Digital Systems
Mid-semester examination. April 28, 2021

Time limit: 90 minutes.

1 Problem: A Finite State Machine (50%)

The graph in *Figure* 1 describes a synchronous (clock-driven) FSM.

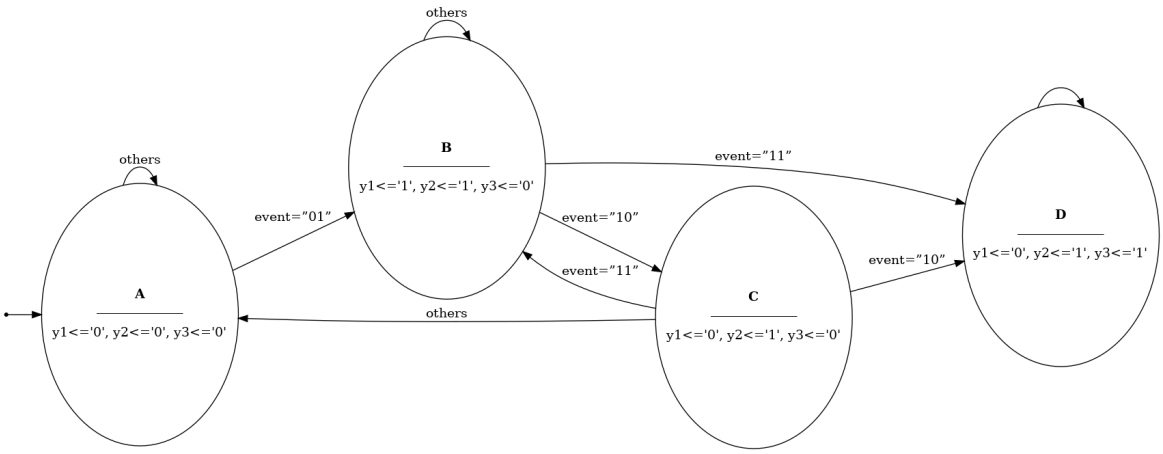

Figure 1: Graph of a synchronous (clock-driven) FSM.

a) Describe the previous FSM in `VHDL`. Help yourself to the following `VHDL` sketch. Use one `process` statement to describe the sequential part, and one or two more `process` statements, as you like, to describe the combinational part.

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

entity fsm is
  port( clk      : in std_logic;
        event    : in std_logic_vector(1 downto 0);
        y1,y2,y3 : out std_logic);
end;

architecture arch of fsm is
  type multiplier_state is (A, B, C, D);
  signal current_state : multiplier_state := A;
  signal next_state : multiplier_state;
begin
  process(clk) is
  begin
  ...
  end process;

  ...

end;
```

b) Complete the waveform of *Figure* 2.

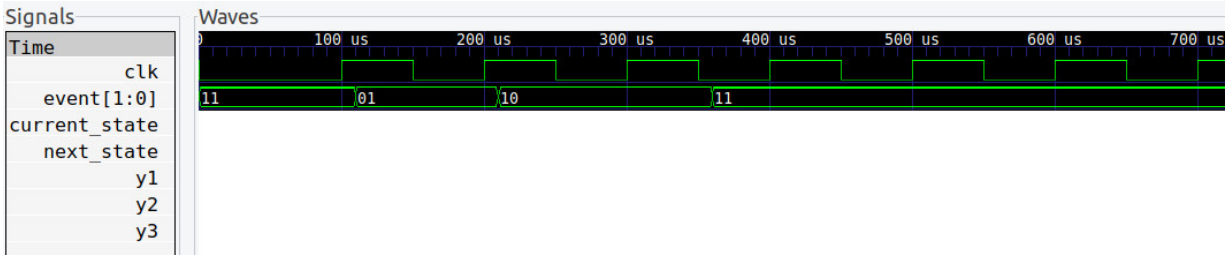

Figure 2: Clock and event input of the FSM.

2 Problem: Pipelining of FIR filters (50 %)

In this problem we are going to consider the so called pipelinig tecnique applied to FIR digital filters (it doesn't matter if now you don't know what a FIR filter is). *Figure* 3 shows the block diagram of a FIR filter (of memory two) in which the output y can be written as a combination of the current input x_0 and the two previous values of this input, x_1 and x_2: $y = a*x_0+b*x_1+c*x_2$. It is worth noting that each *clock* period a new output is computed.

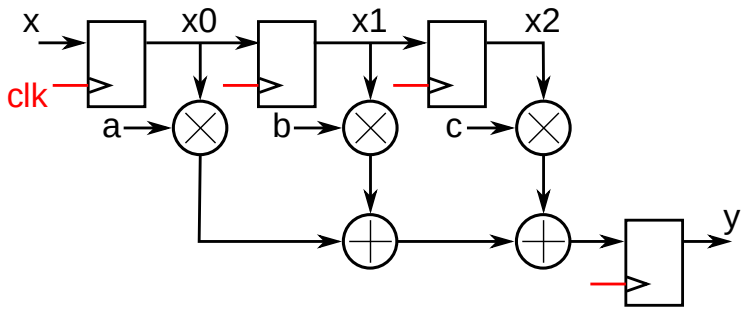

Figure 3: Block diagram of a FIR filter of memory two.

The `VHDL` description of this block diagram is listed below. Note that different signals have different lengths.

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

entity fir is
  port( clk : in std_logic;
        x   : in std_logic_vector(7 downto 0);
        y   : out std_logic_vector(7 downto 0));
end;

architecture arch of fir is
  signal x0,x1,x2 : signed(7 downto 0):=(others=>'0');
  constant a : signed(3 downto 0):=to_signed(1,4);
  constant b : signed(3 downto 0):=to_signed(-2,4);
  constant c : signed(3 downto 0):=to_signed(3,4);
  signal prod_a_x0,prod_b_x1,prod_c_x2,sum_1,sum :
    signed(a'length+x0'length-1 downto 0);
```

```
begin
  process(clk) is
  begin
    if rising_edge(clk) then
      x0<=signed(x);
      x1<=x0;
      x2<=x1;
      y<=std_logic_vector(resize(sum,y'length));
    end if;
  end process;

  prod_a_x0<=a*x0;
  prod_b_x1<=b*x1;
  prod_c_x2<=c*x2;
  sum_1<=prod_a_x0+prod_b_x1;
  sum<=sum_1+prod_c_x2;

end;
```

a) Identify the critical path of *Figure* 3. Compute the maximum clock frequency f_{clk} at which the system can work. Consider the following delays:

- Clock to output delay, t_{co}=8ns.
- Set-up time, t_{su}=2ns.
- Multiplier delay, t_{mult}=85ns.
- Adder delay, t_{add}=45ns.

b) The clock frequency f_{clk} is fixed to 7MHz by the design constraints. Draw the block diagram and write the `VHDL` code that synthesize a new filter using the pipelining tecnique (i.e. shortening the critical path adding some registers) in order to work at f_{clk} = 7MHz. Compare the latency of this design with the previous one.

c) Consider the new design (same delays as before): Identify the critical path. Compute the maximum clock frequency f_{clk} at which the system can work. If this frequency is lower than 10MHz propose another design.