Compte rendu de Travaux pratiques de TCN

Réalisé par: Issa SIDIBE et Jude Alexandres N'GOMA
GROUPE E
L3_IE

Introduction :

Le langage VHDL est un langage de description de circuit les séances ont pour objectifs de bien cerné le langage VHDL faisant intervenir les circuits séquentielles et combinatoire ainsi la programmation réelle de circuit FPGA pour vérifier les codes écrites.

Séance 1

1_Fonctions séquentielle

1-1_ Diviseur de fréquence par 2 (bascule D)

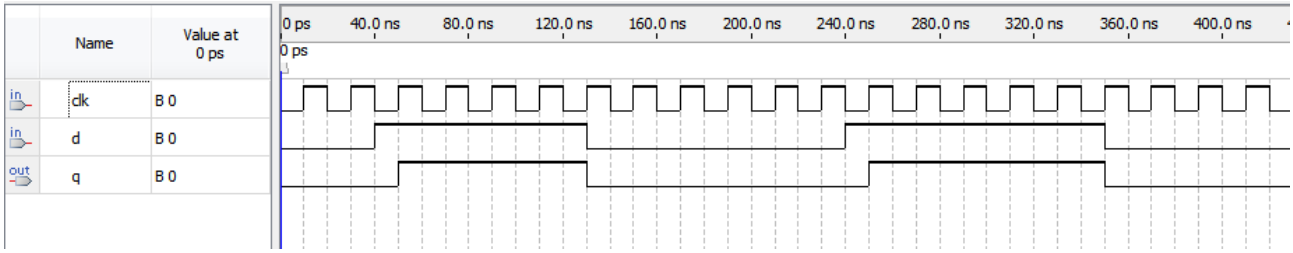
code

```
LIBRARY IEEE;---declaration des bibliothèque indispensable au bon fonctionnement de la descriptioon
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

ENTITY dff_2 IS
PORT (---- declaration des ports
        clk: IN std_logic;
        d: IN std_logic;
        q: OUT std_logic
        );

END dff_2;

ARCHITECTURE archi OF dff_2 IS
        BEGIN
                PROCESS(clk)----pour les instructions séquentielles
                BEGIN
                IF RISING_EDGE(clk) THEN q<=d;----On récopie la valeur de l'entrée sur la sortie à chaque front d'horloge
                END IF;
        END PROCESS;
END archi;
```

Travail demandé

1-

- Simulation



on voit bien que la sortie suit bien l'entrée à chaque front d'horloge.

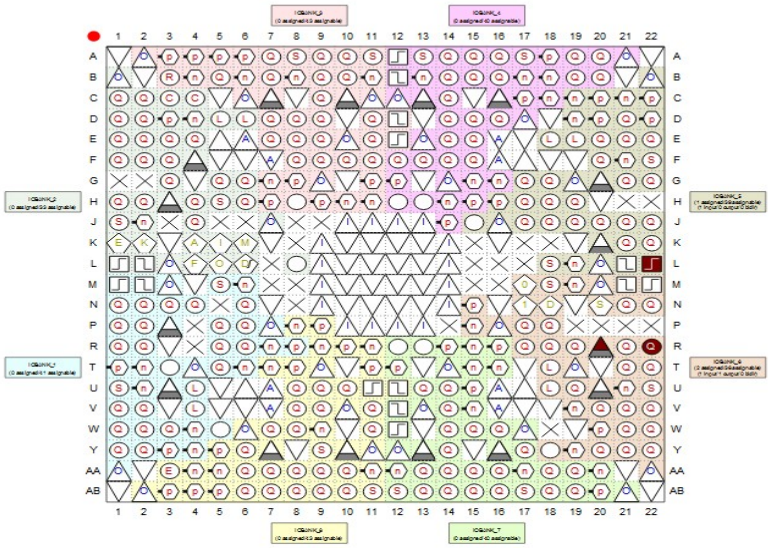
- La programmation du FPGA

Nous avons commencé par l'assignation des pins

Named: * Edit:

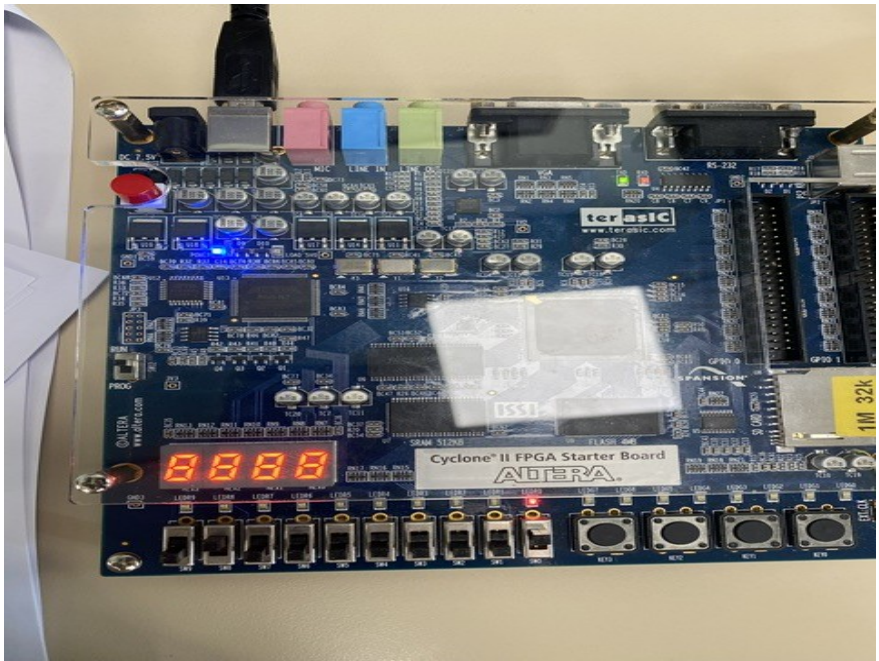
Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Differential Pair	Fitter Location
in clk	Input	PIN_R22	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R22
in d	Input	PIN_L22	5	B5_N1	3.3-V LV...default)		24mA (default)		PIN_L22
out q	Output	PIN_R20	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R20
<<new node>>									

Top View - Wire Bond
Cyclone II - EP2C20F484C7



-Puis nous avons chargé notre code vhdl sur le circuit FPGA

En ayant un front d'horloge(KEY 0) et en appuyant sur SW 0 la LED R0 s'allume (voir image en dessous)



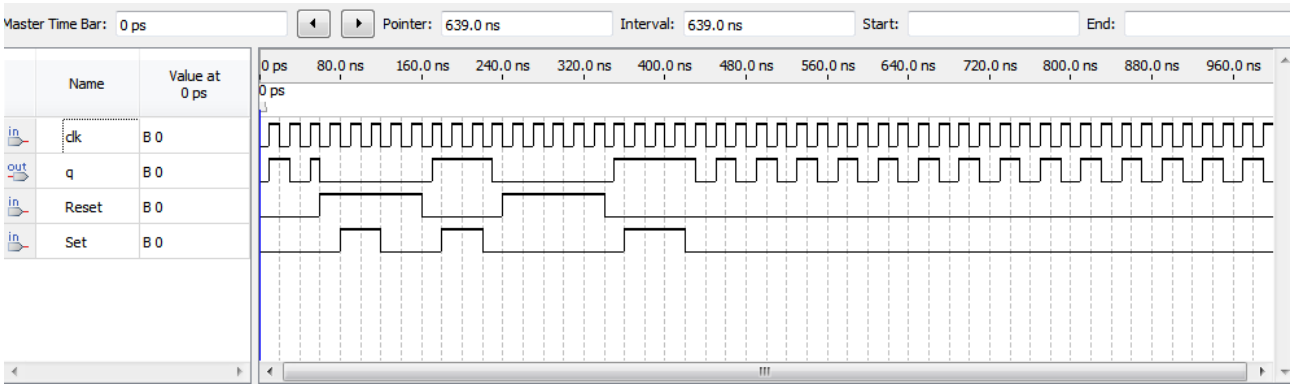
2- **Dans le cas où on boucle la sortie sur l'entrée**

- Rapport entre la fréquence de sortie et celle de l'entée :
on remarque la fréquence de sortie **q** est égale a la fréquence **clk** divisé par 2
Nous aurons bien un diviseur de fréquence par deux grâce au bouclage

CODE

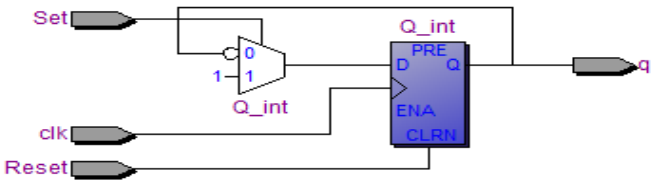
```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

ENTITY dff_21 IS
PORT (clk:  IN std_logic;
        Reset     :  IN std_logic;
        Set       :  In Std_logic;
        q                 : OUT std_logic);
END dff_21;

ARCHITECTURE archi OF dff_21 IS

SIGNAL Q_int : STD_LOGIC;-- car la sortie ne peut pas lire la sortie on passe par un signal intermediarie

BEGIN
        PROCESS(clk,Reset,Set)
        BEGIN
                IF Reset ='1' THEN--remise à 0 de la sortie
                        Q_int     <=        '0';
                ELSIF RISING_EDGE(clk) THEN
                        IF set='1' THEN--remise à 1 de la sortie
                                Q_int     <= '1';
                        ELSE
                                Q_int     <= NOT Q_int;--le signal intermédiaire prend la valeur de son complémentaire

                        END IF;
                END IF;
        END PROCESS;
        q         <= Q_int;---c'est à la fin du process que le sortie prend la valeur du signal intermédiaire
END archi;
```

- Simulation



on remarque bien que la période de la sortie vaut 2 fois la période de l'horloge d'où la fréquence de sortie est bien divisé par 2

-Schéma bloc du code Vhdl

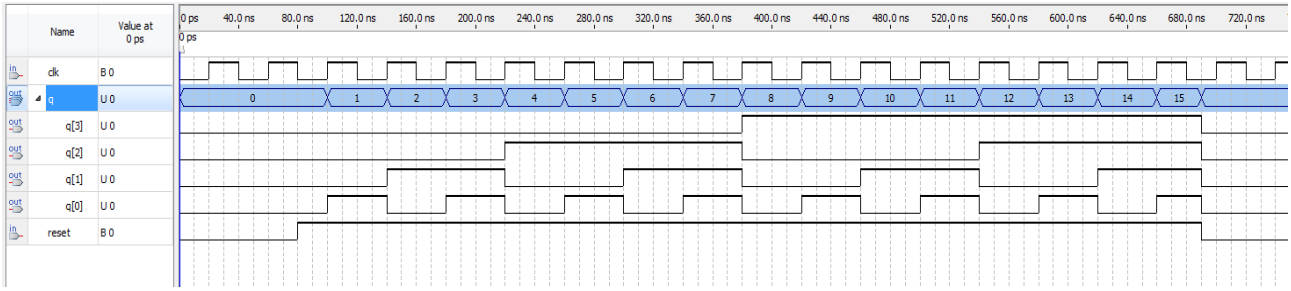


1-2 Compteur avec Reset asynchrone

-Code source

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;--pour prendre en compe les type std logic
use IEEE.numeric_std.all;--pour le front montant
ENTITY compteur_4 is
    PORT(
        clk: In std_logic;
        reset: In std_logic;
        q: OUT unsigned(3 downto 0));--valeur positive
    END compteur_4;
Architecture archi of compteur_4 is
SIGNAL q_int:unsigned(3 downto 0);--signal intermediaire positif
                                  --pour ne pas directement relier la sortie à l'entrée
    BEGIN
        PROCESS(clk,reset)
            BEGIN
            IF reset='0' THEN q_int<=(others=>'0');
               ELSIF rising_edge(clk) THEN
                  q_int<=q_int+1;---incrementaion de la sortie
            END IF;
        END PROCESS;
        q<=q_int;---la sortie prend la valeur du signal à la fin du process

    END archi;
```

- Simulation fonctionnelle



-Rapport entre les fréquences des signaux :
il y a un rapport de 2 entre les entre les fréquences des signaux de q(0) q(1) q(2) et q(3) alors la fonction réalisé : diviseur de fréquence par 2

-La vraie fonction du système entier est d'effectué un comptage de 0 à 15 . C'est un compteur 4 bits

on remarque l'apparition d'un retard sur les signaux de sortie
en conclusion on peut dire que cela est du au fait que les sortie ne bascule pas en même temps on a une simulation réelle

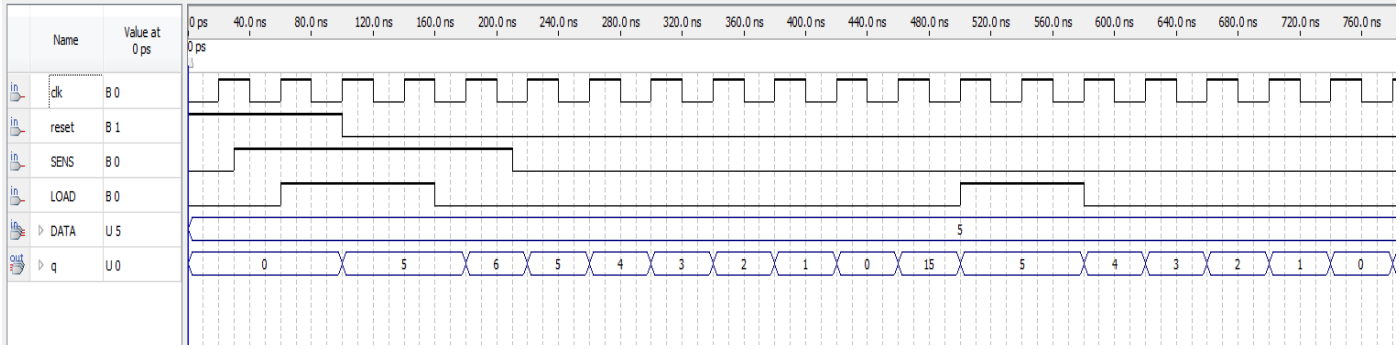
2-<u>Conception et réalisation d'un compteur/ décompteur 4 bits</u>

Code VHDL :

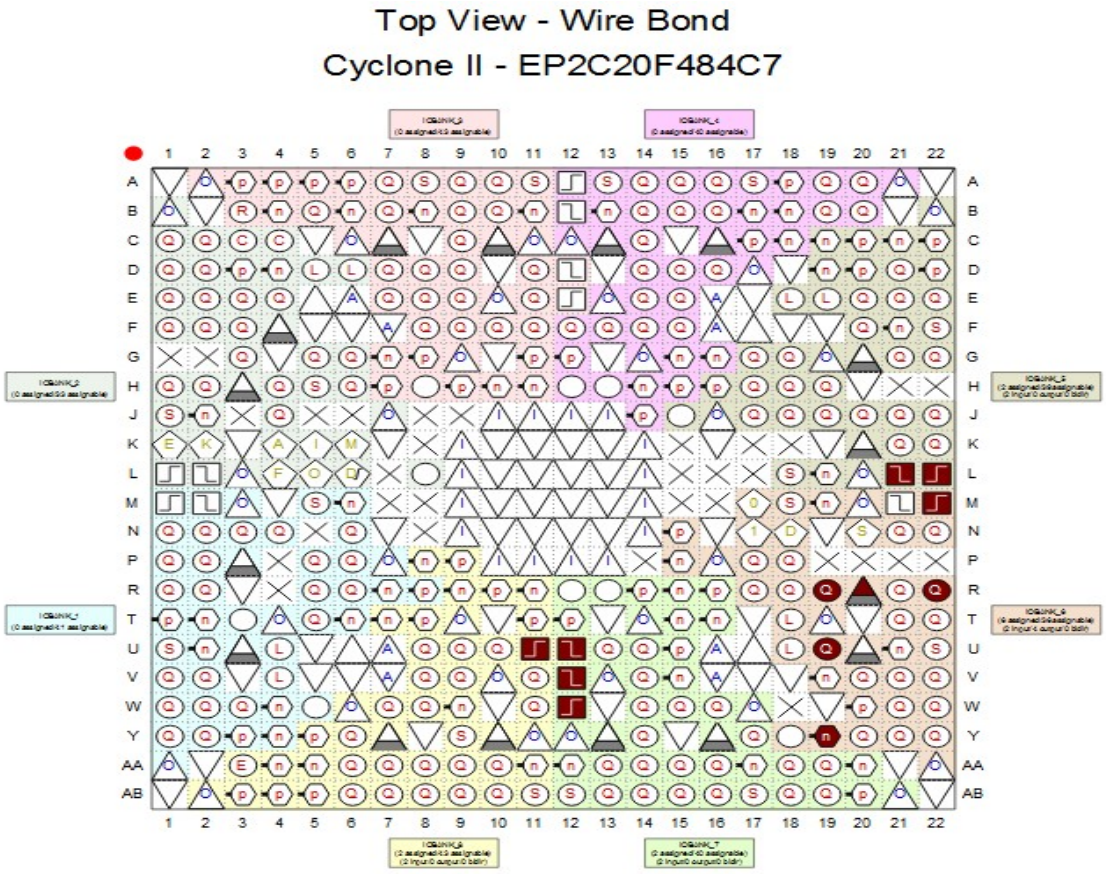
```
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  ENTITY counter_D is
5     PORT(
6        clk    :  In std_logic;
7        reset  :  In std_logic;
8        LOAD   :  In std_logic;
9        SENS   :  In std_logic;
10       DATA   :  In unsigned (3 downto 0);
11       q      : OUT unsigned(3 downto 0));
12    END counter_D;
13
14 Architecture archi of counter_D is
15
16 SIGNAL q_int : unsigned(3 downto 0);
17
18    BEGIN
19       PROCESS(clk,reset,LOAD,SENS,DATA)
20          BEGIN
21          IF reset = '1' THEN
22             q_int <= (others=>'0');
23          ELSIF rising_edge(clk) THEN
24             IF LOAD = '1' THEN
25                q_int <= DATA;
26             ELSE
27                IF SENS = '1' THEN
28                   q_int <= q_int + 1;
29                ELSE
30                   q_int <= q_int - 1;
31                END IF;
32             END IF;
33          END IF;
34       END PROCESS;
35
36    q <= q_int;
37
38    END archi;
39
```

<u>simulation fonctionnelle :</u>

Simulation fonctionnelle :



Test circuit FPGA



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Differential Pair	Fitter Location
clk	Input	PIN_R22	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R22
DATA[3]	Input	PIN_U11	8	B8_N0	3.3-V LV...default)		24mA (default)		PIN_U11
DATA[2]	Input	PIN_U12	8	B8_N0	3.3-V LV...default)		24mA (default)		PIN_U12
DATA[1]	Input	PIN_W12	7	B7_N1	3.3-V LV...default)		24mA (default)		PIN_W12
DATA[0]	Input	PIN_V12	7	B7_N1	3.3-V LV...default)		24mA (default)		PIN_V12
LOAD	Input	PIN_L21	5	B5_N1	3.3-V LV...default)		24mA (default)		PIN_L21
q[3]	Output	PIN_Y19	6	B6_N1	3.3-V LV...default)		24mA (default)		PIN_Y19
q[2]	Output	PIN_U19	6	B6_N1	3.3-V LV...default)		24mA (default)		PIN_U19
q[1]	Output	PIN_R19	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R19
q[0]	Output	PIN_R20	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R20
reset	Input	PIN_L22	5	B5_N1	3.3-V LV...default)		24mA (default)		PIN_L22
SENS	Input	PIN_M22	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_M22
<<new node>>									

symbole graphique :

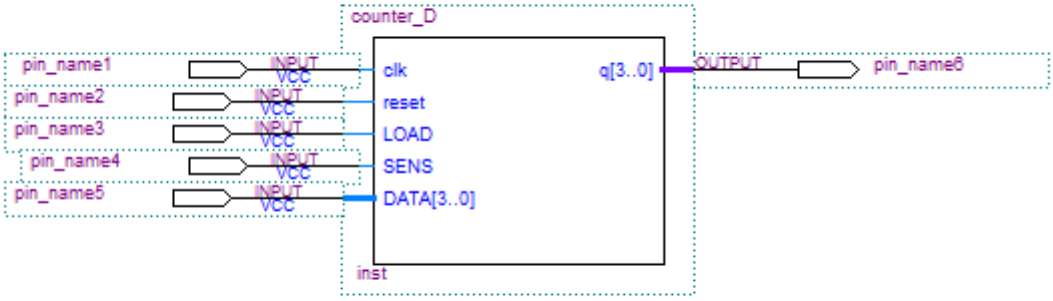


Photo 1 : tout a zero

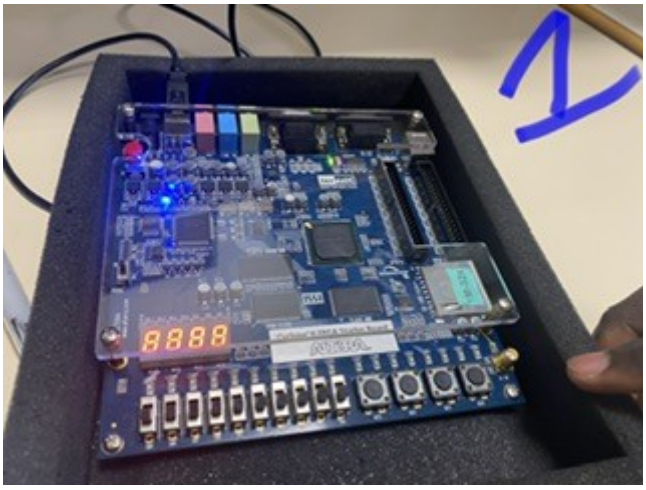


photo 2 appuie sur clock

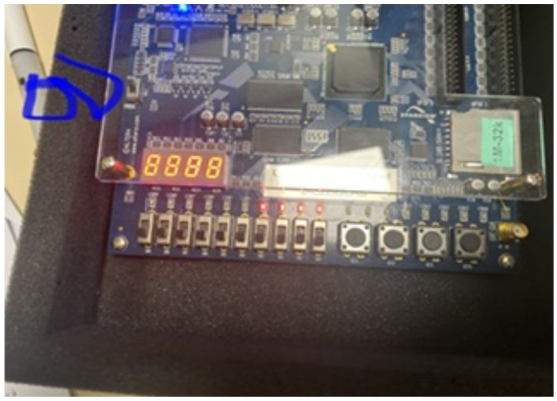


photo 3 test du reset

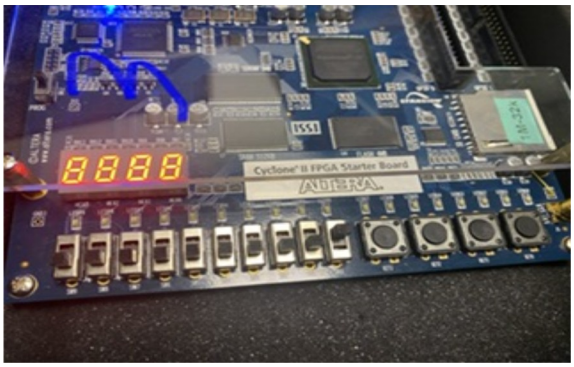
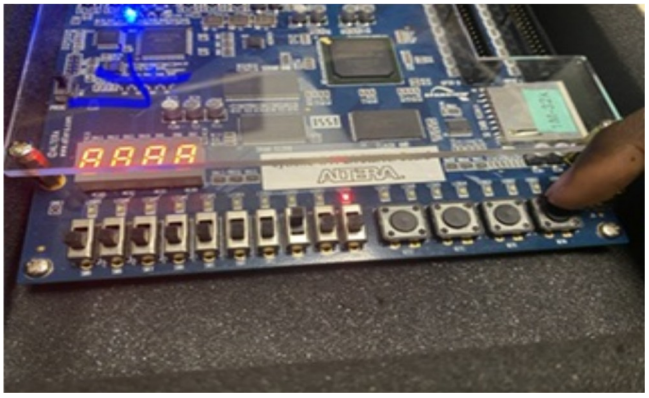
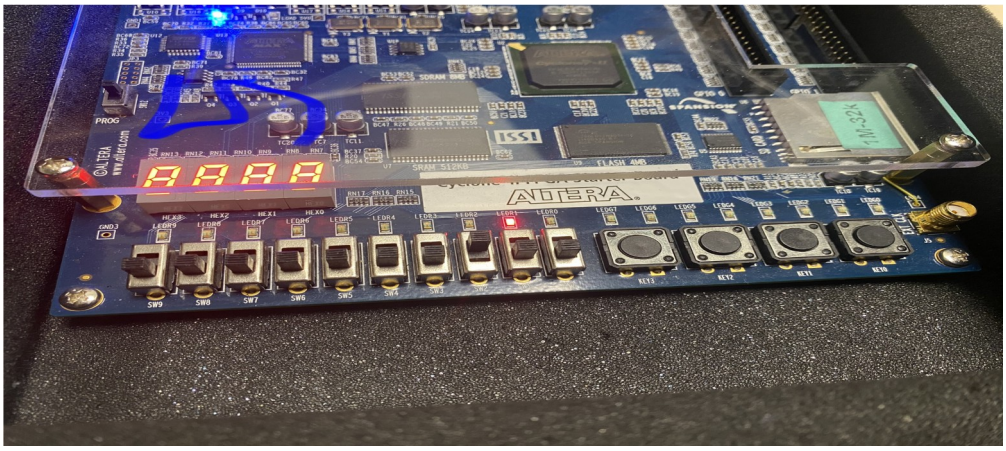
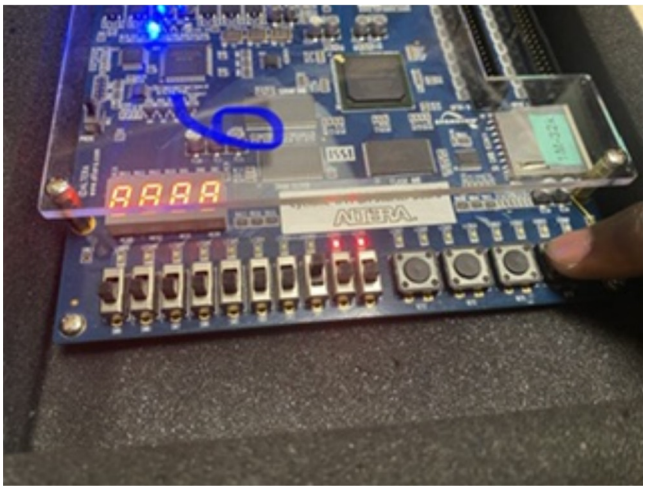


photo 4 test incrementation et decrementation





photot5 : teste du load (sortie prend bien la valeur du Load)
Avec 3 valeurs différentes

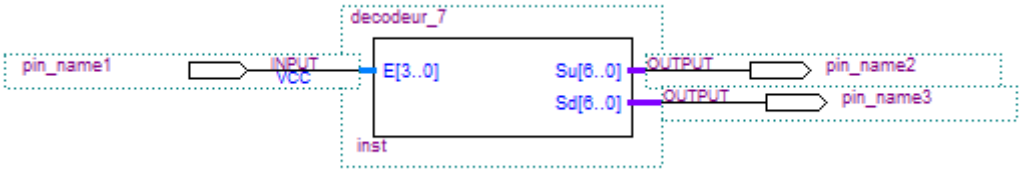


-Decodeur 7 segments

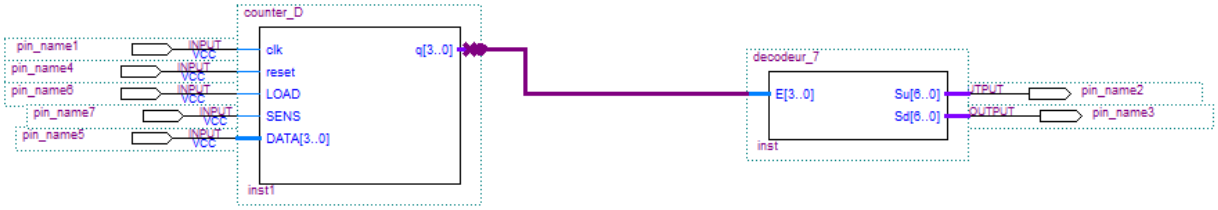
Code Vhdl : 2 afficheurs 7 segments (unité et dizaine) sous récommendation du prof

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
Entity decodeur_7 is
PORT(
     E : in std_logic_vector(3 downto 0);
     Su : out std_logic_vector(6 downto 0);
     Sd : out std_logic_vector(6 downto 0)
     );
END decodeur_7;

Architecture archi of decodeur_7 is
   Begin
     process (E)
        Begin
        case E is
          when "0000" =>
            Sd <= "1000000";---- la sortie des dizaines à 0
            Su <= "1000000";-----la sortie des unités à 0
          when "0001" =>
            Sd <= "1000000";
            Su <="1111001";
          when "0010" =>
             sd <="1000000";
             su <= "0100100";
          when "0011" =>
             sd <= "1000000";
             su <= "0110000";
          when "0100" =>
             sd <= "1000000";
             su <= "0011001";
          when "0101" =>
             sd <= "1000000";
             su <= "0010010";
          when "0110" =>
             sd <= "1000000";
             su <= "0000010";
          when "0111" =>
             sd <= "1000000";
             su <= "1011000";
          when "1000" =>
             sd <= "1000000";
          when "1010" =>
            Sd <="1111001";  ----les dizaines à 1
            Su <= "1000000";----les unités à 0
          when "1011"  =>
            Sd <= "1111001";
            Su <= "1111001";
          when "1100" =>
             Sd <= "1111001";
             su <= "0100100";
          when "1101" =>
             Sd <= "1111001";
             su <= "0110000";
          when "1110" =>
             Sd <="1111001";
             su <= "0011001";
          when "1111" =>
             Sd <= "1111001";
             su <="0010010";
             -- on s'arrête à 15 car on est limité a 4 bit 2^4= 16
             -- 16 combinaison possible au maximum
             END case  ;
     END process ;

   END archi;
```

Schéma bloc du code



Association du compteur avec le decodeur 7 segments

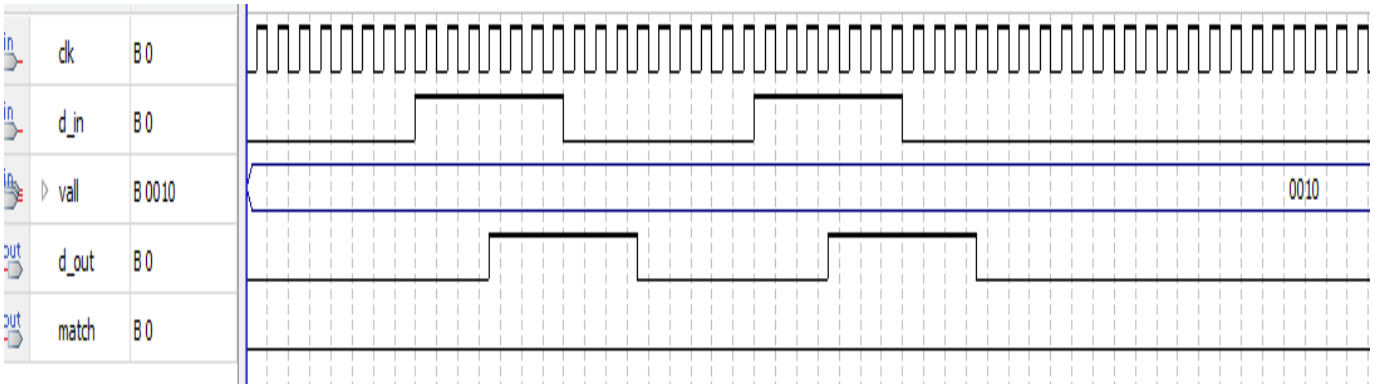


1-3 Registre à décalage

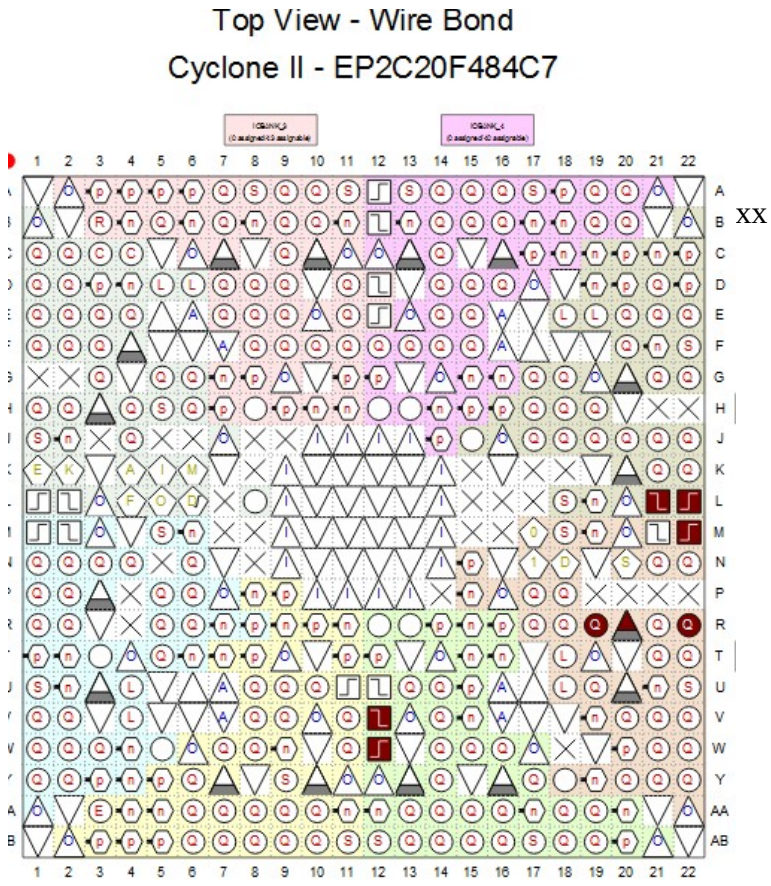
Code VHDL

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
   ENTITY register_4 is
      Port (
         clk : IN std_logic ;
         d_in : IN std_logic ;
         vall : IN std_logic_vector(3 downto 0);
         d_out: OUT std_logic;
         match: OUT std_logic);

   END register_4;
ARCHITECTURE archi OF register_4 IS
   signal reg: std_logic_vector(3 downto 0);
      BEGIN
         process(clk)
         BEGIN
            IF rising_edge(clk) THEN
            --reg(3)<= d_in;
            --reg(3 downto 1)<= reg(2 downto 0);--------decalage vers le poids du bit le plus fort

            reg <= reg(2 downto 0)& d_in;
            END IF;
         END PROCESS;
            d_out <= reg(3);
            match <='1' when vall = reg else '0';
   END archi;
```

simulation fonctionnelle du registre



Programmation du FPGA



XX

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Differential Pair	Fitter Location
clk	Input	PIN_R22	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R22
d_in	Input	PIN_L22	5	B5_N1	3.3-V LV...default)		24mA (default)		PIN_L22
d_out	Output	PIN_R20	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R20
match	Output	PIN_R19	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_R19
vall[3]	Input	PIN_W12	7	B7_N1	3.3-V LV...default)		24mA (default)		PIN_W12
vall[2]	Input	PIN_V12	7	B7_N1	3.3-V LV...default)		24mA (default)		PIN_V12
vall[1]	Input	PIN_M22	6	B6_N0	3.3-V LV...default)		24mA (default)		PIN_M22
vall[0]	Input	PIN_L21	5	B5_N1	3.3-V LV...default)		24mA (default)		PIN_L21
<<new node>>									

Photo 1 tout a zéro val = reg alors match = 1 (LED R1 allumé)

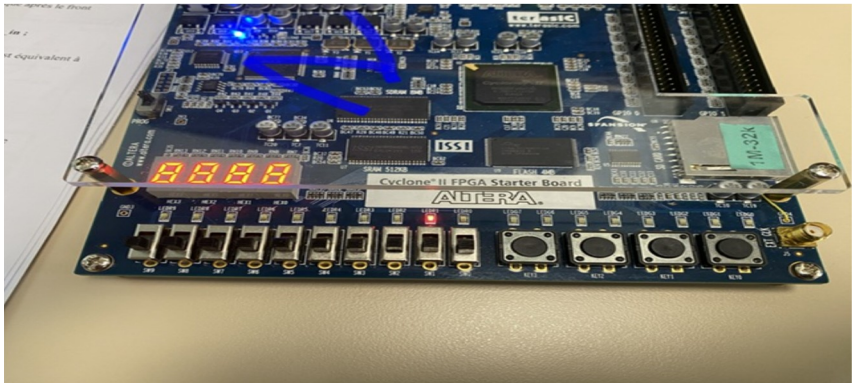
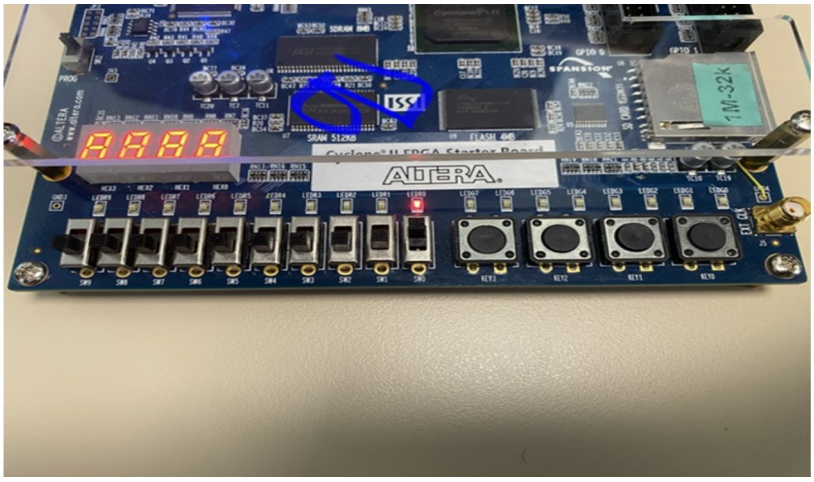


photo 2 : a partir du 4 eme front d’horloge d out = 1 (il prend la valeur de d_in)



Modification du code VHDL pour réaliser un décalage vers le bit du poids faible

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
   ENTITY lowregis is
      Port (
         clk : IN std_logic ;
         d_in : IN std_logic ;
         vall : IN std_logic_vector(3 downto 0);
         d_out: OUT std_logic;
         match: OUT std_logic);

   END lowregis;
ARCHITECTURE archi OF lowregis IS
   signal reg: std_logic_vector(3 downto 0);
      BEGIN
         process(clk)
         BEGIN
            IF rising_edge(clk) THEN

            reg <= d_in & reg(3 downto 1);

            END IF;
         END PROCESS;
            d_out <= reg(0);
            match <='1' when vall = reg else '0';
   END archi;
```

Simulation fonctionelle du code modifier

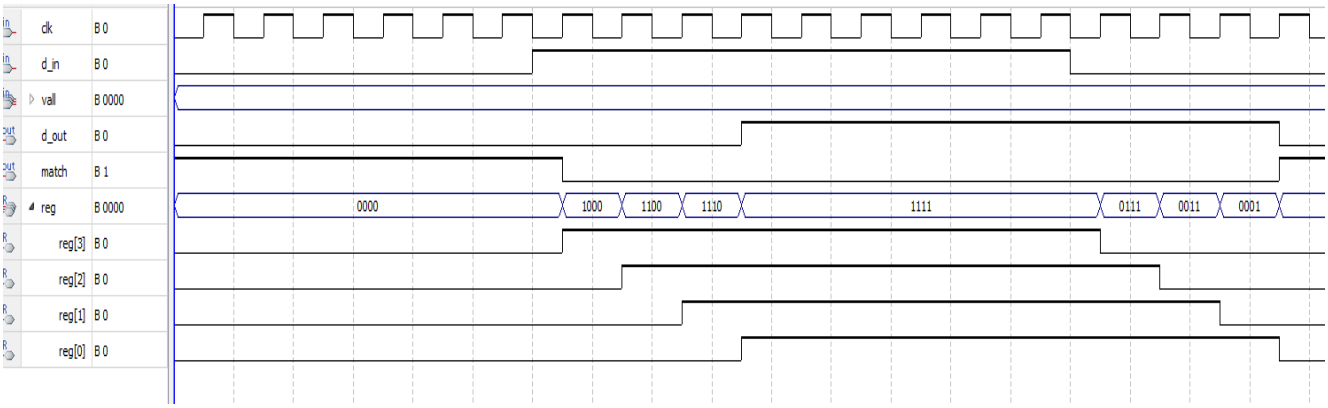


Photo 1 :

—

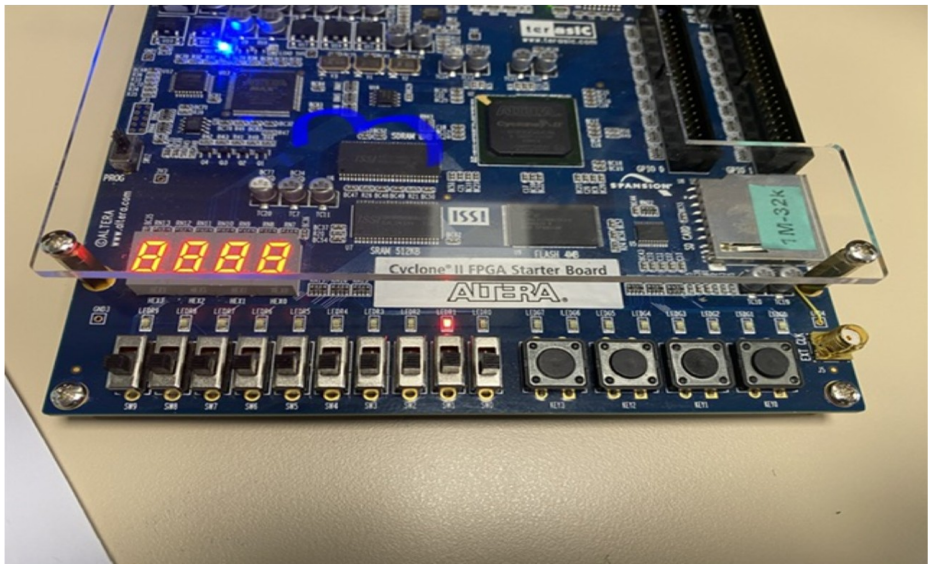
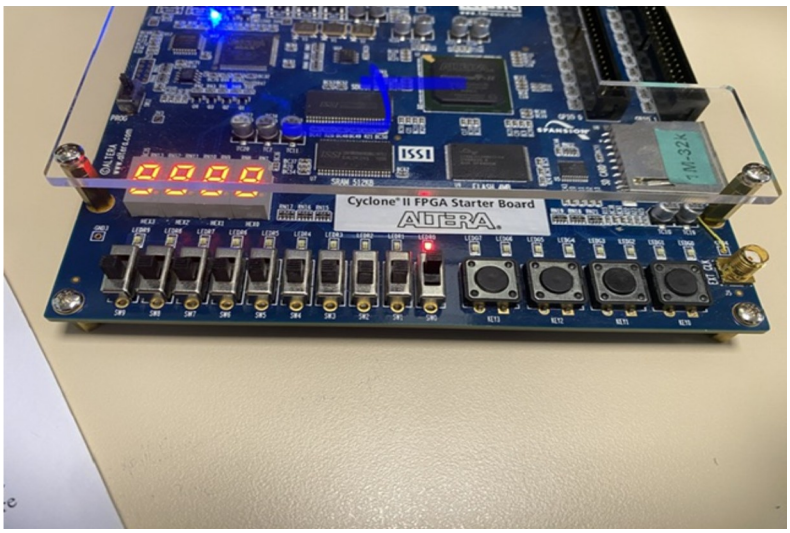


Photo 2 :



Conception et réalisation d'une transformation d'un flux série en bus de 4 bits

Code VHDL registre parallèle

```vhdl
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity PARALLELE2 is
    port (
        clk : in std_logic; -- entrée d'horloge
        data_in : in std_logic_vector(3 downto 0); -- entrée de données
        data_out : out std_logic_vector(3 downto 0) -- sortie de données
    );
end entity;

architecture behavior of PARALLELE2 is
    signal reg_data : std_logic_vector(3 downto 0); -- signal interne pour stocker les données

begin
    process(clk)
    begin
        if rising_edge(clk) then -- détecte le front montant de l'horloge
            reg_data <= data_in; -- stocke les données d'entrée dans le registre
        end if;
    end process;

    data_out <= reg_data; -- renvoie les données stockées en sortie
end architecture;
```

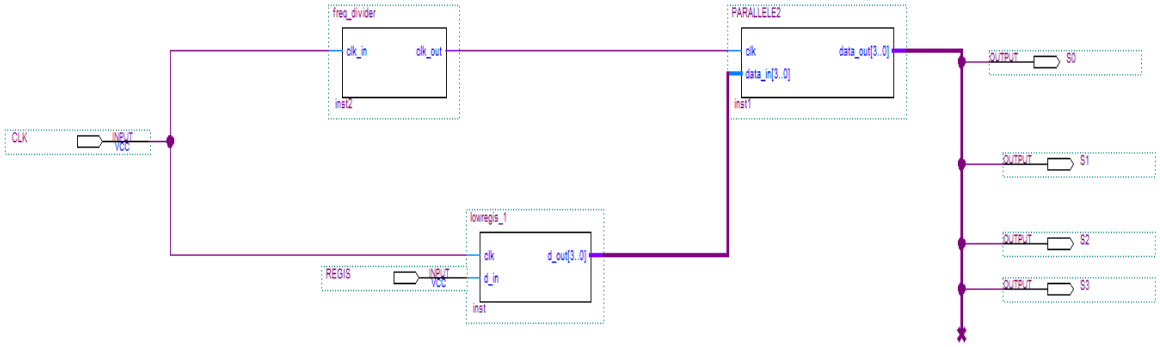
Code VHDL registre à décalage

```vhdl
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

ENTITY lowregis_1 is
    Port (
        clk : IN std_logic ;
        d_in : IN std_logic ;

        d_out: OUT std_logic_vector(3 downto 0)

    );
END lowregis_1;

ARCHITECTURE archi OF lowregis_1 IS
    signal reg : std_logic_vector(3 downto 0);
BEGIN
    process(clk)
    BEGIN
        IF rising_edge(clk) THEN
            reg <= d_in & reg(3 downto 1);
        END IF;
    END PROCESS;
    d_out <= reg;

END archi;
```

Code VHDL diviseur de fréquence

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity freq_divider is
  port (
    clk_in : in std_logic;
    clk_out : out std_logic
  );
end entity freq_divider;

architecture Behavioral of freq_divider is
  signal counter : integer range 0 to 3 := 0;
  signal temp_clk : std_logic := '0';
begin
  process(clk_in)
  begin
    if rising_edge(clk_in) then
      if counter = 3 then
        temp_clk <= not temp_clk;
        counter <= 0;
      else
        counter <= counter + 1;
      end if;
    end if;
  end process;

  clk_out <= temp_clk;
end architecture Behavioral;
```

Description graphique



Simulation fonctionnelle

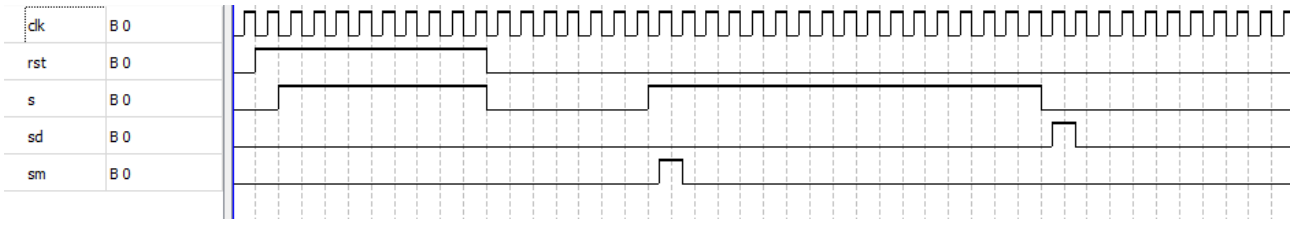
1-4 Machine d'état synchrone :

Code VHDL

```
LIBRARY IEEE;---declaration de la bibliothèque
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
Entity machine is
   port (--declarations des entrées et sortie
   clk: in std_logic;
   s: in std_logic;
   rst: in std_logic;
   sm,sd: out std_logic
   );
end machine;
architecture arch of machine is
   type statetype is (etat0, etat1, etat2, etat3);--declaration des états
   signal etat : statetype;--declaration du signal pour stocker les valeurs de "etat"
 Begin
      process (clk)
      Begin
         if rst = '1' then sm<='0';sd<='0';-- teste du reset

         elsif rising_edge(clk) then

            case etat is
            when etat0 => if s='1'then ---lorsqu'on est a l'etat0 et si "s=0 ou 1"
                             etat <= etat1;
                             sm<='1';-- debut impulsion
                             else etat <= etat0;
                          end if ;

            when etat1 =>     sm<= '0';---remettre sm à 0
                              etat <= etat2 ;

            when etat2 => if s='0' then
                             etat <= etat3;
                                sd<='1';---fin impulsion
                             else etat<= etat2;
                          end if ;

            when etat3 =>  sd<='0';
                           etat <= etat0;

            end case;
         end if;
      end process;
   end arch;
```

Simulation fonctionnelle :

lorsque rst = 1 les deux sorties se mette à zero
on voit bien le debut et la fin de l'impulsion



MINI-PROJET : Convertisseur binaire -décimal :

code VHDL circuit A

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
Entity circA is
PORT(
     V : in std_logic_vector(2 downto 0);
     S : out std_logic_vector(2 downto 0)
     );
END circA;

Architecture archi of circA is
   Begin
      process (V)
         Begin
         case V is
           when "010" =>
             S <= "000";---- entree =10

           when "011" =>
             S <= "001";

           when "101" =>
              S <="010";

           when "110" =>
              S <= "011";

           when "111" =>
              S <= "101";
           when others => S<= "000";
           END case  ;
      END process ;

   END archi;
```

code VHDL circuit B

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
Entity circB is
   port (
        e: in std_logic ;
        Sb: out std_logic_VECTOR (6 downto 0)
        );
end circB;
architecture TOTO of circB is
   begin
   with e select
      Sb<= "1000000" when '0',
           "1111001" when '1';

   end TOTO;
```

code VHDL MUX 2 1

```vhdl
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
Entity mux2_1 is
    port (
          e: in std_logic_vector(1 downto 0) ;
          c: in std_logic;
          s: out std_logic
          );
end mux2_1;
architecture TOTO of mux2_1 is
    begin
    with c select
       s<= e(0) when '0',
           e(1) when '1',
           '0' when others ;
    end TOTO;
```

CODE VHDL 7 SEGMENTS

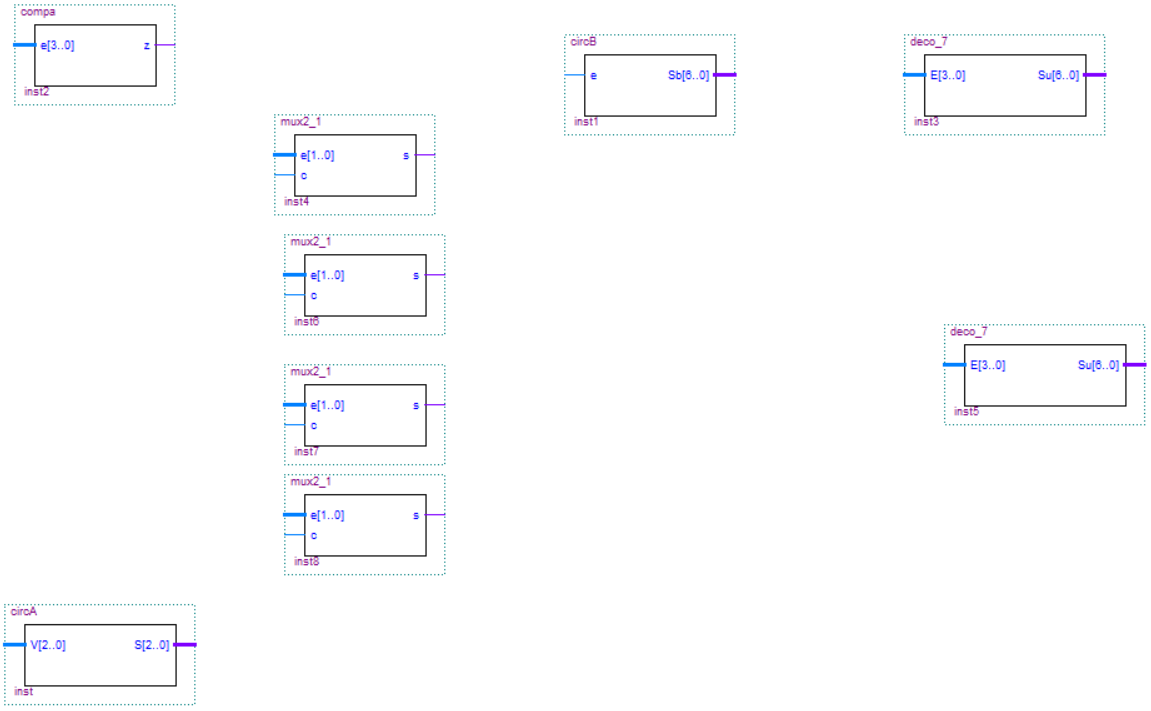
```vhdl
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
Entity deco_7 is
PORT(
     E : in std_logic_vector(3 downto 0);
     Su : out std_logic_vector(6 downto 0)

     );
END deco_7;

Architecture archi of deco_7 is
    Begin
       process (E)
          Begin
          case E is
            when "0000" =>

              Su <= "1000000";-----la sortie des unités à 0
            when "0001" =>

              Su <="1111001";
            when "0010" =>

               su <= "0100100";
            when "0011" =>

               su <= "0110000";
            when "0100" =>

               su <= "0011001";
            when "0101" =>

               su <= "0010010";
            when "0110" =>

               su <= "0000010";
            when "0111" =>

               su <= "1011000";
            when "1000" =>

               su <= "0000000";
            when "1001" =>

               su <= "0010000";
            when others => su<= "1000000";
```

```
            when others => su<= "1000000";
               END case  ;
        END process ;

END archi;
```

CODE VHDL COMPARATEUR

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
Entity compa is
    port (
          e: in unsigned(3 downto 0) ;

          z: out std_logic
          );
end compa;
architecture TOTO of compa is
    begin
       process (e)
          begin
          if e >"1001" then
                z <= '1';
             else z<= '0';
          end if ;
       end process;
    end TOTO;
```

GRAPHIQUE



CONCLUSION :
enfin on on a réalisé un convertisseur binaire decimal avec deux afficheurs
TP TRES INSTRUCTIF