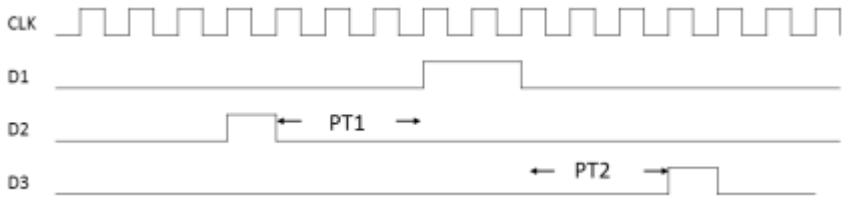
14 November 2014

Create a clock-synchronous VHDL entity which samples the inputs **D1**, **D2**, and **D3**. The entity must transfer to two 4-bits parallel outputs **PT1** and **PT2** the time intervals between the input pulses, measured in number of clock cycles.

Starting from a reset, **PT1** emits the time between the falling edge of the first impulse from one of the inputs and the rising edge of the second input who goes active. **PT2** emits the time between the falling edge of the second input signal to go active and the rising edge of the third input to go active.



If the counting is higher than the range allowed by 4 bits for **PT1** and/or **PT2**, 2 output signals **OV1** and **OV2**, which are normally set to 0, will emit a pulse for 1 clock cycle to signal this overflow condition.

The **ENABLE** signal can force **PT1** and **PT2** to high impedance, but doesn't change the behaviour of the entity and of the **OV1** and **OV2** outputs.

```
1   library IEEE;  use IEEE.STD_LOGIC_1164.ALL;
2   use IEEE.STD_LOGIC_ARITH.ALL; use IEEE.STD_LOGIC_UNSIGNED.ALL;
3
4   entity ESD20141114S is
5       Port ( CLK,D1,D2,D3,RESET,ENABLE : in std_logic;
6              OV1,OV2 : out std_logic;
7              PT1,PT2 : out std_logic_vector(3 downto 0));
8   end ESD20141114S;
9
10  architecture Behavioral of ESD20141114S is
11  signal cont, contOV1, contOV2: integer:=0; -- contatori
12  signal t1up,t1down,t2up, t2down,t3up,t3down: integer:=0; -- segnali per memorizzare i fronti D1,D2 e D3
13  signal PT1_int, PT2_int: integer:=0; -- segnali interni per il calcolo dei tempi parziali
14  signal PT1_out,PT2_out : std_logic_vector(3 downto 0):= (others=>'0');
15  signal OV1_out, OV2_out : std_logic; -- copia interna dei segnali OV1 ed OV2
16
17  begin
18  sincrono: process (CLK)  -- processo sincrono che conta gli impulsi di clock
19  begin
20  if RESET='1' then  -- azzero la macchina
21     cont <= 0; contOV1 <= 0; contOV2<= 0; OV1_out<='0'; OV2_out<='0';
22     PT1_int<=0;PT2_int<=0; PT1_out <= (others=>'0'); PT2_out <= (others=>'0');
23  elsif rising_edge(CLK) then
24        cont <= cont +1;
25  --1 caso: T1 < T2 < T3
26  if (t1down <= t2down and t2down <= t3down) then
27      if t2up > t1down then PT1_int <= t2up-t1down; else PT1_int <= 0; end if;
28      if t3up > t2down then PT2_int <= t3up-t2down; else PT2_int <= 0; end if;
29  end if;
30  --2 caso: T2 < T1 < T3
31  if (t2down <= t1down and t1down <= t3down) then
32      if t1up > t2down then PT1_int <= t1up-t2down; else PT1_int <= 0; end if;
33      if t3up > t1down then PT2_int <= t3up-t1down; else PT2_int <= 0; end if;
34  end if;
35  --3 caso: T1 < T3 < T2
36  if (t1down <= t3down and t3down <= t2down) then
37      if t3up > t1down then PT1_int <= t3up-t1down; else PT1_int <= 0; end if;
38      if t3up > t2down then PT2_int <= t2up-t3down; else PT2_int <= 0; end if;
39  end if;
40  --4 caso: T2 < T3 < T1
41  if (t2down <= t3down and t3down <= t1down) then
42      if t3up > t2down then PT1_int <= t3up-t2down; else PT1_int <= 0; end if;
43      if t1up > t3down then PT2_int <= t1up-t3down; else PT2_int <= 0; end if;
44  end if;
45  --5 caso: T3 < T1 < T2
46  if (t3down <= t1down and t1down <= t2down) then
47      if t1up > t2down then PT1_int <= t1up-t2down; else PT1_int <= 0; end if;
48      if t2up > t1down then PT2_int <= t2up-t1down; else PT2_int <= 0; end if;
49  end if;
50  --6 caso: T3 < T2 < T1
51  if (t3down <= t2down and t2down <= t1down) then
52      if t2up > t3down then PT1_int <= t2up-t3down; else PT1_int <= 0; end if;
53      if t1up > t2down then PT2_int <= t1up-t2down; else PT2_int <= 0; end if;
54  end if;
55  -- gestione degli intervalli di tempo che vanno in overflow (maggiore di 15 clock)
56  if PT1_int >15 then
57     PT1_out <= "1111"; OV1_out <='1'; contOV1 <= contOV1 + 1;
58     else
59     PT1_out <= CONV_STD_LOGIC_VECTOR (PT1_int, 4);
60  end if;
61  if PT2_int >15 then
62     PT2_out <= "1111"; OV2_out <='1'; contOV2 <= contOV2 + 1;
63     else
64     PT2_out <= CONV_STD_LOGIC_VECTOR (PT2_int, 4);
65  end if;
66  end if;
67  end process;
68
69  asincrono: process (D1,D2,D3, CLK)  -- Processo asincrono che intercetta i fronti dei segnali D1, D2 e D3
70  begin
71  if RESET='1' then  -- azzero la macchina
72     t1up<=0; t1down<=0;t2up<=0; t2down<=0;t3up<=0;t3down<=0;
73  else
74       if rising_edge (D1) then t1up<= cont; end if;
75       if rising_edge (D2) then t2up<= cont; end if;
76       if rising_edge (D3) then t3up<= cont; end if;
77       if falling_edge (D1) then t1down<= cont; end if;
78       if falling_edge (D2) then t2down<= cont; end if;
79       if falling_edge (D3) then t3down<= cont; end if;
80  end if;
81  end process;
82  -- Parte combinatoria che genera le uscite vere e proprie e controlla l'ENABLE
83  PT1 <= PT1_out when ENABLE ='1' else "ZZZZ";
84  PT2 <= PT2_out when ENABLE ='1' else "ZZZZ";
85  OV1 <= OV1_out when contOV1 = 1 else '0';
86  OV2 <= OV2_out when contOV2 = 1 else '0';
87  end Behavioral;
```