Lab#2

– Digital Electronics –

Sequential Circuits

Group: Wojciech Paluszkiewicz

Martin Callegarin Demangeat

Nicolas Masson

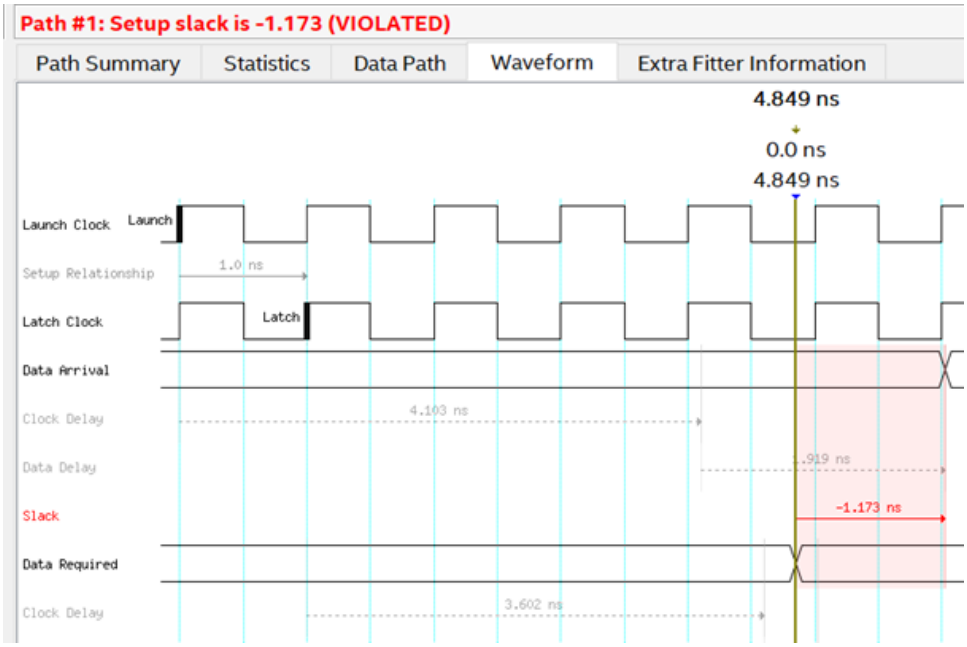
Muhammad Arshad

Unlike the former lab, we will use only sequential process, that means that our circuits use previous input, output, clock and a memory element.

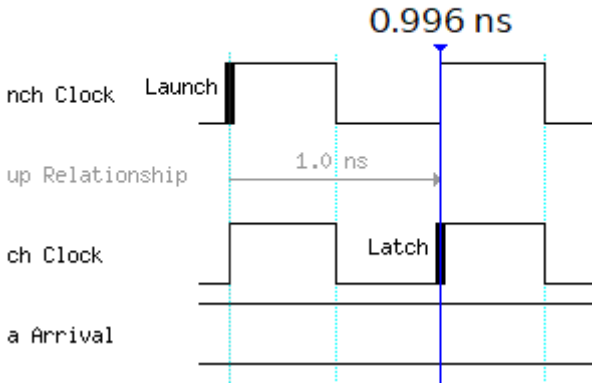
I. 16-bit synchronous counter

To implement a 16-bit synchronous counter we need only one Logic Element.

Timing



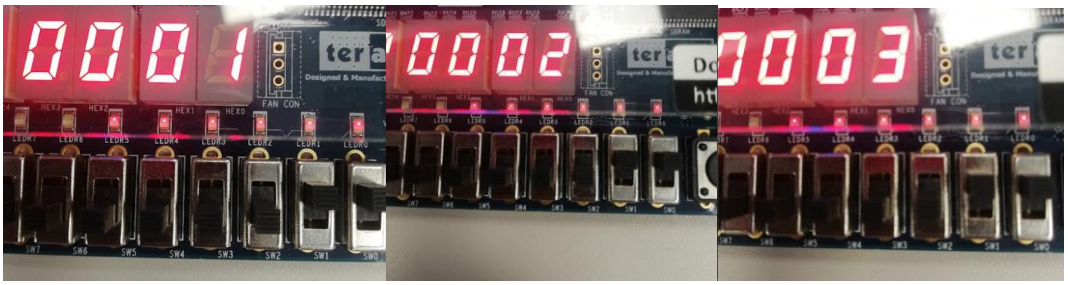
When the slack value is represented in red that means it's negative and the timing results fail to meet the required constraint. So we can calculate the worst-case delay path : 1-(-1.173) = 2.173 ns and it corresponds to fmax=460.193Mhz.

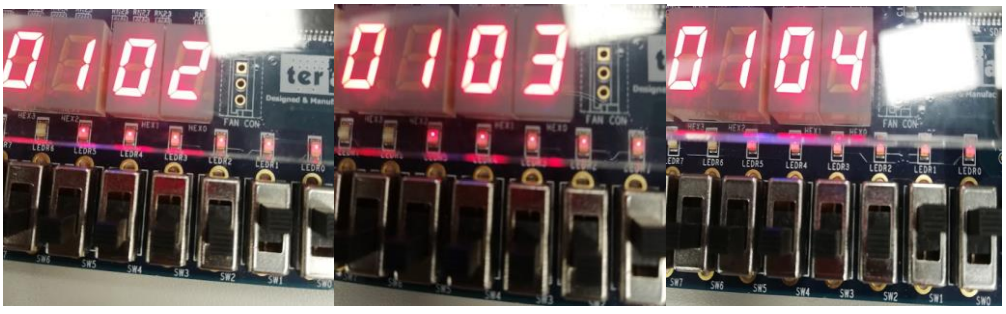


```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity lab2_1 is
port (
    KEY : in std_logic_vector(1 downto 0);--will be our clock
    SW : in std_logic_vector(1 downto 0);--will be the enable and the reset
    HEX0, HEX1, HEX2, HEX3 : out std_logic_vector(0 to 6);--let's try and use the to command this time
    Q : out unsigned(15 downto 0));
end lab2_1;

architecture beh of lab2_1 is

COMPONENT decoder7 --this time it will convert a 4bits to hexa display
PORT (  C : IN STD_LOGIC_VECTOR(3 downto 0);
        Display : OUT STD_LOGIC_VECTOR(0 to 6));
END COMPONENT;

--signals? yes, because we need the value of Q to add to Q,...
--but it is an output signal and we can't read it without a signal
--so;
--signal sum_out : unsigned(15 downto 0) := (others=>'0');--it will then be separated in 4 pieces
signal sum_out : unsigned(15 downto 0) := (others=>'0');
signal stdsum_out : std_logic_vector(15 downto 0);

begin
    process(KEY,SW)
    begin   --process
        if (SW(0) = '0') then
            Q <= (others=> '0');--checkin' the asynchronous reset

        else if (KEY(0) = '1' and KEY(0)' event) then--positiv edge

            if (SW(1) = '1') then --letting the +1 only if enable is on
                sum_out <= sum_out + 1;
                Q <= sum_out;
            end if;--enable
          end if;--clock
        end if;--resetn
    end process;

--the spliting of the sum_out signal
H0: decoder7 port map (std_logic_vector(sum_out(3 downto 0)), HEX0);--probleme here because of type
H1: decoder7 port map (std_logic_vector(sum_out(7 downto 4)), HEX1);--must get unsigned to std_logic
H2: decoder7 port map (std_logic_vector(sum_out(11 downto 8)), HEX2);--because sum_out is unsigned
H3: decoder7 port map (std_logic_vector(sum_out(15 downto 12)), HEX3);--and decoder 7 takes logic_vector

end beh;
```

Decoder:

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

entity decoder7int is
port(
        C : in integer range 0 to 15;
        display : out std_logic_vector (0 to 6));
end decoder7int;

architecture beh of decoder7int is

signal cbits : std_logic_vector(3 downto 0);

begin

cbits <= std_logic_vector(to_unsigned(c,4));

process(cbits)
    begin
    case Cbits is
    when "0000"=>DISPLAY<="0000001";--0
    when "0001"=>DISPLAY<="1001111";--1
    when "0010"=>DISPLAY<="0010010";--2
    when "0011"=>DISPLAY<="0000110";--3
    when "0100"=>DISPLAY<="1001100";--4
    when "0101"=>DISPLAY<="0100100";--5
    when "0110"=>DISPLAY<="0100000";--6
    when "0111"=>DISPLAY<="0001111";--7
    when "1000"=>DISPLAY<="0000000";--8
    when "1001"=>DISPLAY<="0000100";--9
    when "1010"=>DISPLAY<="0001000";--A
    when "1011"=>DISPLAY<="1100000";--B
    when "1100"=>DISPLAY<="0110001";--C
    when "1101"=>DISPLAY<="1000010";--D
    when "1110"=>DISPLAY<="0110000";--E
    when others=>DISPLAY<="0111000";--F
    end case;
end process;
end beh;
```

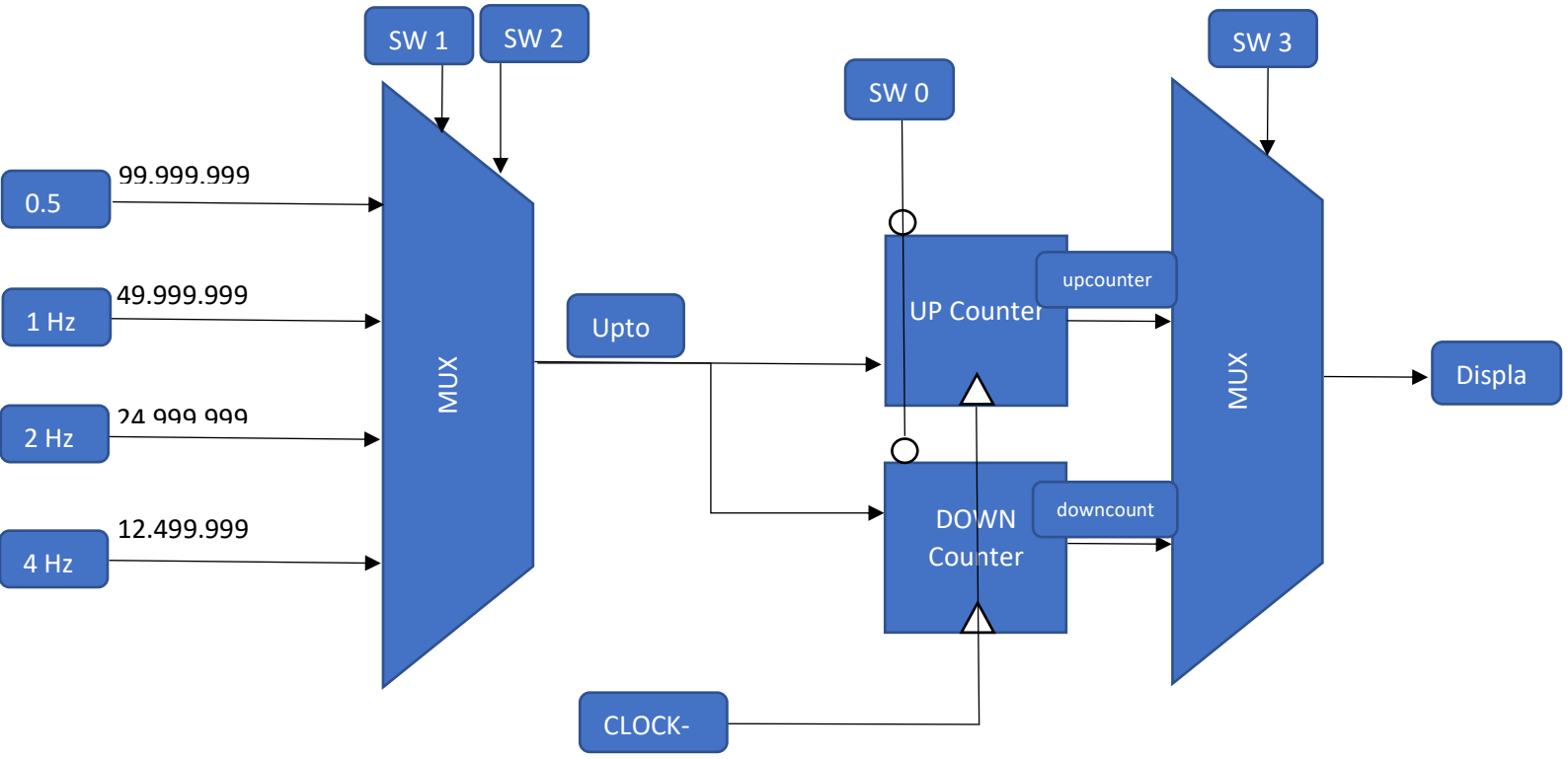




Reset, SW(0) set to ‘0’:

II. Programmable flashing digits

For this exercise we have to display a counter on one 7-segment display HEX0. We have 5 inputs including 4 switches : SW0 acts as the asynchronous reset, SW1 and SW2 define the flashing time of each number and SW3 defines the counting order (up-count or down-count). Like every synchronous sequential circuit, we use a reference clock provided by the board. It corresponds to our fifth input and it is called CLOCK-50 and every signal and flip-flops are directly connected to this internal clock.



Clock operation description :

CLOCK-50 is a 50 MHz clock signal which means that every 50 000 000 ticks, one second just passed, in other words we have a tick every 20ns. The problem is, when SW1 and SW2 are both at 0, the flashing interval is 2s. That is why we have created one signal called *counter50* which goes from 0 to

99999999 so it can reach 2s and one signal called *upto* which contains the output of our first MUX, being the maximum number of tick necessary in order to have the flash frequency required.

Process description :

Then we created a process clocked by CLOCK_50 and we directly began by checking the condition of the reset managed by SW0 and putting our three counters to their initial value. If the condition is not filled, we are counting in parallel with our downcounter and upcounter. Finally, it's the SW3 that will define which one we are going to use to display on HEX0 using also a "case".

We ended by coding the decoder necessary to translate a number by his 7-segment corresponding code. We use the one used for exercise 1.

Main code :

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity lab2_2 is
port (  CLOCK_50 : in std_logic;
        SW : in std_logic_vector(3 downto 0);
        HEX0 : out std_logic_vector(0 to 6));
end lab2_2;

architecture behavioral of lab2_2 is
signal downcounter : integer range 0 to 9 := 9;
signal upcounter,order  : integer range 0 to 9 := 0;

signal counter50 : integer range 0 to 99999999 := 0; --fmax is 50MHz and corresponds to the frequency of the clock, every time it reaches 50MHz, one second just passed
signal upto :integer range 0 to 99999999;--is gonna be changed according to sw2,3

component decoder7int
port(C : in integer range 0 to 9;
     display : out std_logic_vector (0 to 6));
end component;

begin

l1:decoder7int port map(c=>order,
                        display=>HEX0);

frequencychoose : process(clock_50,SW)

begin
case sw(2 downto 1) is
when "00"=>upto<=99999999;
when "01"=>upto<=49999999;
when "10"=>upto<=24999999;
when others=>upto<=12499999;
end case;
end process;

clock_counter: process (CLOCK_50, SW(0))

begin
if sw(0) = '0' then
    upcounter<=0;
    downcounter<=9;
```

```
clock_counter: process (CLOCK_50, SW(0))

begin
if sw(0) = '0' then
    upcounter<=0;
    downcounter<=9;
    counter50<=0;
else
            if CLOCK_50'event and CLOCK_50='1' then--checking clock
            counter50 <= counter50 + 1;
                        if counter50 = upto then -- it means that two second just passed
                        counter50 <= 0;--And we just set it to 0 so that he can recount at infinite

                                            if upcounter < 9 then
                                                upcounter <=upcounter+1;-- for each second we increment the digital counter on HEX0 so that we have a counter set like a real clock
                                            else
                                                upcounter <= 0;
                                            end if;

                        end if;--end of the upcounterchecking
            end if;--end of the counter50 checking
                        if CLOCK_50'event and CLOCK_50='1' then
                        counter50 <= counter50 + 1;
                                            if counter50 = upto then -- it means that two second just passed
                                            counter50 <= 0;--And we just set it to 0 so that he can recount at infinite
                                                                if downcounter > 0 then
                                                                    downcounter <= downcounter-1;-- for each second we increment the digital counter on HEX0 so that we have a counter
                                                                else
                                                                    downcounter<= 9;
                                                                end if;

                                            end if;--downcount checking
                        end if;--coutner50 checking
    --end if;--clock50 cheking
    case sw(3) is
    when '1'=>order<=upcounter;
    when others=>order<=downcounter;
    end case;

end if;--end reset
end process;
end behavioral;
```

Decoder :

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

entity decoder7int is
port(
        C : in integer range 0 to 9;
        display : out std_logic_vector (0 to 6));
end decoder7int;

architecture beh of decoder7int is

signal cbits : std_logic_vector(3 downto 0);

begin

cbits <= std_logic_vector(to_unsigned(c,4));

process(cbits)
    begin
    case Cbits is
    when "0000"=>DISPLAY<="0000001";--0
    when "0001"=>DISPLAY<="1001111";--1
    when "0010"=>DISPLAY<="0010010";--2
    when "0011"=>DISPLAY<="0000110";--3
    when "0100"=>DISPLAY<="1001100";--4
    when "0101"=>DISPLAY<="0100100";--5
    when "0110"=>DISPLAY<="0100000";--6
    when "0111"=>DISPLAY<="0001111";--7
    when "1000"=>DISPLAY<="0000000";--8
    when "1001"=>DISPLAY<="0000100";--9
    when "1010"=>DISPLAY<="0001000";--A
    when "1011"=>DISPLAY<="1100000";--B
    when "1100"=>DISPLAY<="0110001";--C
    when "1101"=>DISPLAY<="1000010";--D
    when "1110"=>DISPLAY<="0110000";--E
    when others=>DISPLAY<="0111000";--F
    end case;
end process;
end beh;
```

Here is the video that shows the running program :

https://drive.google.com/file/d/1_UGyS9N8LA10KvV2XdagmnUWG1LS_JIG/view?usp=sharing