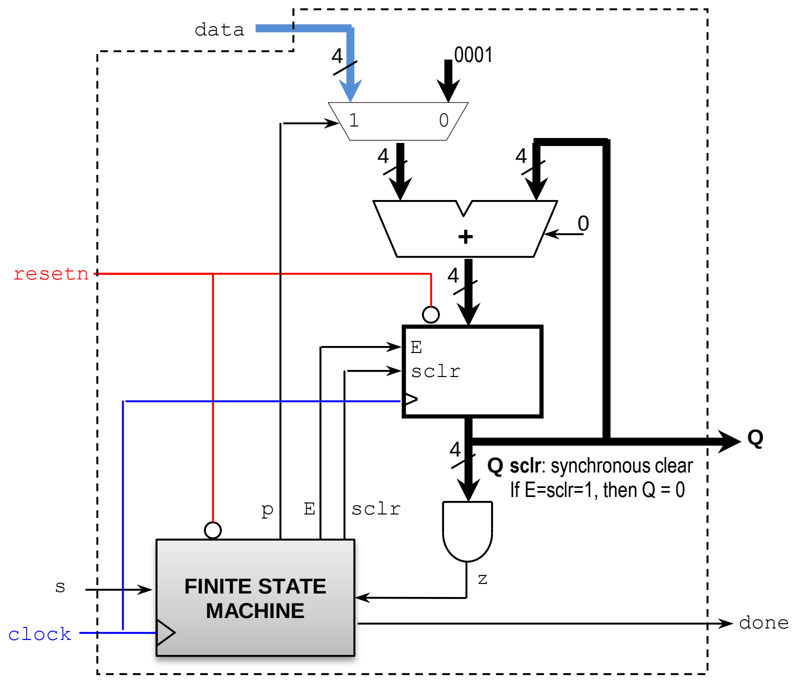
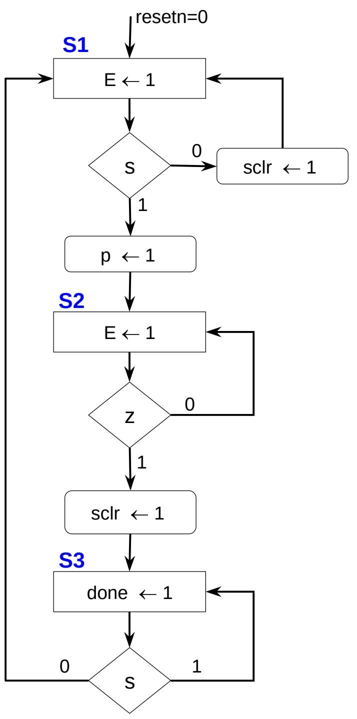
Solutions - Homework 1

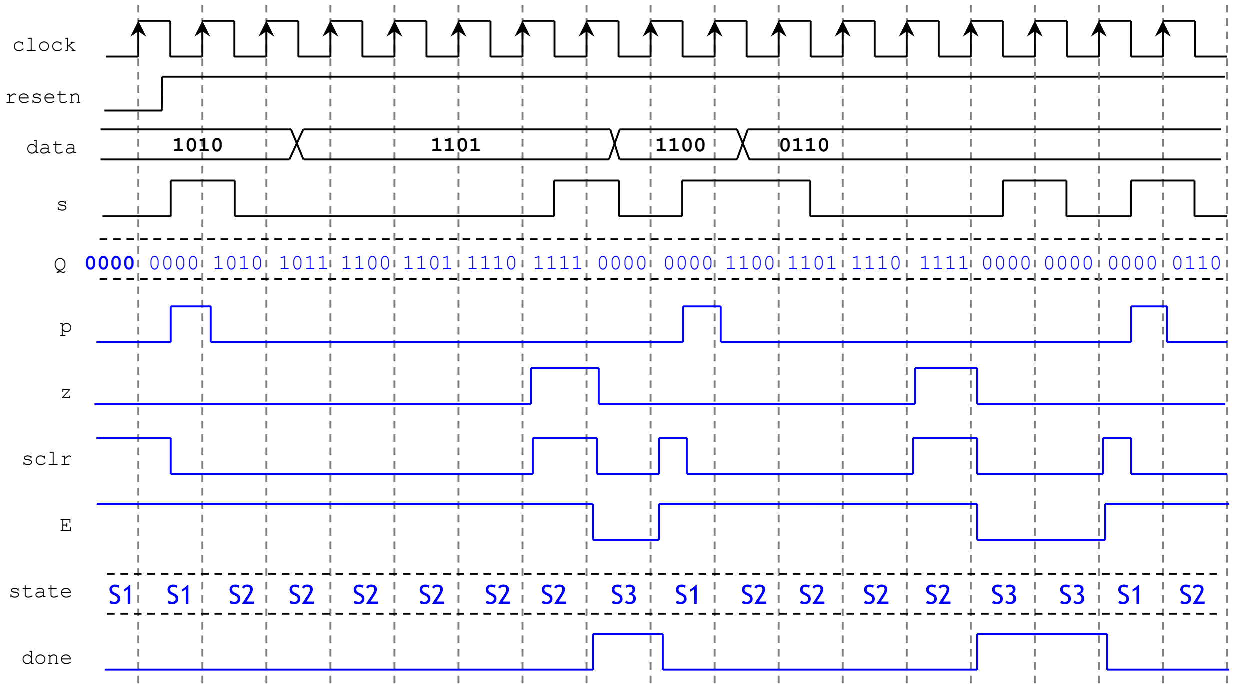
(Due date: January 18th @ 7:30 pm)
Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (50 PTS)

- Given the following digital circuit that includes an FSM (in ASM form) and a datapath circuit:
  Note: Register. If E=1, sclr=0 → Parallel Load. If E=sclr=1 → Synchronous clear.
  - ✓ Complete the timing diagram of the digital circuit.
  - ✓ Write a structural VHDL code. You MUST create a file for i) 4-bit register, ii) 4-bit adder, iii) 4-bit Bus MUX 2-to-1, iv) Finite State Machine, and vi) Top file (where you will interconnect all the components). Provide a printout.
  - ✓ Write a testbench according to the timing diagram shown below. Simulate the circuit (Behavioral simulation). Verify that the simulation is correct by comparing it with the timing diagram you completed manually. Provide a testbench printout.







✓ **VHDL Code**: Top File

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

entity top is
     port ( resetn, clock, s: in std_logic;
             data: in std_logic_vector (3 downto 0);
             Q: out std_logic_vector (3 downto 0);
             done: out std_logic);
end top;

architecture structure of top is

    component my_fsm
        port (clock, resetn: std_logic;
              s, z: in std_logic;
              p, E, sclr, done: out std_logic);
    end component;

     component my_addsub -- download from tutorial
        generic (N: INTEGER:= 4);
        port(    addsub   : in std_logic;
               x, y     : in std_logic_vector (N-1 downto 0);
               s        : out std_logic_vector (N-1 downto 0);
               overflow : out std_logic;
               cout     : out std_logic);
    end component;

    component my_busmux2to1
       generic (N: INTEGER:= 8); -- Length of each input signal
        port (a,b: in std_logic_vector (N-1 downto 0);
              s: in std_logic;
              y: out std_logic_vector (N-1 downto 0));
    end component;

    component my_rege -- download from tutorial
       generic (N: INTEGER:= 4);
        port ( clock, resetn: in std_logic;
               E, sclr: in std_logic; -- sclr: Synchronous clear
                 D: in std_logic_vector (N-1 downto 0);
               Q: out std_logic_vector (N-1 downto 0));
    end component;

    signal p, E, sclr, z: std_logic;
    signal Qt, Xa, SA: std_logic_vector (3 downto 0);

begin

ga: my_addsub generic map (N => 4)
    port map (addsub => '0', x => XA, y => Qt, s => SA);

gm: my_busmux2to1 generic map (N => 4)
    port map (a => "0001", b => data, s => p, y => XA);

gr: my_rege generic map (N=> 4)
    port map (clock => clock, resetn => resetn, E => E, sclr => sclr, D => SA, Q => Qt);
    z <= Qt(3) and Qt(2) and Qt(1) and Qt(0);  Q <= Qt;

fs: my_fsm port map (clock=>clock, resetn=>resetn, s=>s, z=>z, p=>p, E=>E, sclr=>sclr, done=>done);

end structure;
```

✓ **VHDL Code**: FSM

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

entity my_fsm is
     port (clock, resetn: std_logic;
           s, z: in std_logic;
           p, E, sclr, done: out std_logic);
end my_fsm;

architecture behaviour of my_fsm is
     type state is (S1, S2, S3);
     signal y: state;
```

```
   begin

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

                                when S2 =>
                                    if z = '1' then y <= S3; else y <= S2; end if;

                                when S3 =>
                                    if s = '1' then y <= S3; else y <= S1; end if;
                       end case;
                end if;

        end process;

        Outputs: process (y, s, z)
        begin
            E <= '0'; p <= '0'; sclr <= '0'; done <='0'; -- Default values
                case y is
                        when S1 =>
                             E <= '1'; if s = '1' then p <= '1'; else sclr <= '1'; end if;

                        when S2 =>
                             E <= '1'; if z = '1' then sclr <= '1'; end if;

                        when S3 =>
                             done <= '1';

                end case;
        end process;

end behaviour;
```

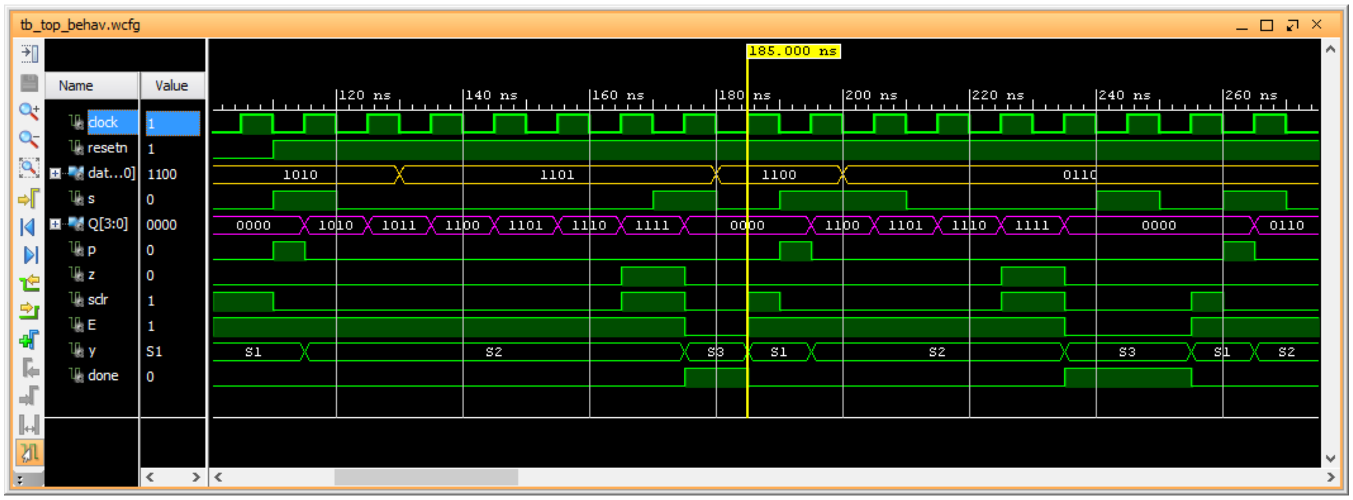
✓ **VHDL Code**: BUS MUX 2-to-1

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

   entity my_busmux2to1 is
      generic (N: INTEGER:= 8); -- Length of each input signal
        port (a,b: in std_logic_vector (N-1 downto 0);
              s: in std_logic;
              y: out std_logic_vector (N-1 downto 0));
   end my_busmux2to1;

   architecture structure of my_busmux2to1 is

   begin
        with s select
                y <= a when '0',
                     b when others;
   end structure;
```

✓ **VHDL Tesbench**:

```
   LIBRARY ieee;
   USE ieee.std_logic_1164.ALL;
   use ieee.math_real.log2;
   use ieee.math_real.ceil;

   ENTITY tb_top IS
   END tb_top;

   ARCHITECTURE behavior OF tb_top IS
       component top -- Component Declaration for the Unit Under Test (UUT)
           port ( resetn, clock, s: in std_logic;
                  data: in std_logic_vector (3 downto 0);
                  Q: out std_logic_vector (3 downto 0);
                  done: out std_logic);
       end component;
```

```
    -- Inputs
    signal clock, resetn : std_logic := '0';
    signal s : std_logic := '0';
    signal data: std_logic_vector(3 downto 0):= (others => '0');

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

    -- Clock period definitions
    constant T: time := 10 ns;
    constant DUTY_CYCLE: real:= 0.5;
    constant OFFSET: time:= 100 ns;

BEGIN

     -- Instantiate the Unit Under Test (UUT)
    uut: top PORT MAP (resetn, clock, s, data, Q, done);

    -- Clock process definitions
    clock_process :process
    begin
             wait for OFFSET;
             clock_loop: loop
                            clock <= '0'; wait for (T - (T*DUTY_CYCLE));
                            clock <= '1'; wait for (T*DUTY_CYCLE);
             end loop clock_loop;
    end process;

    -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 100 ns.
        resetn <= '0'; wait for OFFSET;
        resetn <= '0';
        data <= "1010"; s <= '0'; wait for T;
        resetn <= '1';
        data <= "1010"; s <= '1'; wait for T;
        data <= "1010"; s <= '0'; wait for T;
        data <= "1101"; s <= '0'; wait for 4*T;
        data <= "1101"; s <= '1'; wait for T;
        data <= "1100"; s <= '0'; wait for T;
        data <= "1100"; s <= '1'; wait for T;
        data <= "0110"; s <= '1'; wait for T;
        data <= "0110"; s <= '0'; wait for 3*T;
        data <= "0110"; s <= '1'; wait for T;
        data <= "0110"; s <= '0'; wait for T;
        data <= "0110"; s <= '1'; wait for T;
        data <= "0110"; s <= '0';
        wait;
    end process;

END;
```

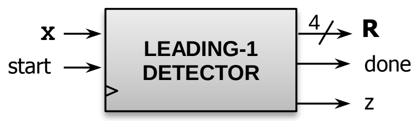


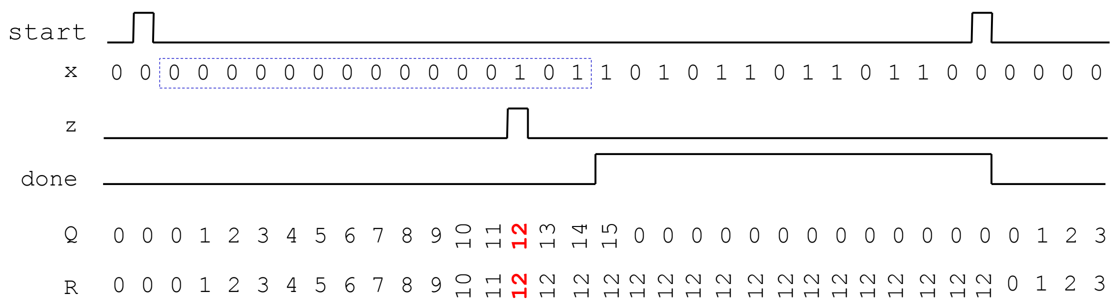
PROBLEM 2 (30 PTS)

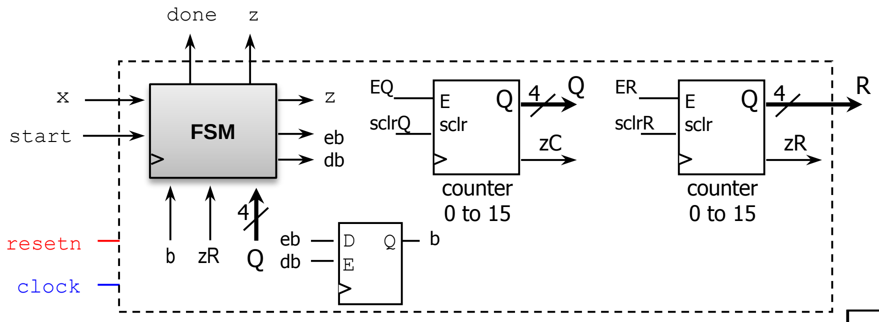
- Design a serial leading-1 detector (datapath + FSM): A binary number of 15 bits is presented to the serial leading-1 detector, most significant bit first, on input X. The circuit generates the number of 0's that exist before the first 1. For example: If the number is 000000000000101, then the output should be 12. If the number is 000000000000000, then the output should be 15.
- Circuit inputs: X (serial data), start. Outputs: z (leading 1 detected), done (15 bits processed), and R (number of 0's before the first 1.
- The process begins with the assertion of the *start* signal (a clock pulse). For every bit, we increase a count Q (0 to 15). As long as the bits applied to X are 0, the output Z = 0, and we also increase a count R (0 to 15). When the first 1 is applied to X, then Z = 1 and we freeze the count R. For all bit values applied to X after the first 1 is applied, Z = 0. When the sequence is completed, we issue done='1'. To process another sequence, the input *start* becomes 1 for a clock cycle (here, done becomes 0). The counter R keeps the result until the *start* input is becomes 1 again.

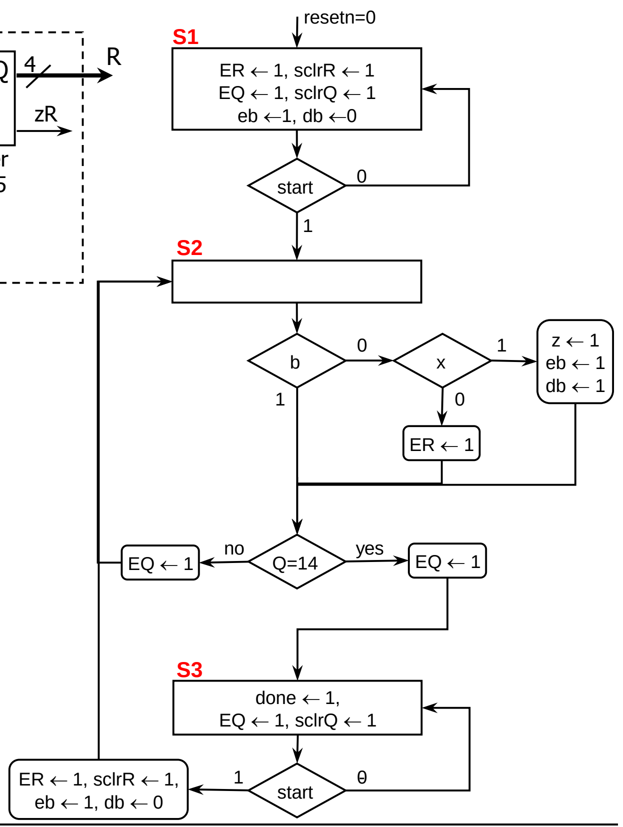
✓ Sketch the circuit: FSM + Datapath. Specify all the I/Os of the FSM, as well as the signals connecting the FSM and the Datapath.
<u>Suggestion</u>: The Datapath only needs two counters modulo-16 (Q and R). You can use the standard counter with *enable* and *sclr* inputs.

✓ Provide the State Diagram (in ASM form) of the FSM.









PROBLEM 3 (20 PTS)

- Calculate the result of the following operations, where the operands are signed integers. For the division, calculate both the quotient and the residue. **No procedure ≡ zero points.**

011001 × 11001	11001 × 1001	101001 ÷ 10001	0110101 ÷ 10101	10100 ÷ 0111

```
011001 x                          11001 x
 11001                             1001
   ↓                                 ↓
011001 x  →      1 1 0 0 1 x      00111 x  →        1 1 1 x
 00111                 1 1 1       0111             1 1 1
                 ---------                        -------
                 1 1 0 0 1                          1 1 1
               1 1 0 0 1                          1 1 1
             1 1 0 0 1                          1 1 1
             ---------------                  -----------
             1 0 1 0 1 1 1 1                  1 1 0 0 0 1
                    ↓                               ↓
           0 1 0 1 0 1 1 1 1                0 1 1 0 0 0 1
                    ↓
           1 0 1 0 1 0 0 0 1
```

✓ $\frac{101001}{10001} = \frac{-23}{-15}$

```
        00001
1111 ) 10111
        1111
        ----
        1000
```

To unsigned: $\frac{010111}{01111}$

Unsigned Integer Division: $Q' = 1, R' = 1000$
$\rightarrow Q = Q' = 01, \rightarrow R = -R' = 2C(01000) = 1000$

Verification: $-23 = (-15 \times 1) - 8$

✓ $\frac{0110101}{10101} = \frac{53}{-11}$

```
        000100
1011 ) 110101
       1011↓↓
       ----
          1001
```

To unsigned: $\frac{0110101}{01011}$

Unsigned Integer Division: $Q' = 100, R' = 1001$
$\rightarrow Q = -Q' = 2C(0100) = 100, \rightarrow R = R' = 01001$

Verification: $53 = (-11 \times -4) + 9$

✓ $\frac{10100}{0111} = \frac{-12}{7}$

```
       0001
111 ) 1100
       111
      ----
       101
```

To unsigned: $\frac{01100}{0111}$

Unsigned Integer Division: $Q' = 1, R' = 101$
$\rightarrow Q = -Q' = 2C(01) = 1, \rightarrow R = -R' = 2C(0101) = 1011$

Verification: $-12 = (7 \times -1) - 5$