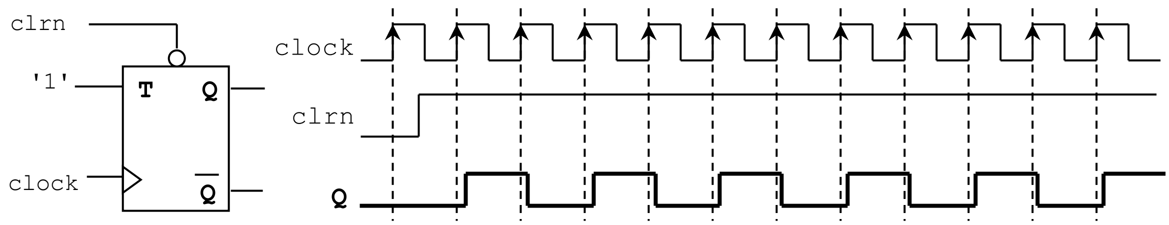
Solutions - Homework 3

(Due date: October 31st @ 9:30 am)
Presentation and clarity are very important!

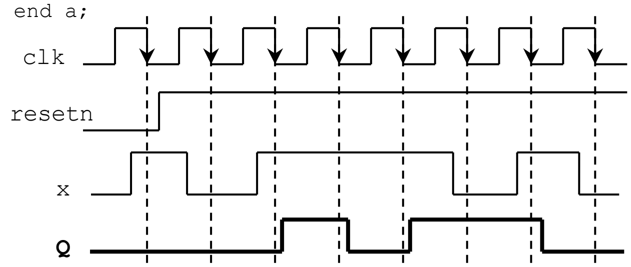
PROBLEM 1 (20 PTS)

- Complete the timing diagram of the circuit shown below. If the frequency of the signal clock is 25 MHz, what is the frequency (in MHz) of the signal Q? (5 pts). ***Frequency of Q***: 25/2 = 12.5 MHz

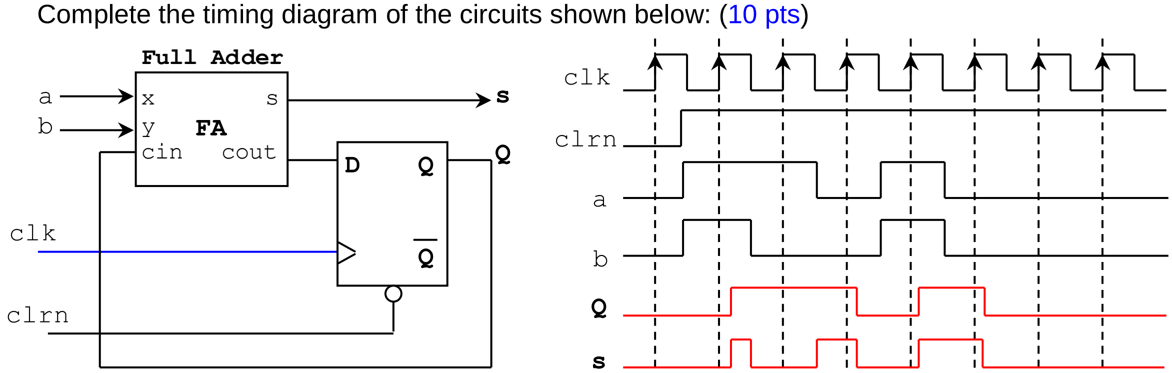


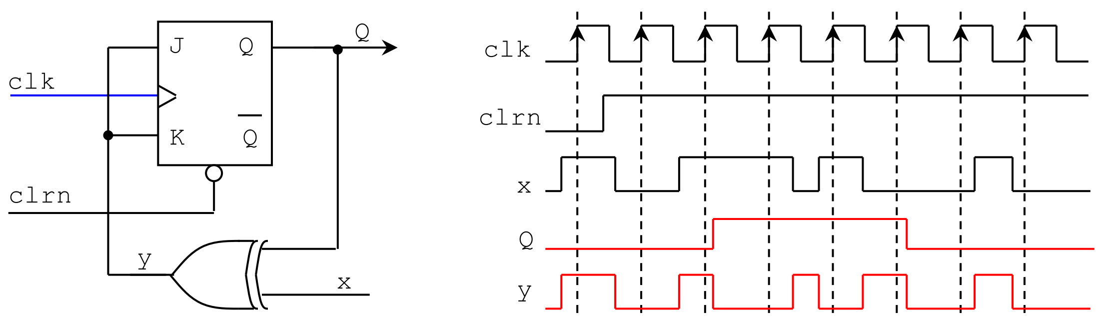
- Complete the timing diagram of the circuit whose VHDL description is shown below: (5 pts)

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

entity circ is
  port ( resetn, x, clk: in std_logic;
         q: out std_logic);
end circ;

architecture a of circ is

   signal qt: std_logic;

begin
   process (resetn, clk, x)
   begin
     if resetn = '0' then
        qt <= '0';
     elsif (clk'event and clk = '0') then
        if x = '1' then
           qt <= not(qt);
        end if;
     end if;
   end process;
   q <= qt;
end a;
```



- Complete the timing diagram of the circuits shown below: (10 pts)



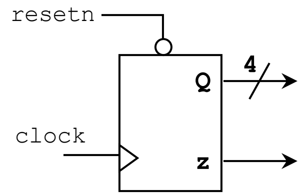


PROBLEM 2 (15 PTS)

Design a BCD counter via a Finite State Machine (FSM):

BCD counter features:

- count: **0000**, 0001, 0010, 0011,…, 1000, 1001, **0000**, ….
- resetn: Asynchronous input signal. It initializes the count to "0000"
- output 'z': It becomes '1' when the count is 1001.

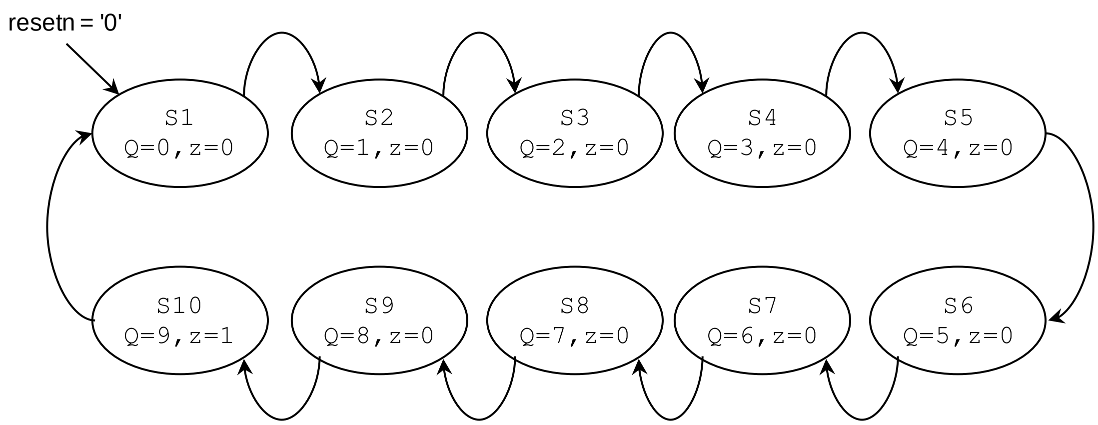


BCD counter

✓ Provide the State Diagram and the Excitation table. Is this a Moore or Mealy machine?

✓ Sketch the circuit (simplify your circuit using K-maps).

Moore-type FSM:



PRESENT STATE	NEXT STATE	Q	z
S1	S2	0000	0
S2	S3	0001	0
S3	S4	0010	0
S4	S5	0011	0
S5	S6	0100	0
S6	S7	0101	0
S7	S8	0110	0
S8	S9	0111	0
S9	S10	1000	0
S10	S1	1001	1

PRESENT STATE $Q_3Q_2Q_1Q_0(t)$	NEXTSTATE $Q_3Q_2Q_1Q_0(t+1)$	z
0 0 0 0	0 0 0 1	0
0 0 0 1	0 0 1 0	0
0 0 1 0	0 0 1 1	0
0 0 1 1	0 1 0 0	0
0 1 0 0	0 1 0 1	0
0 1 0 1	0 1 1 0	0
0 1 1 0	0 1 1 1	0
0 1 1 1	1 0 0 0	0
1 0 0 0	1 0 0 1	0
1 0 0 1	0 0 0 0	1
1 0 1 0	X X X X	X
1 0 1 1	X X X X	X
1 1 0 0	X X X X	X
1 1 0 1	X X X X	X
1 1 1 0	X X X X	X
1 1 1 1	X X X X	X

$Q_3(t+1)$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	X	1
01	0	0	X	0
11	0	1	X	X
10	0	0	X	X

$Q_2(t+1)$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	1	0	X	X
10	0	1	X	X

$Q_1(t+1)$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	X	0
01	1	1	X	0
11	0	0	X	X
10	1	1	X	X

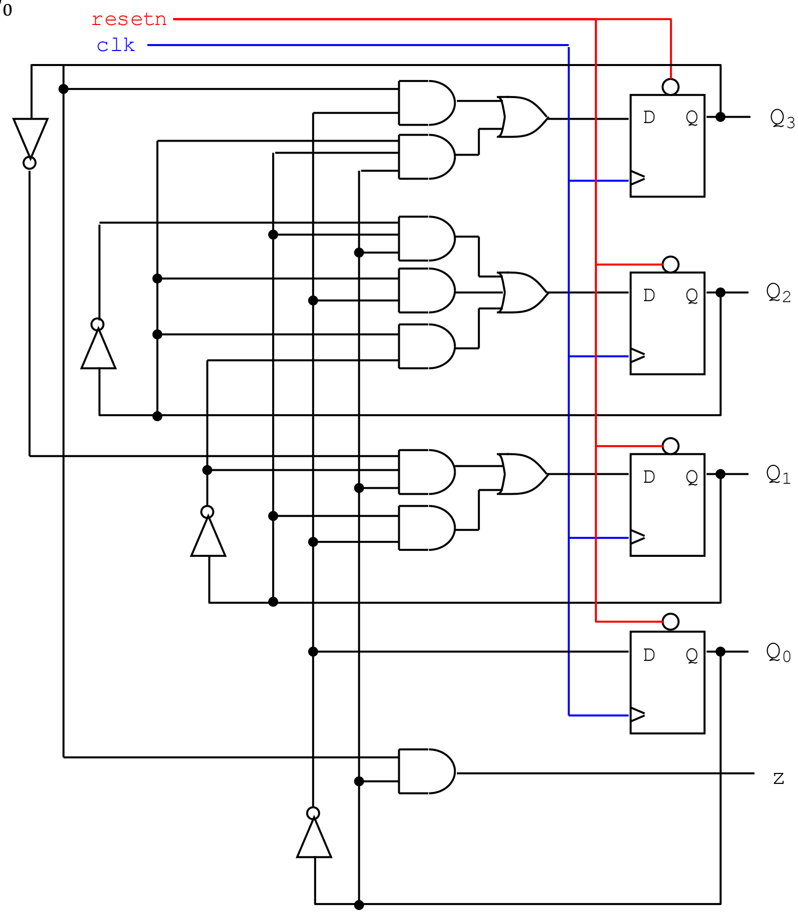
$Q_0(t+1)$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	1	1	X	1
01	0	0	X	0
11	0	0	X	X
10	1	1	X	X

z

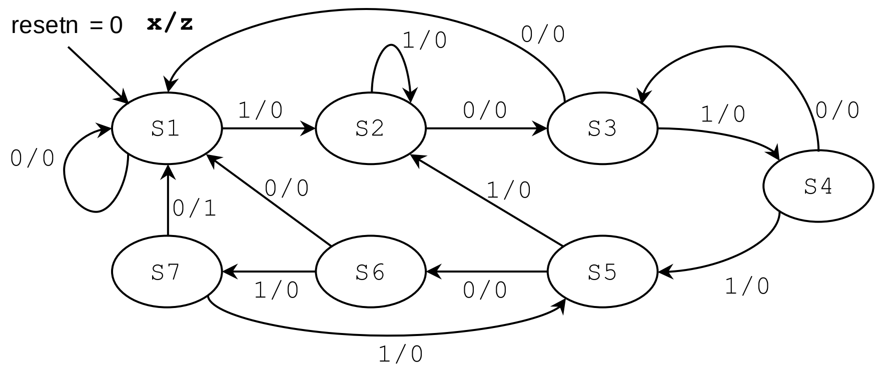
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	X	0
01	0	0	X	1
11	0	0	X	X
10	0	0	X	X

$$Q_3(t+1) = Q_2Q_1Q_0 + Q_3\overline{Q_0}$$
$$Q_2(t+1) = \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_0} + Q_2\overline{Q_1}$$
$$Q_1(t+1) = \overline{Q_1}Q_0\overline{Q_3} + Q_1\overline{Q_0}$$
$$Q_0(t+1) = \overline{Q_0}$$
$$z = Q_3Q_0$$

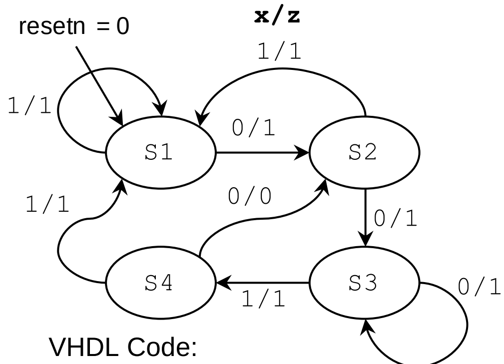


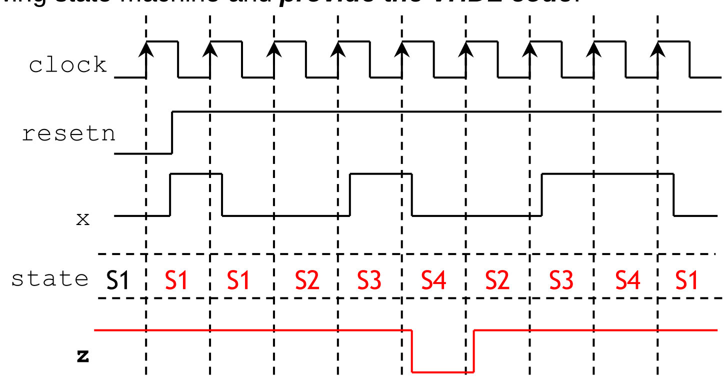
PROBLEM 3 (15 PTS)

- Sequence detector (with overlap): Draw the state diagram of a circuit that detects the following sequence: `1011010`. The detector must assert an output '`z=1`' when the sequence is detected.



- Complete the timing diagram of the following state machine and ***provide the VHDL code***:

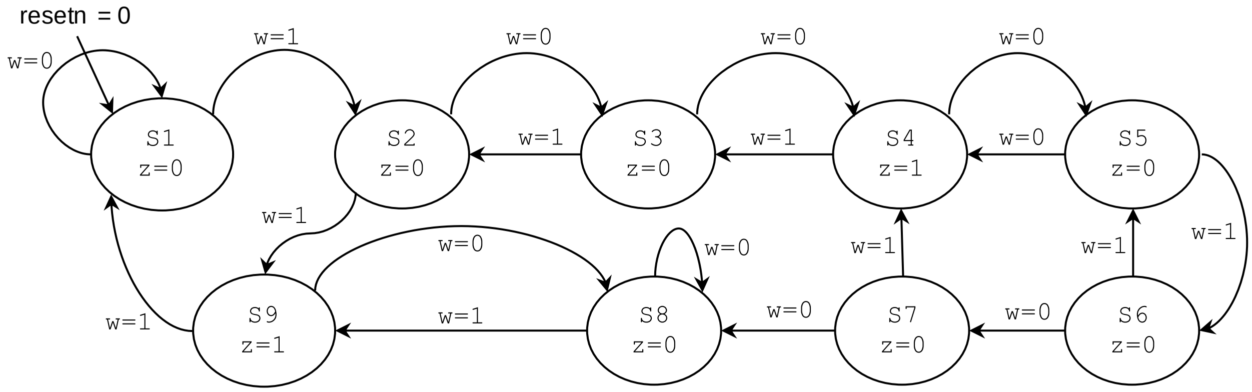


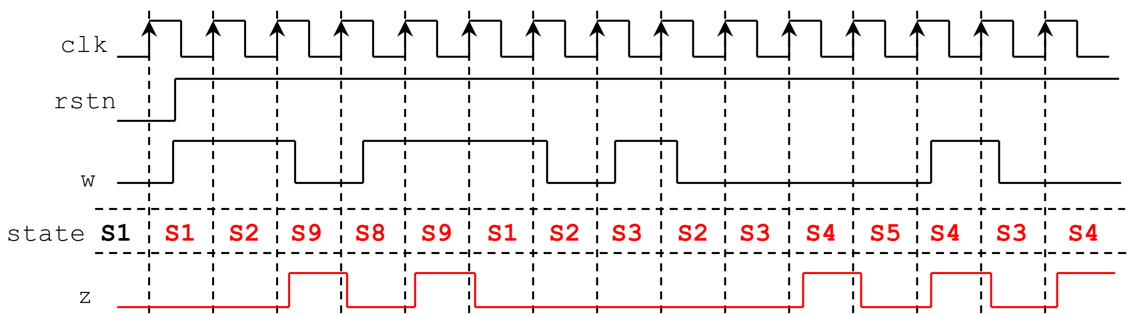


VHDL Code:

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

entity HW3_p3b is
    port (clock, resetn, x: std_logic;
          z: out std_logic);
end HW3_p3b;

architecture Behavioral of HW3_p3b is
    type state is (S1, S2, S3, S4);
    signal y: state;
begin

    Transitions: process (resetn, clock,x)
    begin
            if resetn = '0' then
                    y <= S1;
            elsif (clock'event and clock = '1') then
                    case y is
                            when S1 => if x = '1' then y <= S1; else y <= S2; end if;
                            when S2 => if x = '1' then y <= S1; else y <= S3; end if;
                            when S3 => if x = '1' then y <= S4; else y <= S3; end if;
                            when S4 => if x = '1' then y <= S1; else y <= S2; end if;
                    end case;
            end if;
    end process;

    Outputs: process (x,y)
    begin
            case y is
                    when S1 => z <= '1';
                    when S2 => z <= '1';
                    when S3 => z <= '1';
                    when S4 => if x = '1' then z <= '1'; else z <= '0'; end if;
            end case;
    end process;
end Behavioral;
```

- Complete the timing diagram of the following Moore-type FSM and ***provide the VHDL code***:





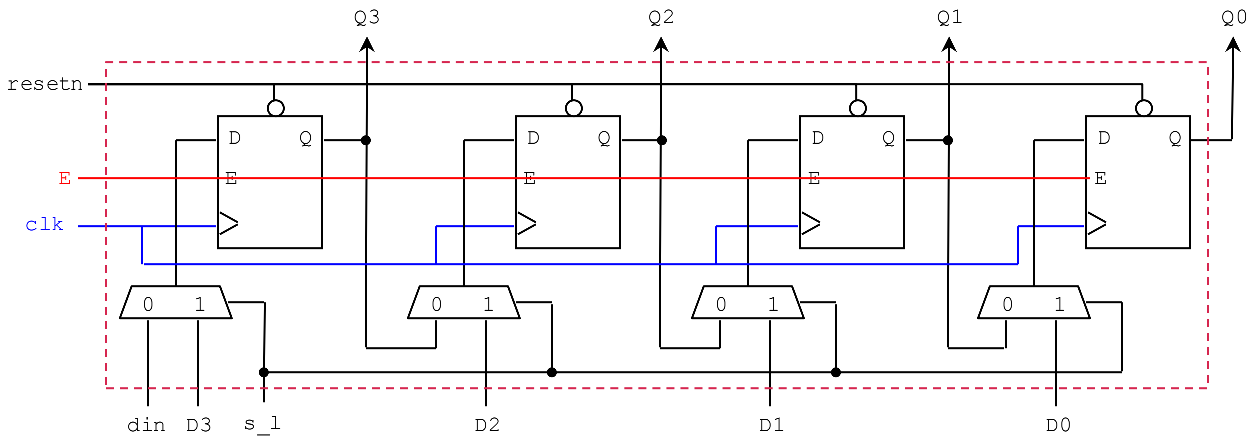
VHDL Code:

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

entity HW3_p3c is
    port (clock, resetn: std_logic;
          w: in std_logic;
          z: out std_logic);
end HW3_p3c;

architecture Behavioral of HW3_p3c is
    type state is (S1, S2, S3, S4, S5, S6, S7, S8, S9);
    signal y: state;
begin

    Transitions: process (resetn, clock,w)
    begin
            if resetn = '0' then
                    y <= S1;
            elsif (clock'event and clock = '1') then
                    case y is
                            when S1 => if w = '1' then y <= S2; else y <= S1; end if;
                            when S2 => if w = '1' then y <= S9; else y <= S3; end if;
                            when S3 => if w = '1' then y <= S2; else y <= S4; end if;
                            when S4 => if w = '1' then y <= S3; else y <= S5; end if;
                            when S5 => if w = '1' then y <= S6; else y <= S4; end if;
                            when S6 => if w = '1' then y <= S5; else y <= S7; end if;
                            when S7 => if w = '1' then y <= S4; else y <= S8; end if;
                            when S8 => if w = '1' then y <= S9; else y <= S8; end if;
                            when S9 => if w = '1' then y <= S1; else y <= S8; end if;
                    end case;
            end if;
    end process;

    Outputs: process (w,y)
    begin
            z <= '0';
            case y is
                    when S1 =>
                    when S2 =>
                    when S3 =>
                    when S4 => z <= '1';
                    when S5 =>
```

```
                    when S6 =>
                    when S7 =>
                    when S8 =>
                    when S9 => z <= '1';
            end case;
    end process;
end Behavioral;
```

PROBLEM 4 (20 PTS)

- Parallel/serial load shift register with enable input. Shifting operation: `s_l=0`. Parallel load: `s_l=1`.
- ✓ Provide the VHDL code of the circuit shown below.
- ✓ Create a VHDL testbench according to the timing diagram shown below. Complete the timing diagram by simulating your circuit. The clock frequency must be 50 MHz.

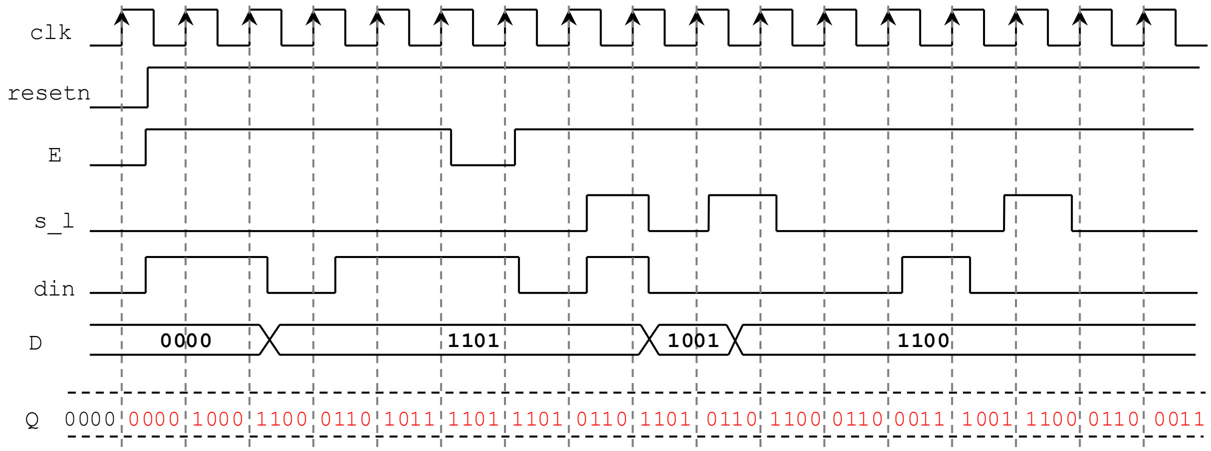


- VHDL Code:

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

entity my_pashiftreg is
   generic (N: INTEGER:= 4);
    port ( clock, resetn: in std_logic;
           din, E, s_l: in std_logic;
           D: in std_logic_vector (N-1 downto 0);
           Q: out std_logic_vector (N-1 downto 0);
          shiftout: out std_logic);
end my_pashiftreg;

architecture Behavioral of my_pashiftreg is
    signal Qt: std_logic_vector (N-1 downto 0);
begin

    process (resetn, clock)
    begin
            if resetn = '0' then
                    Qt <= (others => '0');
            elsif (clock'event and clock = '1') then
                    if E = '1' then
                            if s_l = '1' then
                                    Qt <= D;
                            else
                                    Qt(N-1) <= din;
                                    for i in 0 to N-2 loop
                                            Qt(i) <= Qt(i+1);
                                    end loop;
                            end if;
                    end if;
            end if;
    end process;

    Q <= Qt;
    shiftout <= Qt(0);
end Behavioral;
```

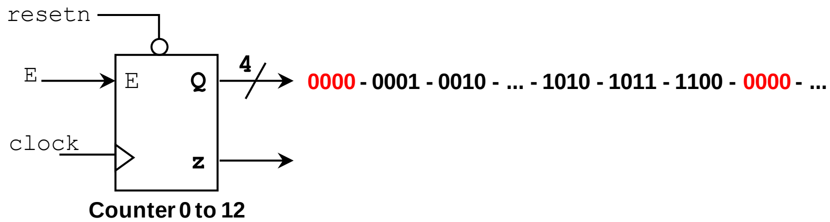
- VHDL Testbench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_my_pashiftreg IS
END tb_my_pashiftreg;

ARCHITECTURE behavior OF tb_my_pashiftreg IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT my_pashiftreg
    PORT(
         clock : IN  std_logic;
         resetn : IN  std_logic;
         din : IN  std_logic;
         E : IN  std_logic;
         s_l : IN  std_logic;
         D : IN  std_logic_vector(3 downto 0);
         Q : OUT  std_logic_vector(3 downto 0);
         shiftout : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal clock : std_logic := '0';
   signal resetn : std_logic := '0';
   signal din : std_logic := '0';
   signal E : std_logic := '0';
   signal s_l : std_logic := '0';
   signal D : std_logic_vector(3 downto 0) := (others => '0');

    --Outputs
   signal Q : std_logic_vector(3 downto 0);
   signal shiftout : std_logic;

   -- Clock period definitions
   constant T : time := 20 ns; -- Frequency: 50 MHz

BEGIN
    -- Instantiate the Unit Under Test (UUT)
   uut: my_pashiftreg PORT MAP (clock => clock, resetn => resetn, din => din, E => E,
                                s_l => s_l, D => D, Q => Q, shiftout => shiftout);

   -- Clock process definitions
   clock_process :process
   begin
          clock <= '0'; wait for T/2;
          clock <= '1'; wait for T/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
          resetn <= '0'; wait for 100 ns;
          resetn <= '0'; wait for T*2;
          resetn <= '1';
          D <= "0000"; din <= '1'; E <= '1'; s_l <= '0'; wait for T*2;
          D <= "1101"; din <= '0'; E <= '1'; s_l <= '0'; wait for T;
          D <= "1101"; din <= '1'; E <= '1'; s_l <= '0'; wait for T*2;
          D <= "1101"; din <= '1'; E <= '0'; s_l <= '0'; wait for T;

          E <= '1';
          D <= "1101"; din <= '0'; s_l <= '0'; wait for T;
          D <= "1101"; din <= '1'; s_l <= '1'; wait for T;
          D <= "1001"; din <= '0'; s_l <= '0'; wait for T;
          D <= "1100";
          din <= '0'; s_l <= '1'; wait for T;
          din <= '0'; s_l <= '0'; wait for T*2;
          din <= '1'; s_l <= '0'; wait for T;
          din <= '0'; s_l <= '1'; wait for T;
          din <= '0'; s_l <= '0'; wait for T*2;
          wait;
   end process;

END;
```



PROBLEM 5 (20 PTS)

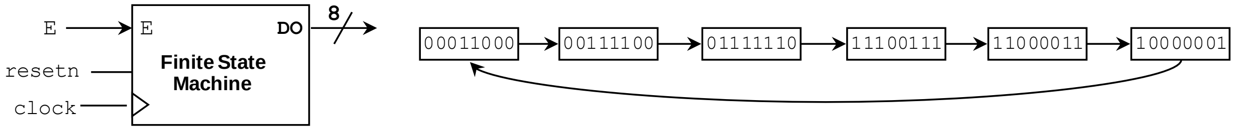
- Provide the VHDL code of a counter (from 0 to 12) with enable (E). If E=0, the count stops. The output 'z' is asserted (z=1) when Q=1100. (5 pts)



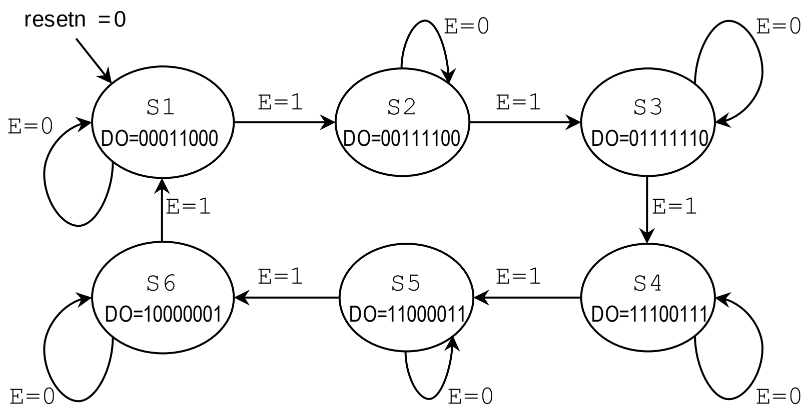
VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

entity my_mod13count is
    port ( clock, resetn, E: in std_logic;
           Q: out std_logic_vector (3 downto 0);
           z: out std_logic);
end my_mod13count;

architecture Behavioral of my_mod13count is
    signal Qt: integer range 0 to 12;
begin

    process (resetn, clock, E)
    begin
        if resetn = '0' then
            Qt <= 0;
        elsif (clock'event and clock = '1') then
            if E = '1' then
                if Qt = 12 then
                    Qt <= 0;
                else
                    Qt <= Qt + 1;
                end if;
            end if;
        end if;
    end process;
    Q <= conv_std_logic_vector(Qt,4);

    z <= '1' when Qt = 12 else '0';

end Behavioral;
```

- Provide the state diagram and the VHDL code of a Moore-type Finite State Machine that generates the following sequence: (15 pts)



  - **Important:** Include an enable input in your state machine. When E=1, state transitions do occur. When E=0 there are no state transitions.
  - **Tip**: the output bits don't have to be the outputs of the flip flops. You can create this state machine with only 6 states, thereby requiring only 3 flip flops.



VHDL code:

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

entity my_ledseq is
    port (clock, resetn: std_logic;
          E: in std_logic;
                   DO: out std_logic_vector (7 downto 0));
end my_ledseq;

architecture Behavioral of my_ledseq is
    type state is (S1, S2, S3, S4, S5, S6);
    signal y: state;
begin

    Transitions: process (resetn, clock, E)
    begin
        if resetn = '0' then
            y <= S1;
        elsif (clock'event and clock = '1') then
            case y is
                when S1 => if E = '1' then y <= S2; else y <= S1; end if;
                when S2 => if E = '1' then y <= S3; else y <= S2; end if;
                when S3 => if E = '1' then y <= S4; else y <= S3; end if;
                when S4 => if E = '1' then y <= S5; else y <= S4; end if;
                when S5 => if E = '1' then y <= S6; else y <= S5; end if;
                when S6 => if E = '1' then y <= S1; else y <= S6; end if;
            end case;
        end if;
    end process;

    Outputs: process (y)
    begin
        case y is
            when S1 => DO <= "00011000";
            when S2 => DO <= "00111100";
            when S3 => DO <= "01111110";
            when S4 => DO <= "11100111";
            when S5 => DO <= "11000011";
            when S6 => DO <= "10000001";
        end case;
    end process;

end Behavioral;
```

PROBLEM 6 (10 PTS)

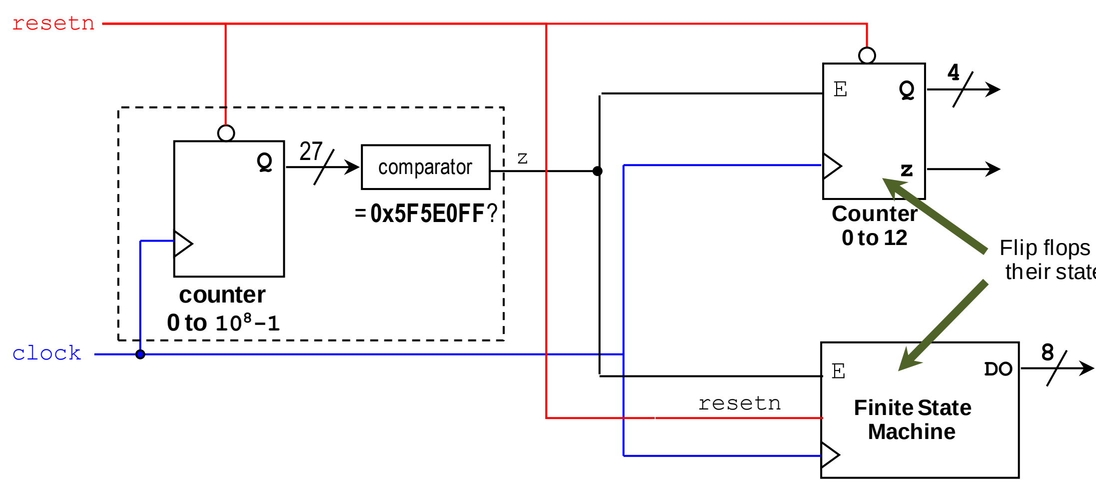
- We want to connect the output bits of the circuits in Problem 5 to LEDs in the NEXYS3 Board. And we want to see the output transitions before our eyes. In the NEXYS3 Board, the input clock frequency is 100 MHz (Period: 10 ns), making it impossible for our eyes to perceive the transitions.

- If we want the output bits to change every ***1 second*** (for example) a straightforward solution is to modify the clock frequency to 1 Hz. But this can be a hard problem if a precise input clock is required.

- **Alternative solution**: We create a circuit that generates a one-period (10 ns) pulse every 1 second. This output is then connected to the enable input of every flip flop, counter, and register whose rate of operation we would like to modify. This way, we get the same effect as modifying the clock frequency to 1 Hz. And we get to use the 100 MHz clock for all the flip flops. The figure below depicts this circuit (black dotted box).

  - We need to count 10^8 clock cycles to get to 1 second (100 MHz amounts to a period of 10 ns). When the count becomes $10^8 - 1$, the circuit generates a pulse (z=1). The counter requires $\lceil \log_2 10^8 \rceil = 27$ bits, and the comparison has to be with $10^8 - 1 = 0x5F5E0FF$.

  - In VHDL, the 'z' output signal can be generated internally in the counter description, or we can attach an external comparator to the output of the counter.



✓ Provide the VHDL code of the circuit that generates a 10 ns pulse every ***500 milliseconds*** (this circuit allows the circuits of Problem 5 to operate every 500 milliseconds).

VHDL code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee.math_real.log2;
use ieee.math_real.ceil;

entity my_genpulse is
       generic (COUNT: INTEGER:= (10**8)/2); -- (10**8)/2 cycles of T = 10 ns --> 0.5 s
       port (clock, resetn: in std_logic;
             z: out std_logic);
end my_genpulse;

architecture Behavioral of my_genpulse is
       constant nbits: INTEGER:= integer(ceil(log2(real(COUNT))));
       signal Qt: std_logic_vector (nbits -1 downto 0);
```

```
begin

	process (resetn, clock)
	begin
		if resetn = '0' then
			Qt <= (others => '0');
		elsif (clock'event and clock = '1') then
			if Qt = conv_std_logic_vector (COUNT-1,nbits) then
				Qt <= (others => '0');
			else
				Qt <= Qt + conv_std_logic_vector (1,nbits);
			end if;
		end if;
	end process;

	z <= '1' when Qt = conv_std_logic_vector (COUNT-1,nbits) else '0';

end Behavioral;
```

EXTRA CREDIT (+15 PTS)

- Implement the circuits of Problem 5 in VHDL, with the output bits changing every 500 milliseconds.
- Demonstrate the circuits working on the NEXYS3 Board.