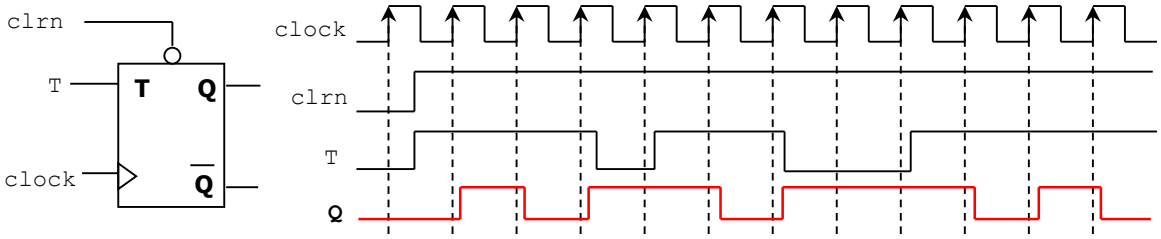
Solutions - Homework 3

(Due date: March 12th @ 5:30 pm)

Presentation and clarity are very important! Show your procedure!

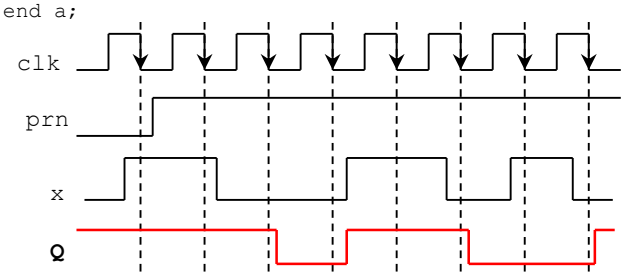
PROBLEM 1 (25 PTS)

a) Complete the timing diagram of the circuit shown below. (5 pts)

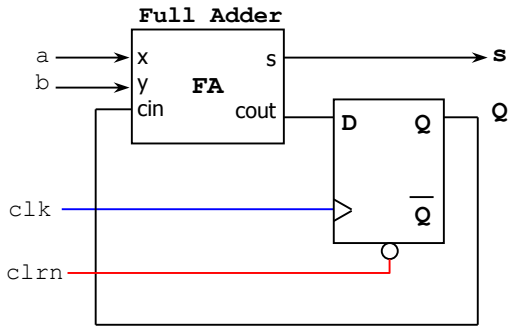


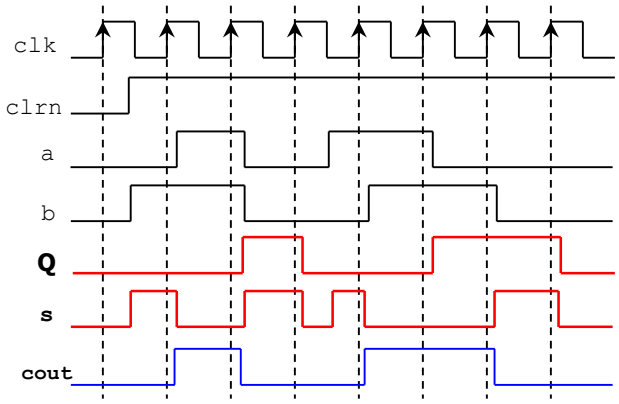
b) Complete the timing diagram of the circuit whose VHDL description is shown below: (5 pts)

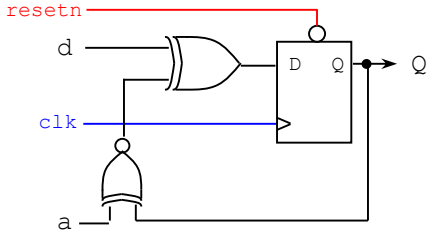
```
library ieee;
use ieee.std_logic_1164.all;

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

architecture a of circ is

   signal qt: std_logic;

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

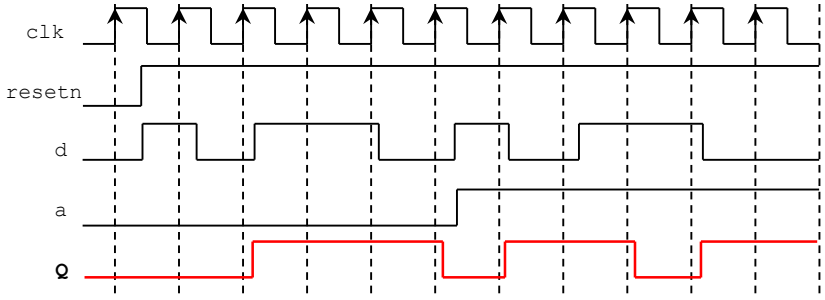


c) Complete the timing diagram of the circuits shown below: (15 pts)



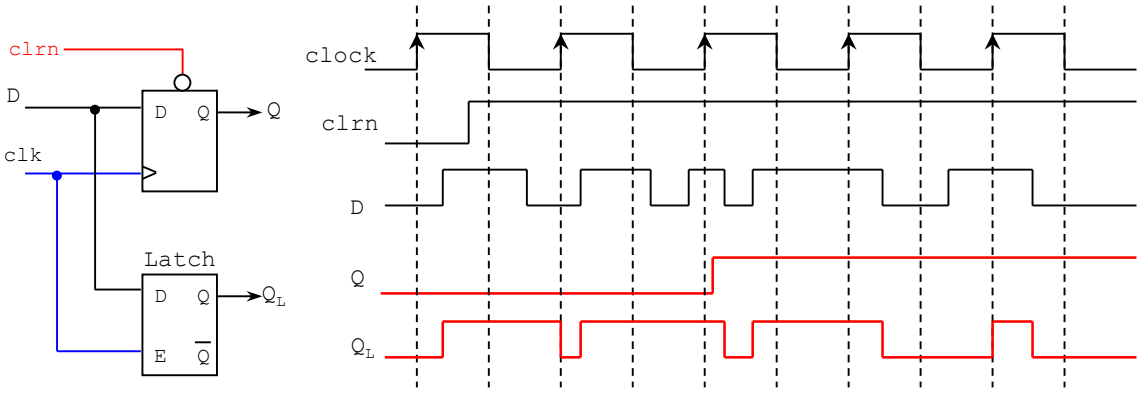






PROBLEM 2 (15 PTS)

- Complete the timing diagram of the circuit shown below: (8 pts)



- Complete the VHDL description of the synchronous sequential circuit whose truth table is shown below: (7 pts)

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

entity my_ff is
  port ( a, b, c: in std_logic;
         clrn, clk: in std_logic;
         q: out std_logic);
end my_ff;

architecture a of my_ff is
  signal qt: std_logic;
begin
  process (clrn, clk, a, b)
  begin
    if clrn = '0' then qt <= '0';
    elsif (clk'event and clk='1') then
      if (a = '0' and b = '0') then
         qt <= c;
      elsif (a = '1' and b = '0') then
         qt <= '1';
      elsif (a = '1' and b = '1') then
         qt <= not (qt);
      end if;
   end process;
   q <= qt;
end a;
```

clrn	clk	A	B	Q_{t+1}
1	↑	0	0	C
1	↑	0	1	Q_t
1	↑	1	0	1
1	↑	1	1	$\overline{Q_t}$
0	X	X	X	0

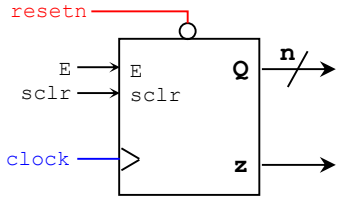
PROBLEM 3 (15 PTS)

- Design a modulo-200 counter with enable, synchronous clear, and synchronous load.

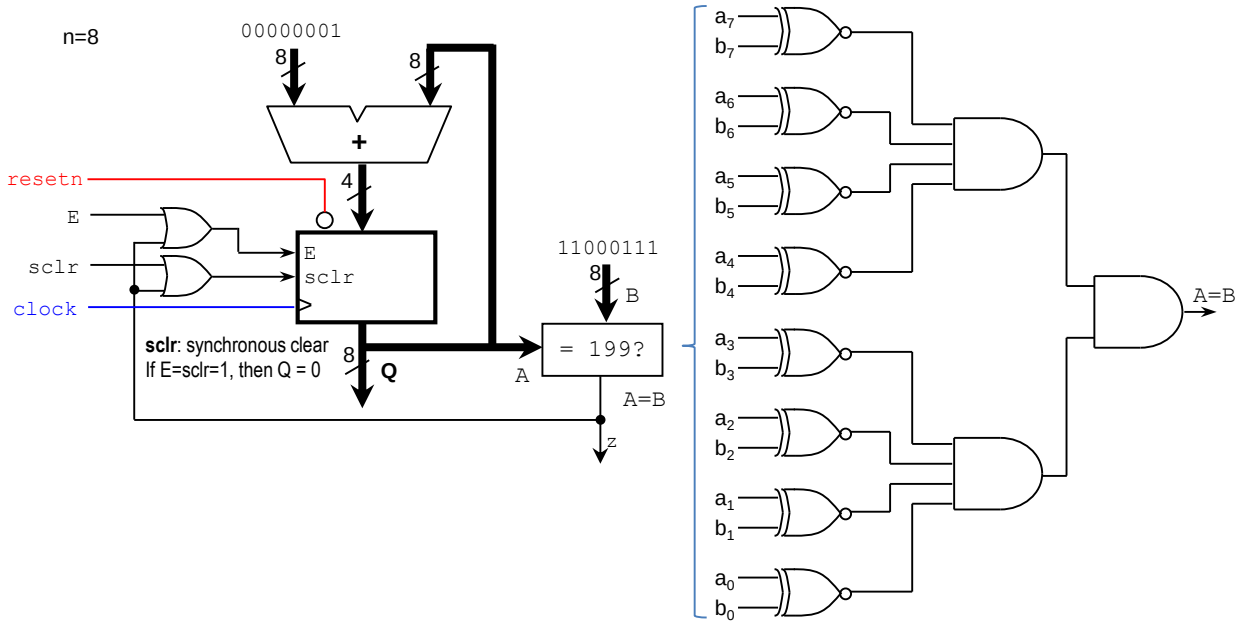
- Asynchronous input: `resetn`
- Synchronous inputs:
  - ✓ `E`: This is the enable input. It increases the count every time it is asserted (`E = 1`).
  - ✓ `sclr`: It clears the count (it requires `E = 1`)

- Outputs:
  - ✓ `Q`: This is the count.
  - ✓ `z`: It is asserted only when the maximum count is reached.

- You need to determine the minimum number of bits n that it is required for the count.
- You can use adder units, registers, logic gates, and MUXes.

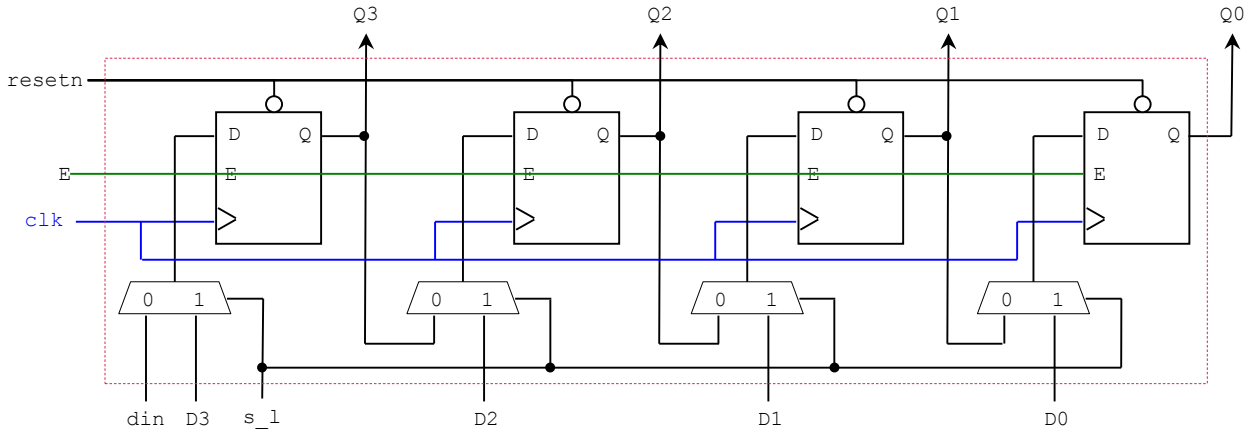


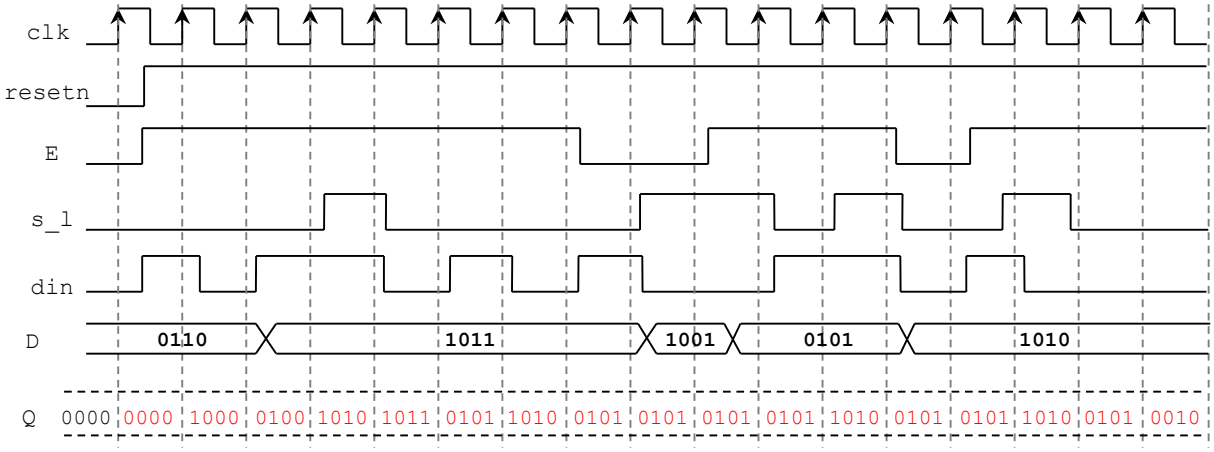
counter modulo-200



PROBLEM 4 (35 PTS)

- The following circuit is a parallel/serial load shift register with enable input. Shifting operation: `s_l=0`. Parallel load: `s_l=1`.
  - ✓ Write a structural VHDL code. You MUST create a file for: i) flip flop, ii) MUX 2-to-1, and iii) top file (where you will interconnect the flip flops and MUXes). Provide a printout. (15 pts)
  - ✓ Write a VHDL testbench according to the timing diagram shown below. Complete the timing diagram by simulating your circuit (Timing Simulation). The clock frequency must be 100 MHz with 50% duty cycle. Provide a printout. (20 pts)





✓ **VHDL Code**: Top File

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

entity my_st_shiftreg is
     generic (N: INTEGER:= 4;
              DIR: STRING:= "RIGHT"); -- RIGHT/LEFT
     port (D: in std_logic_vector (N-1 downto 0);
           resetn, clock, din, E_, s_l: in std_logic;
           Q: out std_logic_vector (N-1 downto 0));
end my_st_shiftreg;

architecture structure of my_st_shiftreg is
  component dffe
     port ( d : in  STD_LOGIC;
            clrn, prn, clk, ena: in std_logic;
            q : out  STD_LOGIC);
  end component;

  component mux2to1
     port (a,b, sel : in std_logic;
           y : out std_logic);
  end component;

  signal ds, md, Qt: std_logic_vector (N-1 downto 0);
begin

a0: assert (DIR = "LEFT" or DIR = "RIGHT")
    report "DIR can only be LEFT or RIGHT"
    severity error;

rr: if DIR = "RIGHT" generate
        ds(N-1) <= din;
        ds(N-2 downto 0) <= Qt(N-1 downto 1);
    end generate;

rl: if DIR = "LEFT" generate
        ds(0) <= din;
        ds(N-1 downto 1) <= Qt(N-2 downto 0);
    end generate;

ti: for i in N-1 downto 0 generate
        fi: dffe port map (d => md(i), clrn => resetn, prn =>'1', clk => clock, ena => E, q => Qt(i));
        mi: mux2to1 port map (a => ds(i), b => D(i), sel => s_l, y => md(i));
    end generate;

    Q <= Qt;
end structure;
```

✓ **VHDL Code**: D-Type flip flop

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

entity dffe is
    port ( d : in  STD_LOGIC;
           clrn, prn, clk, ena: in std_logic;
           q : out  STD_LOGIC);
end dffe;

architecture behaviour of dffe is

begin
     process (clk, ena, prn, clrn)
     begin
        if clrn = '0' then q <= '0';
        elsif prn = '0' then q <= '1';
        elsif (clk'event and clk='1') then
          if ena = '1' then q <= d; end if;
        end if;
     end process;
end behaviour;
```

✓ **VHDL Code**: MUX 2-to-1

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

entity mux2to1 is
     port (a,b, sel: in std_logic;
           y : out std_logic);
end mux2to1;

architecture structure of mux2to1 is

begin

     with sel select
              y <= a when '0',
                   b when others;

end structure;
```

✓ **VHDL Tesbench**:

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

ENTITY tb_my_st_shiftreg IS
     generic (N: INTEGER:= 4);
END tb_my_st_shiftreg;

ARCHITECTURE behavior OF tb_my_st_shiftreg IS
    COMPONENT my_st_shiftreg
    PORT(
         D : IN  std_logic_vector(N-1 downto 0);
         resetn, clock, din : IN  std_logic;
         E, s_l : IN  std_logic;
         Q : OUT  std_logic_vector(N-1 downto 0));
    END COMPONENT;

   --Inputs
   signal D : std_logic_vector(N-1 downto 0) := (others => '0');
   signal resetn : std_logic := '0';
   signal clock : std_logic := '0';
   signal din : std_logic := '0';
   signal E : std_logic := '0';
   signal s_l : std_logic := '0';

    --Outputs
   signal Q : std_logic_vector(N-1 downto 0);

   -- Clock period definitions
   constant T : time := 10 ns;

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

   -- 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
     D <= "0110"; resetn <= '0'; wait for 100 ns;
     resetn <= '0'; wait for T*2;
     resetn <= '1';
     D <= "0110"; din <= '1'; E <= '1'; s_l <= '0'; wait for T;
     D <= "0110"; din <= '0'; E <= '1'; s_l <= '0'; wait for T;
     D <= "1011"; din <= '1'; E <= '1'; s_l <= '0'; wait for T;
     D <= "1011"; din <= '1'; E <= '1'; s_l <= '1'; wait for T;
     D <= "1011"; din <= '0'; E <= '1'; s_l <= '0'; wait for T;
     D <= "1011"; din <= '1'; E <= '1'; s_l <= '0'; wait for T;
     D <= "1011"; din <= '0'; E <= '1'; s_l <= '0'; wait for T;
     D <= "1011"; din <= '1'; E <= '0'; s_l <= '0'; wait for T;
     D <= "1001"; din <= '0'; E <= '0'; s_l <= '1'; wait for T;
     D <= "0101"; din <= '0'; E <= '1'; s_l <= '1'; wait for T;
     D <= "0101"; din <= '1'; E <= '1'; s_l <= '0'; wait for T;
     D <= "0101"; din <= '1'; E <= '1'; s_l <= '1'; wait for T;
     D <= "1010"; din <= '0'; E <= '0'; s_l <= '0'; wait for T;
     D <= "1010"; din <= '1'; E <= '1'; s_l <= '1'; wait for T;
     D <= "1010"; din <= '0'; E <= '1'; s_l <= '0'; wait for T;
     D <= "1010"; din <= '0'; E <= '1'; s_l <= '0'; wait for T;
     wait;
   end process;
END;
```

PROBLEM 5 (10 PTS)

- Attach a printout of your Initial Project Report (no more than a page). This report should contain the project title, the project description, and the current status of the project. Use the provided template (`Final Project – Report Template.docx`).