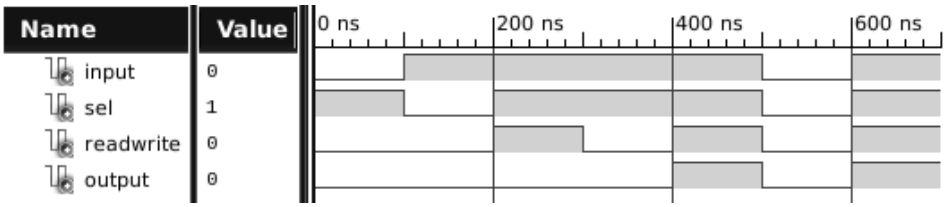
VHDL - Binary Storage Cell

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q4_A is
    Port ( Input, Sel, ReadWrite : in STD_LOGIC;
           Output : out STD_LOGIC);
end EXP5_PRE_Q4_A;
architecture Behavioral of EXP5_PRE_Q4_A is
component EXP5_PRE_Q4_SRLATCH
    port ( S_sr, R_sr, En_sr : in  std_logic; Q_sr, Qn_sr : out std_logic);
end component;
signal s_sr, r_sr, sr_outp, sr_outpn : STD_LOGIC;
begin
    s_sr <= not ReadWrite and Input and Sel;
    r_sr <= not ReadWrite and not Input and Sel;
    latch1:EXP5_PRE_Q4_SRLATCH port map(s_sr, r_sr, Sel, sr_outp, sr_outpn);
    Output <= Sel and ReadWrite and sr_outp;
end Behavioral;
```

3. Results

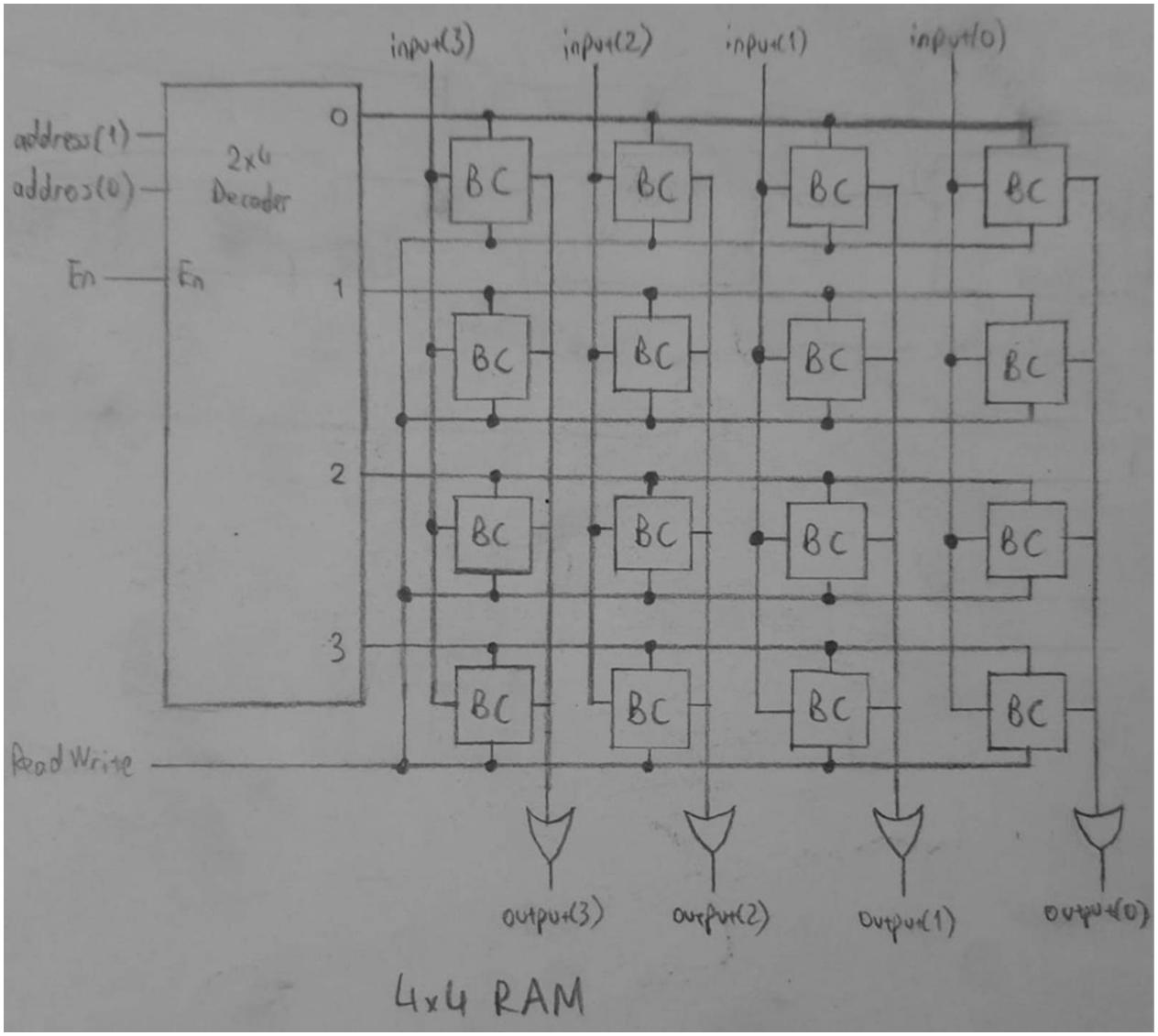
Test bench

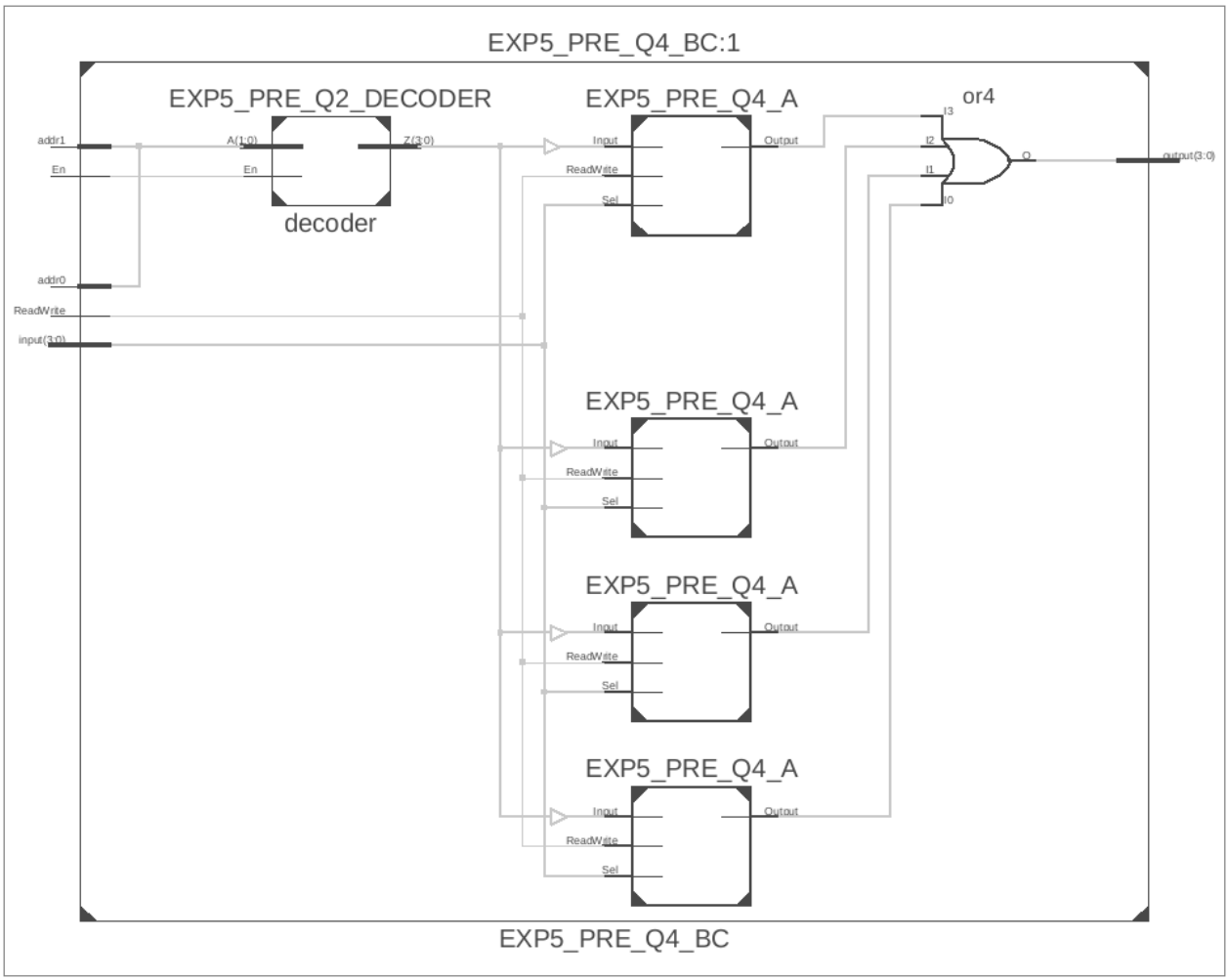
```
stim_proc: process
begin
    Input <= '0'; Sel <= '1'; ReadWrite <= '0'; wait for 100 ns;
    Input <= '1'; Sel <= '0'; ReadWrite <= '0'; wait for 100 ns;
    Input <= '1'; Sel <= '1'; ReadWrite <= '1'; wait for 100 ns;
    Input <= '1'; Sel <= '1'; ReadWrite <= '0'; wait for 100 ns;
    Input <= '1'; Sel <= '1'; ReadWrite <= '1'; wait for 100 ns;
    Input <= '0'; Sel <= '0'; ReadWrite <= '0'; wait for 100 ns;
    Input <= '1'; Sel <= '1'; ReadWrite <= '1'; wait for 100 ns;
end process;
```



Q4: b) c) 1. Analytical Solution

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - RAM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q4_BC is
    port (ReadWrite, En : in STD_LOGIC;
          address : in STD_LOGIC_VECTOR (1 downto 0);
          input : in STD_LOGIC_VECTOR (3 downto 0);
          output : out STD_LOGIC_VECTOR (3 downto 0));
end EXP5_PRE_Q4_BC;
architecture Behavioral of EXP5_PRE_Q4_BC is
component EXP5_PRE_Q2_DECODER
    port ( A : in STD_LOGIC_VECTOR (1 downto 0); En : in STD_LOGIC;
           Z : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component EXP5_PRE_Q4_A
    port ( Input, Sel, ReadWrite : in STD_LOGIC;
           Output : out STD_LOGIC);
```

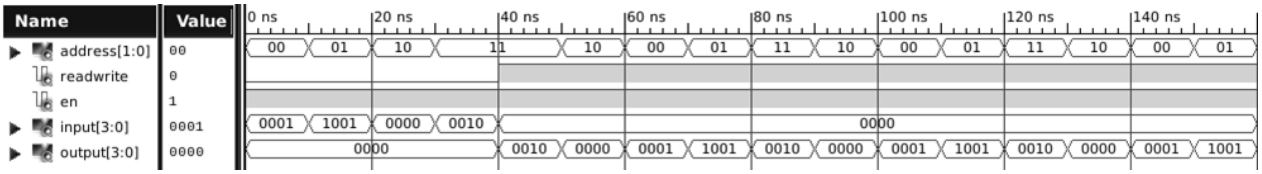
```
end component;
signal pos, outp0, outp1, outp2, outp3 : STD_LOGIC_VECTOR (3 downto 0);
begin
    decoder : EXP5_PRE_Q2_DECODER port map (address, En, pos);

    bc00_1 : EXP5_PRE_Q4_A port map(input(0), pos(0), ReadWrite, outp0(0));
    bc00_2 : EXP5_PRE_Q4_A port map(input(1), pos(0), ReadWrite, outp0(1));
    bc00_3 : EXP5_PRE_Q4_A port map(input(2), pos(0), ReadWrite, outp0(2));
    bc00_4 : EXP5_PRE_Q4_A port map(input(3), pos(0), ReadWrite, outp0(3));

    bc01_1 : EXP5_PRE_Q4_A port map(input(0), pos(1), ReadWrite, outp1(0));
    bc01_2 : EXP5_PRE_Q4_A port map(input(1), pos(1), ReadWrite, outp1(1));
    bc01_3 : EXP5_PRE_Q4_A port map(input(2), pos(1), ReadWrite, outp1(2));
    bc01_4 : EXP5_PRE_Q4_A port map(input(3), pos(1), ReadWrite, outp1(3));

    bc10_1 : EXP5_PRE_Q4_A port map(input(0), pos(2), ReadWrite, outp2(0));
    bc10_2 : EXP5_PRE_Q4_A port map(input(1), pos(2), ReadWrite, outp2(1));
    bc10_3 : EXP5_PRE_Q4_A port map(input(2), pos(2), ReadWrite, outp2(2));
    bc10_4 : EXP5_PRE_Q4_A port map(input(3), pos(2), ReadWrite, outp2(3));

    bc11_1 : EXP5_PRE_Q4_A port map(input(0), pos(3), ReadWrite, outp3(0));
    bc11_2 : EXP5_PRE_Q4_A port map(input(1), pos(3), ReadWrite, outp3(1));
    bc11_3 : EXP5_PRE_Q4_A port map(input(2), pos(3), ReadWrite, outp3(2));
    bc11_4 : EXP5_PRE_Q4_A port map(input(3), pos(3), ReadWrite, outp3(3));

    output <= outp0 or outp1 or outp2 or outp3;
end Behavioral;
```

3. Results

Test bench

```
stim_proc: process
begin
    address <= "00"; ReadWrite <= '0'; En <= '1'; input <= "0001";
    wait for 10 ns;
    address <= "01"; ReadWrite <= '0'; En <= '1'; input <= "1001";
    wait for 10 ns;
    address <= "10"; ReadWrite <= '0'; En <= '1'; input <= "0000";
    wait for 10 ns;
    address <= "11"; ReadWrite <= '0'; En <= '1'; input <= "0010";
    wait for 10 ns;
    input <= "0000"; -- redundant but improves the graph
    address <= "11"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "10"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "00"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "01"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "11"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
```

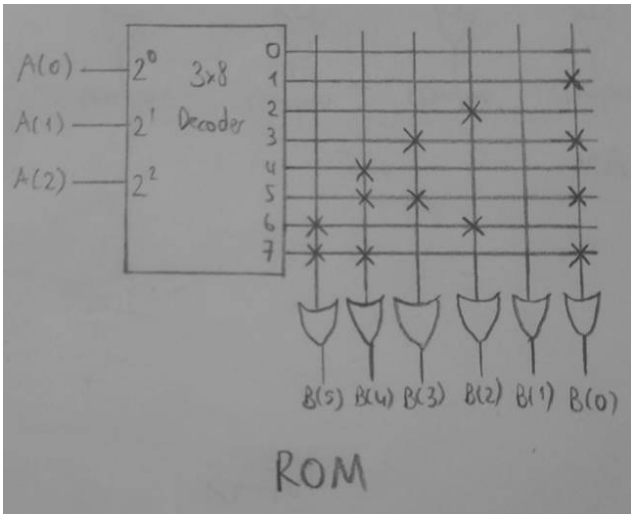
```
    address <= "10"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "00"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "01"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "11"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "10"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "00"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
    address <= "01"; ReadWrite <= '1'; En <= '1'; wait for 10 ns;
end process;
```

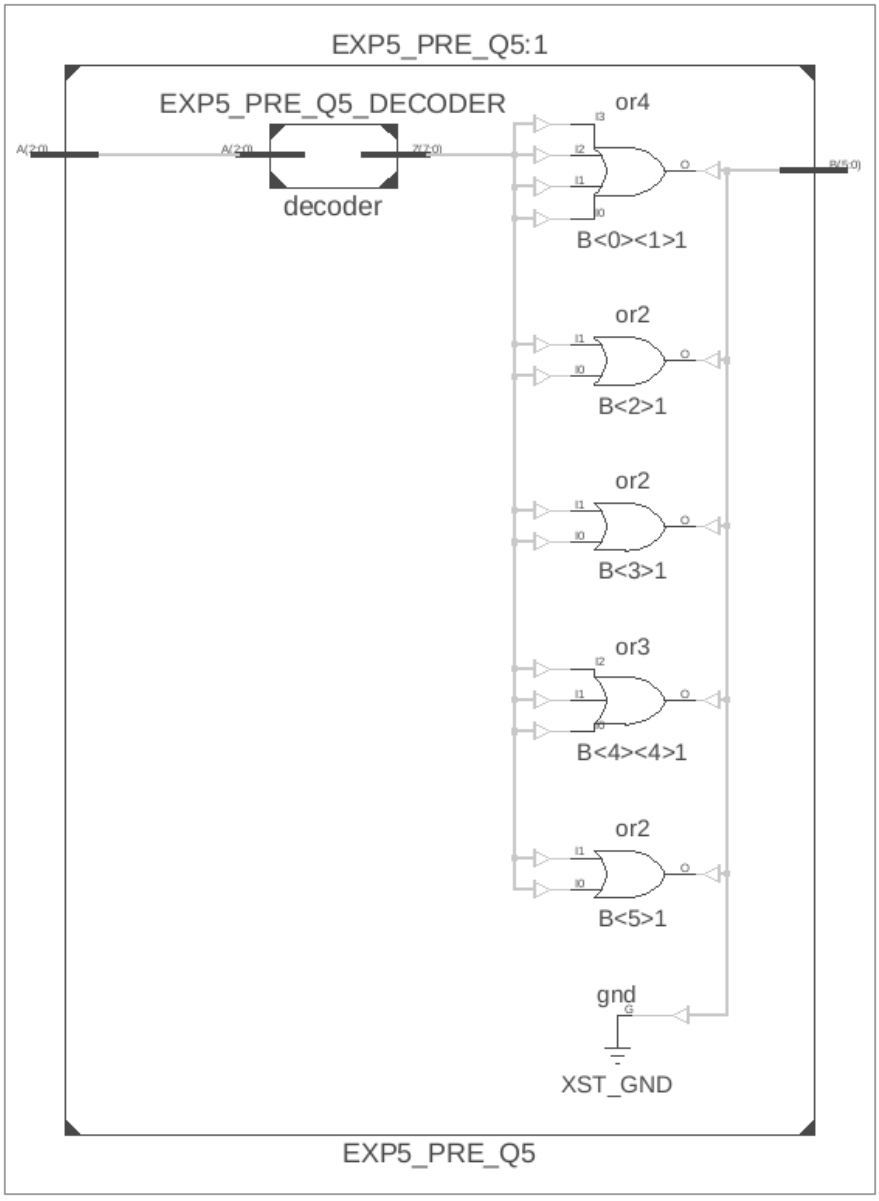


Q5: 1. Analytical Solution

Input A				Output B						
Decimal	Binary			Decimal	Binary					
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

Hand-drawn circuit and RTL schematic





2. Codes

VHDL - 3x8 Decoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q5_DECODER is
    Port ( A : in  STD_LOGIC_VECTOR (2 downto 0);
           Z : out  STD_LOGIC_VECTOR (7 downto 0));
end EXP5_PRE_Q5_DECODER;
architecture Behavioral of EXP5_PRE_Q5_DECODER is
begin
    Z(0) <= not A(2) and not A(1) and not A(0);
    Z(1) <= not A(2) and not A(1) and A(0);
```

```
    Z(2) <= not A(2) and A(1) and not A(0);
    Z(3) <= not A(2) and A(1) and A(0);
    Z(4) <= A(2) and not A(1) and not A(0);
    Z(5) <= A(2) and not A(1) and A(0);
    Z(6) <= A(2) and A(1) and not A(0);
    Z(7) <= A(2) and A(1) and A(0);
end Behavioral;
```

VHDL - ROM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity EXP5_PRE_Q5 is
    port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B : out STD_LOGIC_VECTOR (5 downto 0));
end EXP5_PRE_Q5;
architecture Behavioral of EXP5_PRE_Q5 is
component EXP5_PRE_Q5_DECODER
    Port ( A : in  STD_LOGIC_VECTOR (2 downto 0);
           Z : out  STD_LOGIC_VECTOR (7 downto 0));
end component;
signal d_outp : STD_LOGIC_VECTOR (7 downto 0));
begin
    decoder : EXP5_PRE_Q5_DECODER port map(A, d_outp);
    B(0) = d_outp(1) or d_outp(3) or d_outp(5) or d_outp(7);
    B(1) = '0';
    B(2) = d_outp(2) or d_outp(6);
    B(3) = d_outp(3) or d_outp(5);
    B(4) = d_outp(4) or d_outp(5) or d_outp(7);
    B(5) = d_outp(6) or d_outp(7);
end Behavioral;
```

3. Results

Test bench

```
stim_proc: process
begin
    A <= "000"; wait for 100 ns; A <= "001"; wait for 100 ns;
    A <= "010"; wait for 100 ns; A <= "011"; wait for 100 ns;
    A <= "100"; wait for 100 ns; A <= "101"; wait for 100 ns;
    A <= "110"; wait for 100 ns; A <= "111"; wait for 100 ns;
end process;
```