

## **Rapport TP TCN**

### **SYNTHESE DE CIRCUITS NUMERIQUES EN LANGUAGE VHDL ET IMPLEMENTATION DANS UN FPGA**

**Groupe C : OUEDRAOGO TOUWENDE  
SANAE KHLIF**

- **Introduction**

Le VHDL est basé sur la logique de description de comportement, qui permet aux concepteurs de spécifier le comportement d'un circuit numérique en utilisant des constructions logiques et des opérations mathématiques.

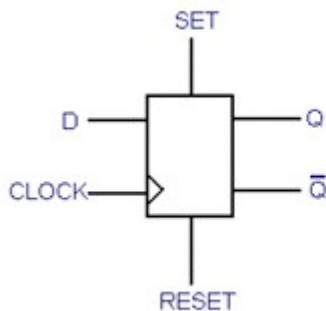
Durant ces séances de Travaux pratique nous allons étudier les concepts de modélisation, de simulation et la vérification des circuits par la réalisation de quelques exemples pratiques sur implantation dans un FPGA afin de mieux visualiser et comprendre le comportement physique des circuits .

## ● Fonctions séquentielles

Les systèmes séquentiels sont des systèmes électroniques qui peuvent stocker de l'information et effectuer des opérations sur cette information en fonction de l'état précédent du système. Contrairement aux systèmes combinatoires qui ne dépendent que des entrées actuelles, ils prennent en compte les entrées précédentes et l'état interne du système pour déterminer les sorties actuelles et sont couramment utilisés dans les circuits logiques, les systèmes de contrôle, les ordinateurs, les télécommunications et de nombreuses autres applications.

## Projet1: Réalisation d'un Diviseur de fréquence par 2, exemple de la bascule D

### table de vérité



Clk	D	Q	$\bar{Q}$	
0, 1, ↓	X	Q <sub>0</sub>	$\bar{Q}_0$	Mémoire
↑	0	0	1	(Reset)
↑	1	1	0	(Set)

## ● Description VHDL

```

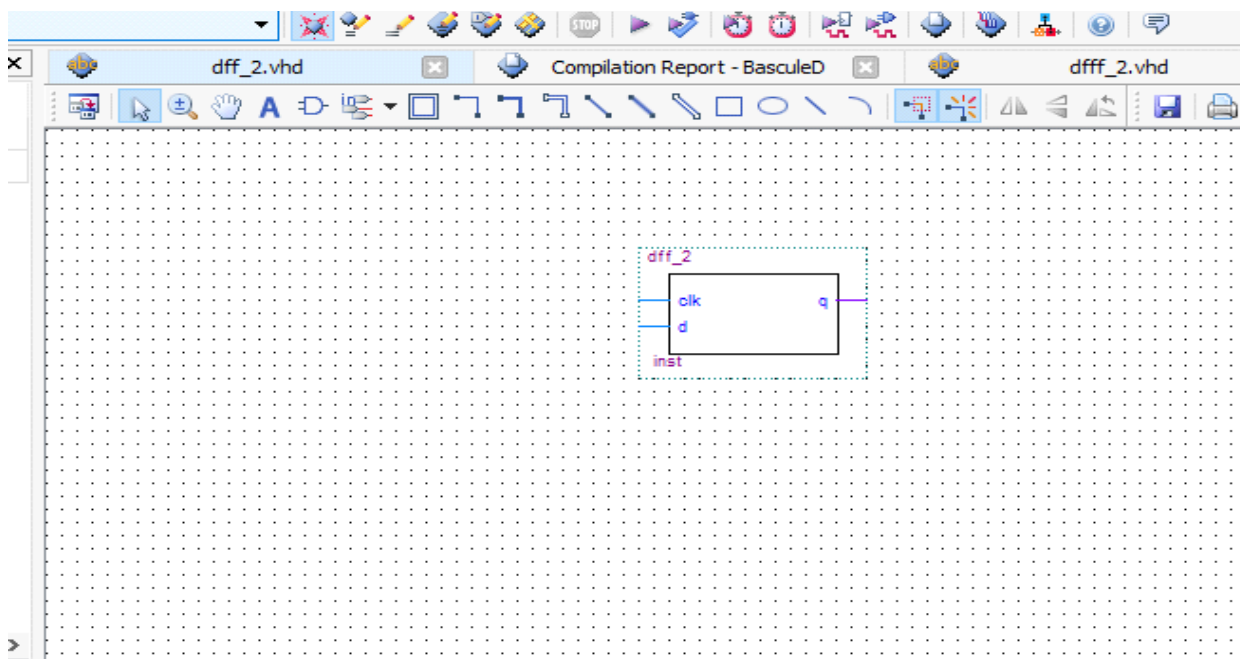
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY BASCULE_D IS
6  PORT (
7      clk: IN std_logic;
8      d: IN std_logic;
9      q: OUT std_logic;
10     h:OUT std_logic --cette sortie nous permet de visualiser l'horloge clk à l'aide d'une LED
11 );
12 end BASCULE_D;
13
14 ARCHITECTURE archi OF BASCULE_D is
15 BEGIN
16     PROCESS(clk)
17     BEGIN
18         if RISING_EDGE(clk) THEN q<=d;
19         END IF;
20         h<=clk; --h recopie l'horloge principale clk
21     end PROCESS;
22 END archi;
23
24 -

```

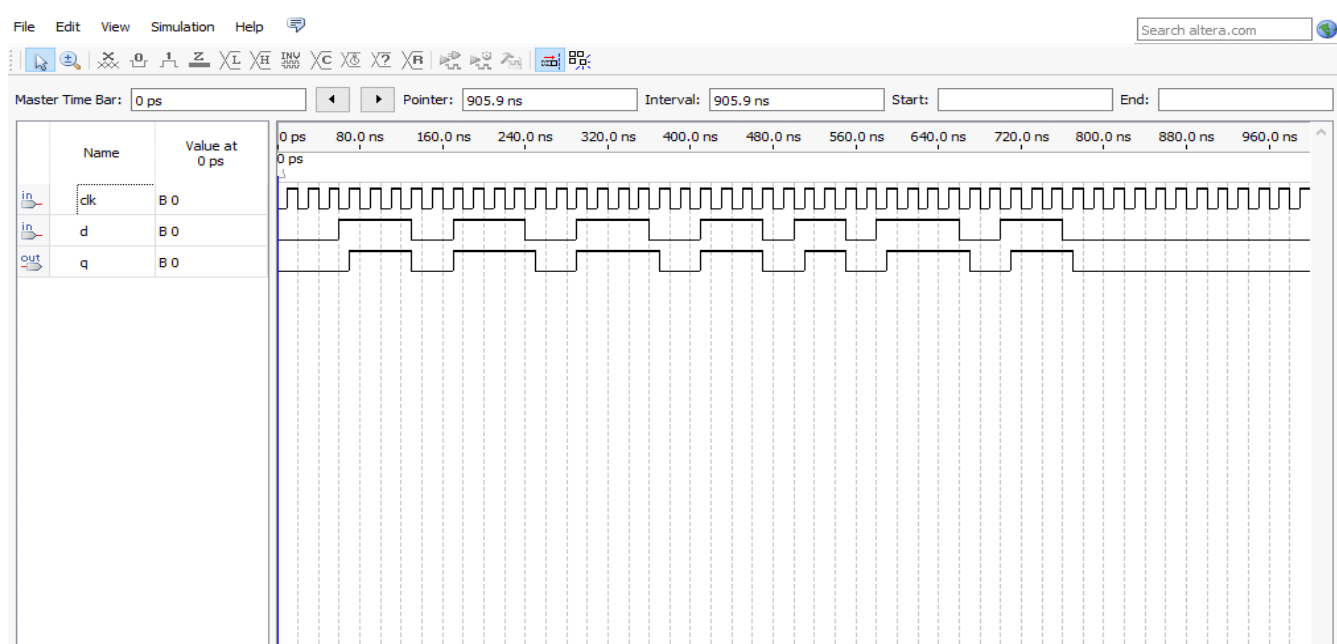
Le processus est déclenché uniquement lors d'un front montant de l'horloge, et la sortie est mise à jour

avec la valeur de l'entrée D. Il n'y a pas de condition de réinitialisation dans ce code, ce qui signifie que la sortie conserve sa valeur précédente même si l'horloge est désactivée.

- **Bloc schématique**



- **Chronogramme**



le chronogramme décrit le fonctionnement de la bascule **D**. A chaque front montant de l'horloge fixée à **20ns** on peut constater que la sortie **Q** recopie l'état de l'entrée **D** de la bascule comme le décrit sa Table de vérité.

- **Assignment**

afin de pouvoir visualiser le comportement physique du circuit nous avons procédé a une assignation de Pin des entrées et sorties avec ceux de la carte FPGA. Les affectations choisies sont:

<b>Clk</b>	<b>Input</b>	<b>PIN_R22</b>
<b>d</b>	<b>Input</b>	<b>PIN_L2</b>
<b>q</b>	<b>Output</b>	<b>PIN_R20</b>

- **Programmation**

Report

Report not available

Groups

Report

Tasks

Run Anal

Early Pin

Early

Run

Expo

Change \

Show

Show

Show

### Top View - Wire Bond

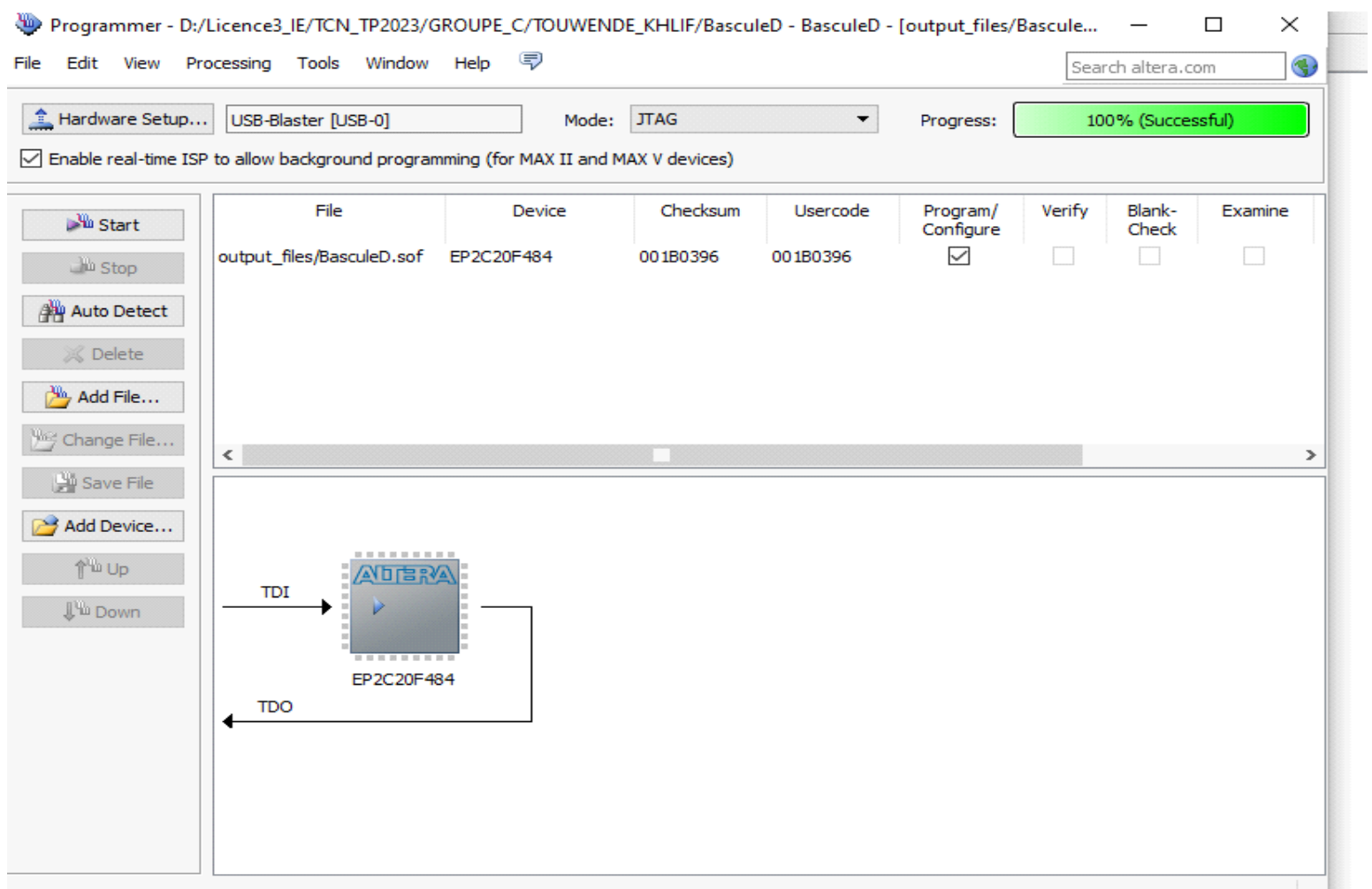
## Cyclone II - EP2C20F484C7

Named: \*

Edit:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
clk	Input	PIN_R22	6	B6_N0	PIN_R22	3.3-V LV...default		24mA (default)	
d	Input	PIN_L2	2	B2_N1	PIN_L2	3.3-V LV...default		24mA (default)	
q	Output	PIN_R20	6	B6_N0	PIN_R20	3.3-V LV...default		24mA (default)	
<<new node>>									

une fois l'assignation des pins réalisée, après s'être assuré que la configuration est bien faite sur **Cyclone II EP2C20F484**, nous effectuons à nouveau une compilation afin que les modifications apportées puissent être en compte par la carte. A présent lançons le programme à l'aide de la fonction **Programmer**.



On constate que la programmation s'est réalisée avec succès les manipulation sur la carte nous laisse constater le comportement physique du circuit comme le décrit son chronogramme.

2. Dans le cas où on utilise la configuration bouclant la sortie ( $/Q = \text{NOT } Q$ ) sur l'entrée D on réalise une inversion de la sortie dans ce cas la fréquence de la sortie Q sera égale à la moitié de la fréquence du signal clk. Il est important de noter que cette configuration peut conduire à une instabilité si les temps de propagation de la porte logique ne sont pas correctement synchronisés avec la fréquence de l'horloge clk.

- **Code VHDL de la bascule à inverseur**

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY basculeDboucle IS
6  PORT (
7      clk: IN std_logic;
8      set : IN std_logic;
9      reset: IN std_logic;
10     q: BUFFER std_logic;
11     h:OUT std_logic --cette sortie nous permet de visualiser l'horloge clk à l'aide d'une LED
12 );
13 end basculeDboucle ;
14
15 ARCHITECTURE archi OF basculeDboucle IS
16
17 BEGIN
18     PROCESS(clk,reset)
19     BEGIN
20         if(reset='1') then
21             q<='0';
22         else
23             if RISING_EDGE(clk) THEN
24                 if (set ='1') then
25                     q<='1';
26                 else
27
28                     q<= not q;
29                 end if;
30             END IF;
31         end if;
32         h<=clk; --h recopie l'horloge principale clk
33     end PROCESS;
34
35     END archi;
36
37

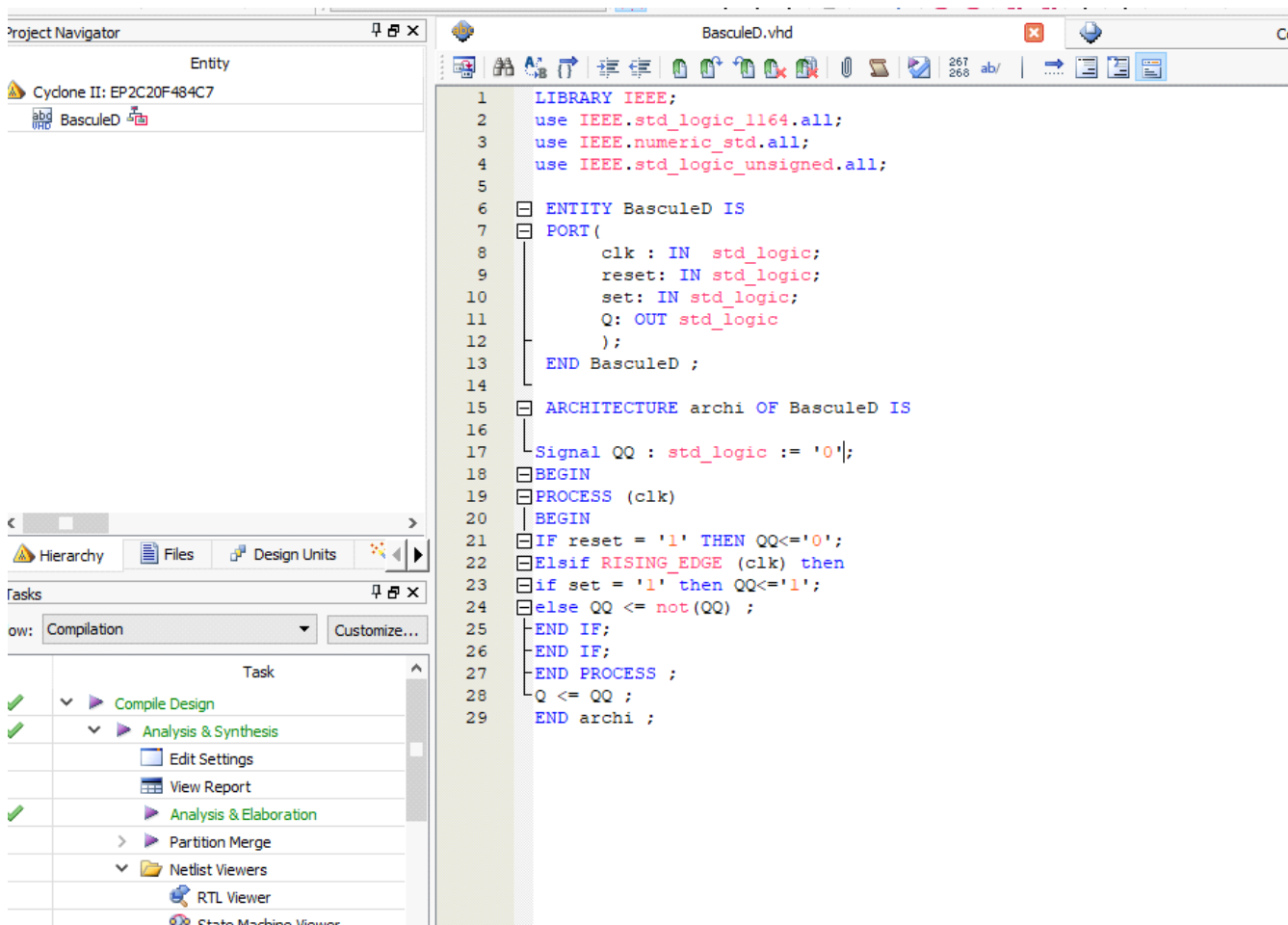
```

la bascule D est un composant utile dans les circuits numériques pour stocker et transférer des bits de données pendant une période de temps donnée. Elle est couramment utilisée dans les compteurs binaires, les mémoires RAM, la synchronisation de signaux et le stockage temporaire des données. Ses avantages sont sa simplicité et sa facilité d'utilisation, mais elle a aussi des limites, notamment sa capacité limitée à stocker des données et sa nécessité de signaux de synchronisation pour fonctionner correctement.

## Projet 2:    Réalisation d'un Compteur avec Reset asynchrone

pour la suite on boucle ( $Q = \text{not}(Q)$ ) sur l'entrée D et on ajoute les entrées **Reset (clear)** et **set (Preset)**. la fréquence de la sortie Q est égale à la moitié de la fréquence de clk.

### ● Description VHDL

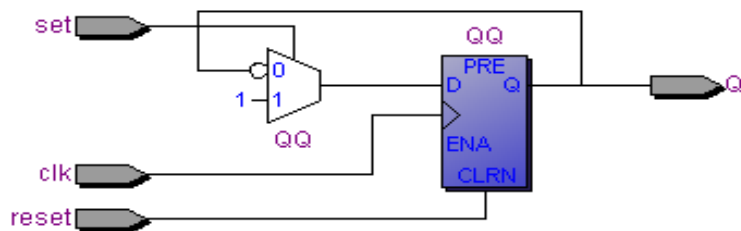


Le comportement du circuit est défini à travers un processus qui est exécuté à chaque front montant de l'horloge. Si reset est activé, la sortie Q est remise à 0. Sinon, si set est activé, la sortie Q est mise à

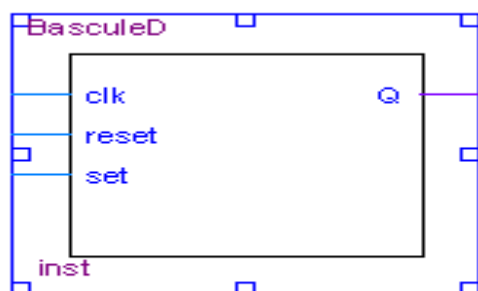
1. Si aucun des deux signaux n'est activé, la sortie Q est inversée.

## ● Circuit RTL

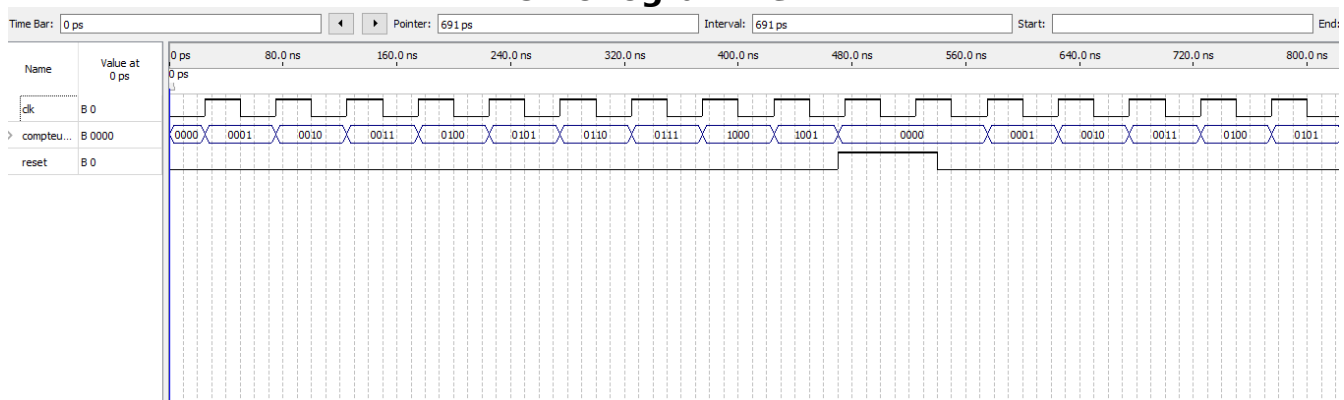




## ● Bloc schématique



## ● Chronogramme



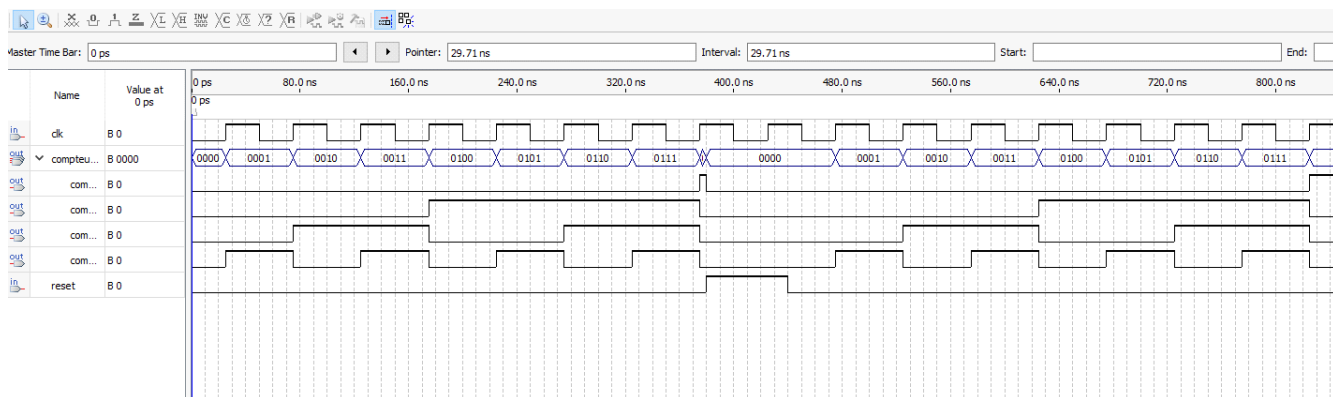
la fonction réalisée est un compteur 4

- **Assignation des PINS**

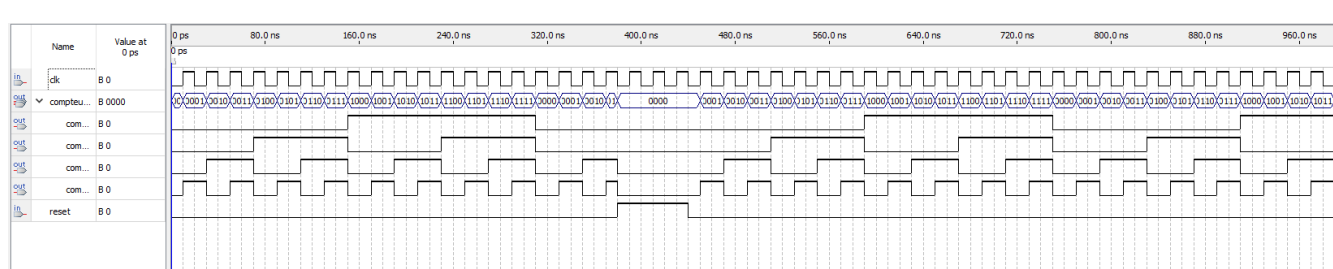
Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
clk	Input	PN_R22	6	B6_N0	PN_M1	3.3-V LV..default		24mA (default)	
h	Output	PN_R20	6	B6_N0	PN_N1	3.3-V LV..default		24mA (default)	
q[0]	Output	PN_V21	6	B6_N1	PN_T2	3.3-V LV..default		24mA (default)	
q[2]	Output	PN_V22	6	B6_N1	PN_R2	3.3-V LV..default		24mA (default)	
q[1]	Output	PN_U21	6	B6_N1	PN_T1	3.3-V LV..default		24mA (default)	
q[0]	Output	PN_U22	6	B6_N1	PN_R1	3.3-V LV..default		24mA (default)	
reset	Input	PN_L22	5	B5_N1	PN_M2	3.3-V LV..default		24mA (default)	
<-new node>>									

la fréquence de chaque bit dans le compteur est la moitié de la fréquence du bit précédent, ce qui signifie que q[0] change deux fois plus souvent que q[1], q[1] change deux fois plus souvent que q[2], et ainsi de suite

- **chronogramme illustratif pour une période d'horloge de 50ns**



- **Chronogramme pour T= 20ns**



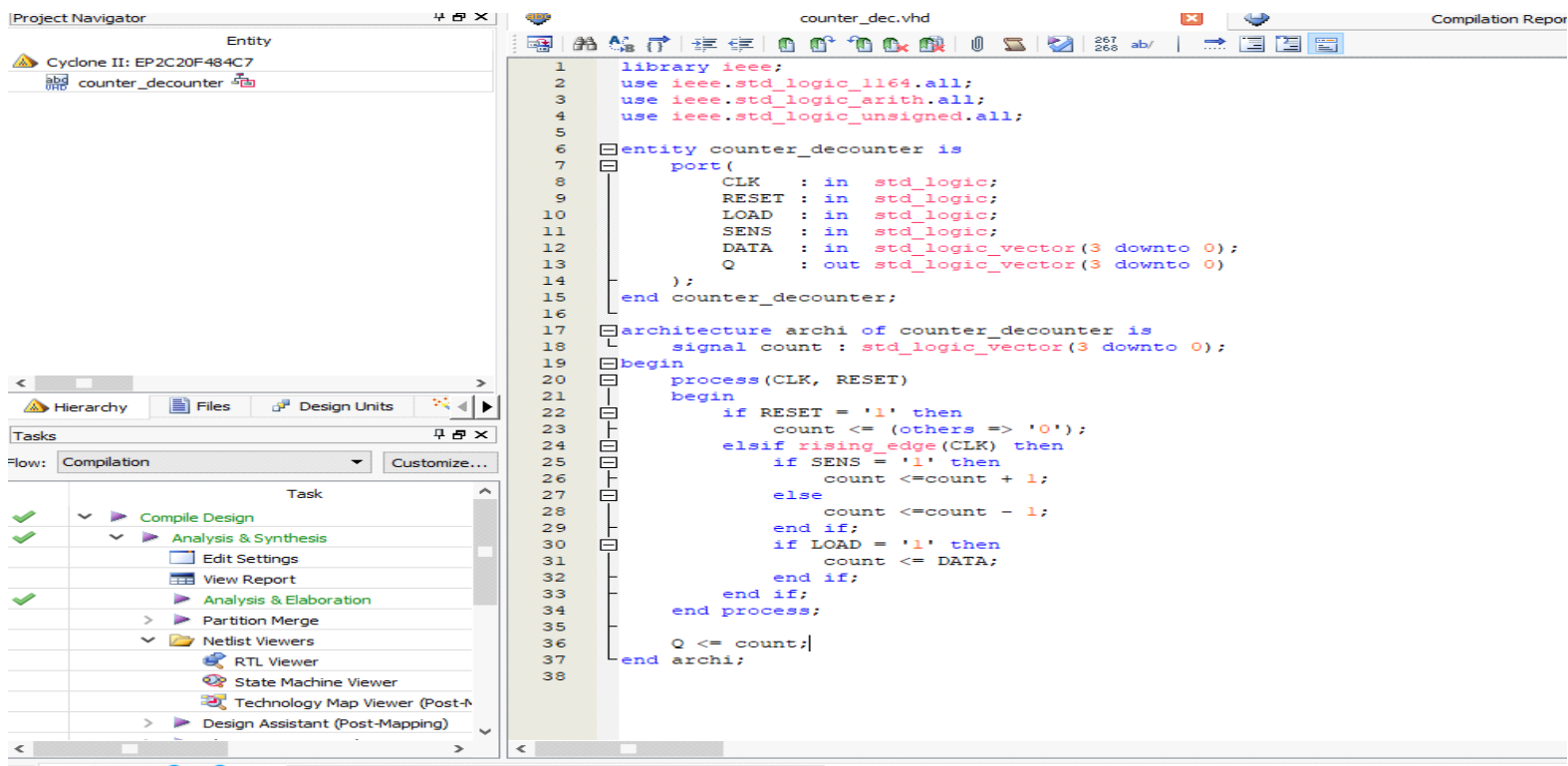
Nous remarquerons qu' avec T=20ns que le cycle de comptage est plus visible

En résumé le compteur permet de compter les impulsions d'entrée et de réinitialiser la valeur du compteur à tout moment grâce à un signal de reset asynchrone. Cependant il est important de noter que

l'utilisation d'un signal de reset asynchrone peut entraîner des problèmes de synchronisation et de timing.

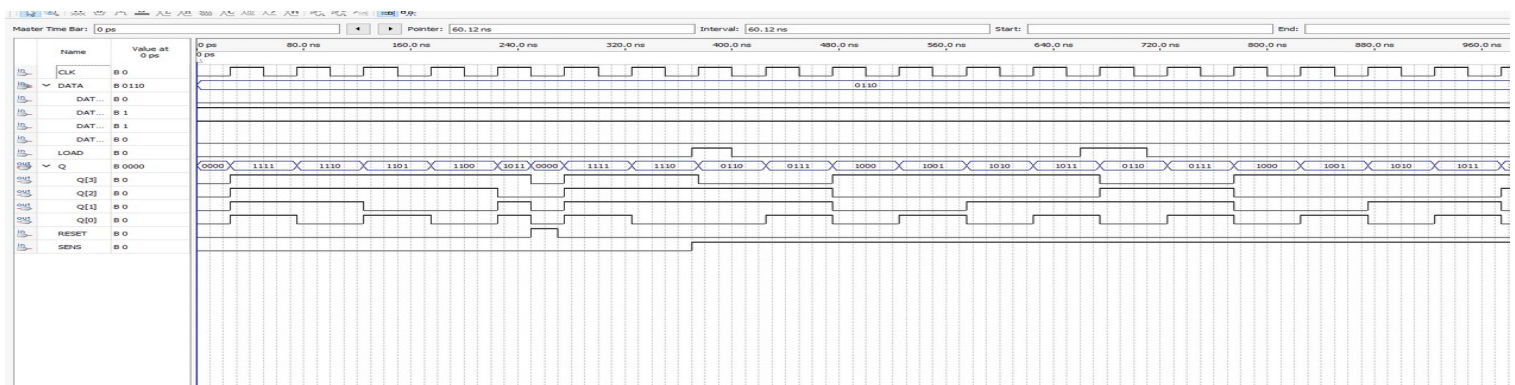
## conception et réalisation d'un compteur/décompteur

- code VHDL



```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity counter_decounter is
7     port(
8         CLK      : in  std_logic;
9         RESET    : in  std_logic;
10        LOAD     : in  std_logic;
11        SENS     : in  std_logic;
12        DATA    : in  std_logic_vector(3 downto 0);
13        Q        : out std_logic_vector(3 downto 0)
14    );
15 end counter_decounter;
16
17 architecture archi of counter_decounter is
18     signal count : std_logic_vector(3 downto 0);
19 begin
20     process(CLK, RESET)
21     begin
22         if RESET = '1' then
23             count <= (others => '0');
24         elsif rising_edge(CLK) then
25             if SENS = '1' then
26                 count <= count + 1;
27             else
28                 count <= count - 1;
29             end if;
30             if LOAD = '1' then
31                 count <= DATA;
32             end if;
33         end if;
34     end process;
35
36     Q <= count;
37 end archi;
```

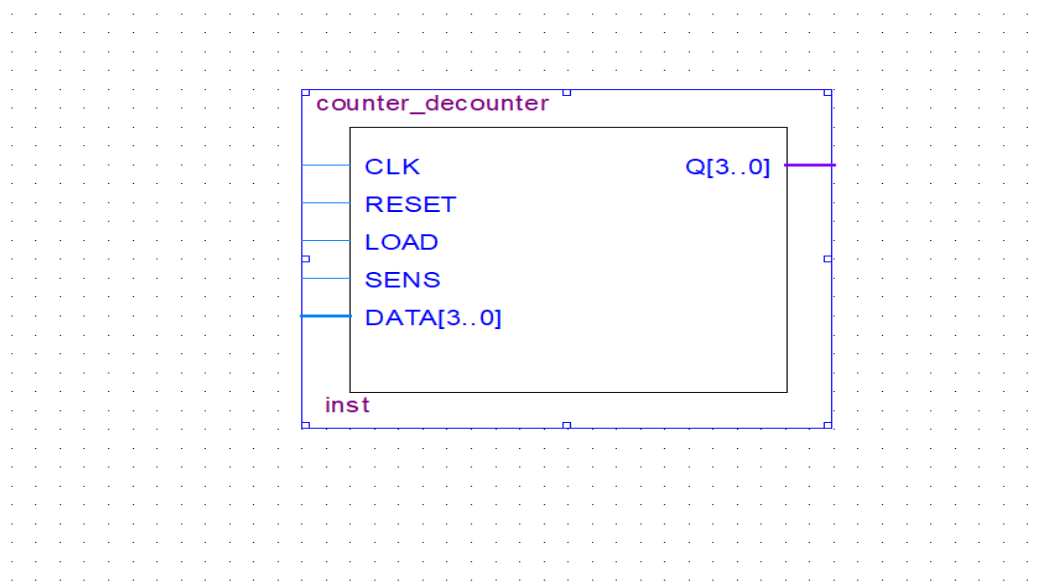
- Chronogramme



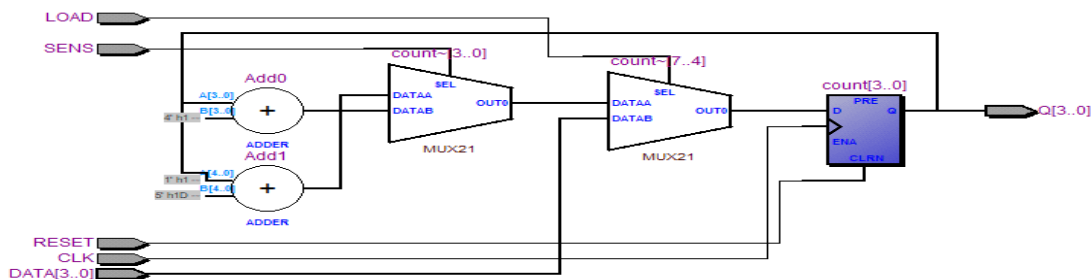
- Assignment des PINS

Node Name	Direction	Location	I/O Bank	VREF Group	Filter Location	I/O Standard	Reserved	Current Strength	Differential Pair
clk	Input	PIN_R22	6	86_N0	PIN_R22	3.3-V LV..default		24mA (default)	
DATA[3]	Input	PIN_W12	7	87_N1	PIN_W12	3.3-V LV..default		24mA (default)	
DATA[2]	Input	PIN_V12	7	87_N1	PIN_V12	3.3-V LV..default		24mA (default)	
DATA[1]	Input	PIN_M22	6	86_N0	PIN_M22	3.3-V LV..default		24mA (default)	
DATA[0]	Input	PIN_L21	5	85_N1	PIN_L21	3.3-V LV..default		24mA (default)	
h	Output	PIN_R20	6	86_N0	PIN_R20	3.3-V LV..default		24mA (default)	
LOAD	Input	PIN_U12	8	88_N0	PIN_U12	3.3-V LV..default		24mA (default)	
q[3]	Output	PIN_V21	6	86_N1	PIN_V21	3.3-V LV..default		24mA (default)	
q[2]	Output	PIN_V22	6	86_N1	PIN_V22	3.3-V LV..default		24mA (default)	
q[1]	Output	PIN_U21	6	86_N1	PIN_U21	3.3-V LV..default		24mA (default)	
q[0]	Output	PIN_U22	6	86_N1	PIN_U22	3.3-V LV..default		24mA (default)	
reset	Input	PIN_L22	5	85_N1	PIN_L22	3.3-V LV..default		24mA (default)	
SENS	Input	PIN_U11	8	88_N0	PIN_U11	3.3-V LV..default		24mA (default)	
<-new node-->									

- block schematic



- Schéma RTL



- **Décodeur 7 segments**

- **code VHDL**

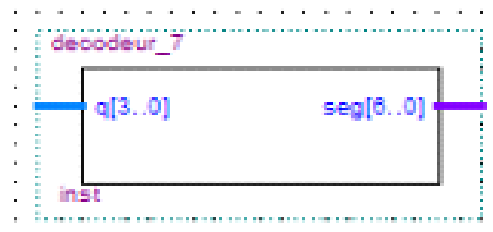
```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY decodeur_7 IS
6  PORT (
7      q: IN std_logic_vector(3 DOWNTO 0);
8      seg: out std_logic_vector(6 downto 0)
9  );
10 end decodeur_7;
11
12 ARCHITECTURE archi OF decodeur_7 is
13 BEGIN
14     process(q)
15     BEGIN
16         CASE q is
17             when "0000" => seg <="1000000";
18             when "0001" => seg <="1111001";
19             when "0010" => seg <="0100100";
20             when "0011" => seg <="0110000";
21             when "0100" => seg <="0011001";
22             when "0101" => seg <="0010010";
23             when "0110" => seg <="0000010";
24             when "0111" => seg <="1111000";
25             when "1000" => seg <="0000000";
26             when "1001" => seg <="0010000";
27             when "1010" => seg <="0001000";
28             when "1011" => seg <="0000011";
29             when "1100" => seg <="1000110";
30             when "1101" => seg <="0100001";
31             when "1110" => seg <="0000110";
32             when "1111" => seg <="0001110";
33         end CASE;
34     end process;
35 end archi;

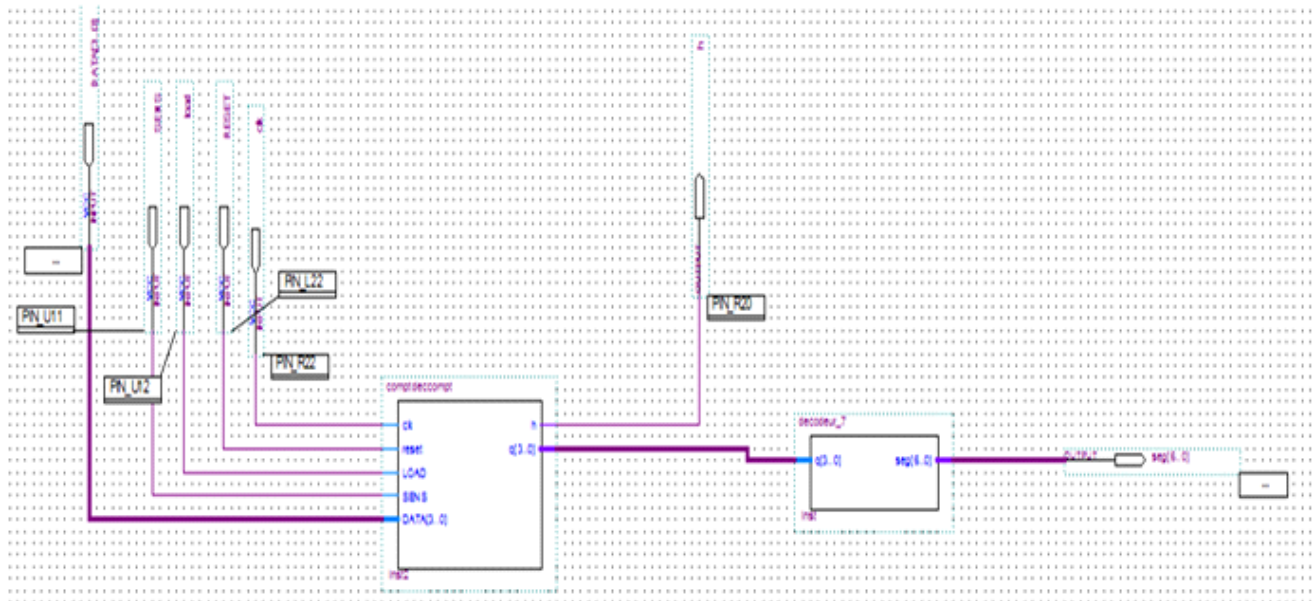
```

16

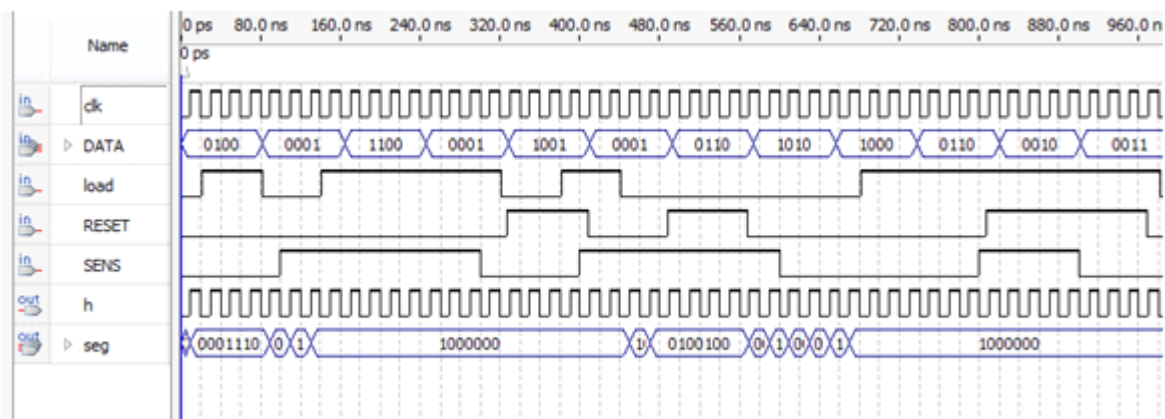
- **Bloc schématique**



- **Graphique du décodeur associé à l’Afficheur**



- **Résultat de simulation limité à 9**



- **Code VHDL (Comptage limité à 9)**

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY comptdec compt IS
6  PORT (
7      clk: IN std_logic;
8      reset: IN std_logic;
9      LOAD: IN std_logic;
10     SENS: IN std_logic;
11     DATA: IN unsigned(3 downto 0);
12     h: out std_logic;
13     q: out unsigned(3 downto 0)
14 );
15 end comptdec compt;
16
17 ARCHITECTURE archi OF comptdec compt is
18     signal qint: unsigned(3 downto 0);
19 BEGIN
20     PROCESS(clk, reset)
21     BEGIN
22         if(reset='1') then
23             q<=(others => '0');
24         elsif RISING_EDGE(clk) THEN
25
26             if(LOAD='1') then
27                 q<=DATA;
28
29             elsif (SENS='1') then
30                 qint<= qint + 1;
31                 if(qint="1001") then
32                     qint<= "0000";
33                 end if;
34             elsif (SENS='0') then
35                 qint<= qint - 1;
36                 if(qint="0000") then
37                     qint<= "1001";
38                 end if;
39             end if;
40         end if;
41         q<=qint;
42         h<=clk; --h recopie l'horloge principale clk
43     end PROCESS;
44
45 END archi;

```

## Projet 3 : Régistre à décalage

- code du registre à décalage 4 bits



Project Navigator: Entity  
Cyclone II: EP2C20F484C7  
registre\_4

Tasks: Compilation

Task List:

- Compile Design
- Analysis & Synthesis
  - Edit Settings
  - View Report
- Analysis & Elaboration
  - Partition Merge
    - View Report
  - Design Partition Planner
- Netlist Viewers
  - RTL Viewer
  - State Machine Viewer

registre\_4.vhd

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY registre_4 IS
6  PORT (
7      clk:IN std_logic;
8      d_in:IN std_logic;
9      vall:IN std_logic_vector(3 downto 0);
10     d_out:OUT std_logic;
11     match:OUT std_logic
12 );
13 END registre_4;
14
15 ARCHITECTURE archi OF registre_4 is
16     signal reg :std_logic_vector(3 downto 0);
17 BEGIN
18     PROCESS(clk)
19     BEGIN
20         IF rising_edge(clk) THEN
21             reg(3 downto 1)<=reg(2 downto 0);--décalage vers le bit du poids fort
22             reg(0)<=d_in;
23         END IF;
24     END PROCESS;
25     d_out <=reg(3);
26     match <='1'when vall =reg else '0';
27 END archi;
28
29

```

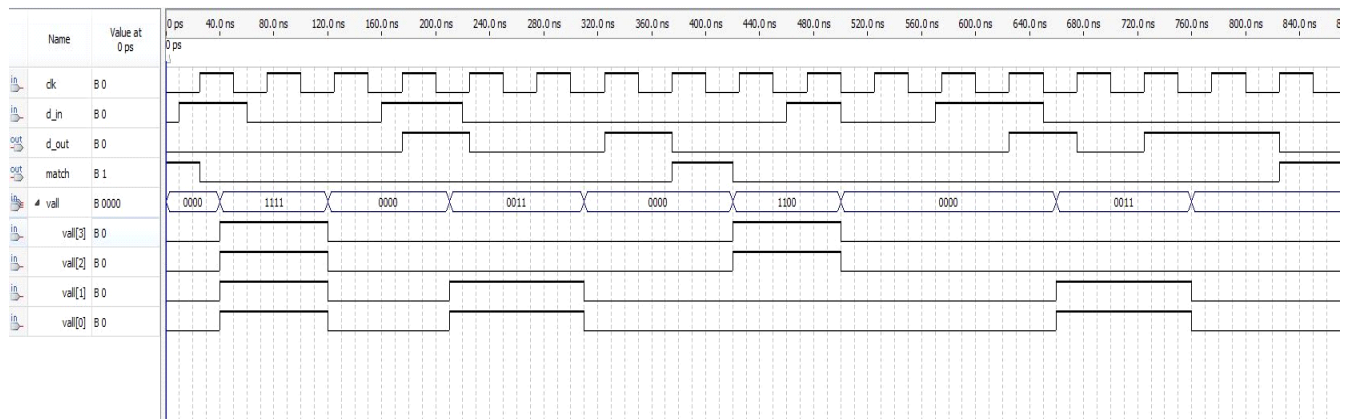
Messages:

```

*****
Running Quartus II 64-Bit EDA Netlist Writer
Command: quartus_eda --read_settings_files=off --write_settings_files=off registre_4 -c registre_4
204019 Generated file registre_4.vo in folder "D:/LicenceIE/TCN_2223/GrC/simulation/modelsim/" for EDA simulation tool
Quartus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 11 warnings

```

## ● chronogramme du registre





- **Assignation des PINS**

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
Horloge	Input	PIN_R22	6	B6_N0	PIN_R22	3.3-V LV...default)		24mA (default)	
IN	Input	PIN_L22	5	B5_N1	PIN_L22	3.3-V LV...default)		24mA (default)	
out[3]	Output	PIN_Y19	6	B6_N1	PIN_Y19	3.3-V LV...default)		24mA (default)	
out[2]	Output	PIN_U19	6	B6_N1	PIN_U19	3.3-V LV...default)		24mA (default)	
out[1]	Output	PIN_R19	6	B6_N0	PIN_R19	3.3-V LV...default)		24mA (default)	
out[0]	Output	PIN_R20	6	B6_N0	PIN_R20	3.3-V LV...default)		24mA (default)	
<new node>>									

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

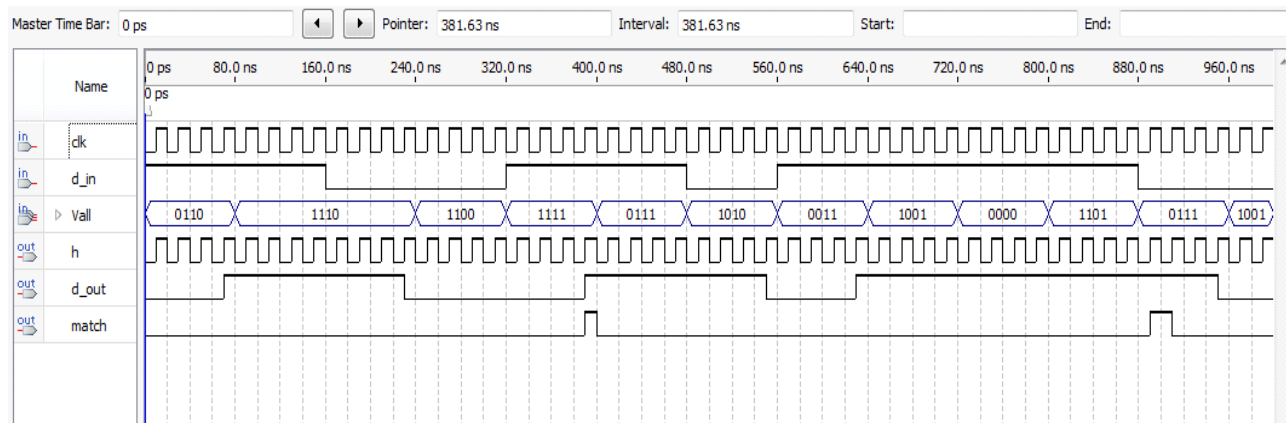
- **code VHDL**

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY Registre IS
6  PORT (
7      clk: IN std_logic;
8      d_in: IN std_logic;
9      Vall: IN std_logic_vector(3 downto 0);
10     h: OUT std_logic;
11     d_out: OUT std_logic;
12     match: OUT std_logic
13 );
14 end Registre;
15
16
17
18
19 ARCHITECTURE archi OF Registre is
20     signal reg: std_logic_vector(3 downto 0);
21 BEGIN
22     process (CLK)
23     BEGIN
24         IF rising_edge (CLK) then
25             reg(2 downto 0) <= reg(3 downto 1);
26             reg(3) <= d_in;
27         END IF;
28     END PROCESS;
29     d_out <= reg(0);
30     match <= '1' when Vall = reg else '0';
31     h <= clk;
32 END archi;

```

- **Le résultat de la simulation :**



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

Associons un registre de décalage série, un diviseur de fréquence par 4 et un registre parallèle 4 bits dans une description graphique.

- Diviseur de fréquence par 4

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY compteur2 IS
6  PORT (
7      clk: IN std_logic;
8      q: out std_logic
9  );
10 end compteur2 ;
11
12 ARCHITECTURE archi OF compteur2 is
13     signal q_int: unsigned(1 downto 0);
14 BEGIN
15     PROCESS(clk)
16     BEGIN
17         if RISING_EDGE(clk) THEN
18             q_int<= q_int + 1;
19         end if;
20     end PROCESS;
21     q<=q_int(1);
22
23 END archi;

```

- Registre de décalage parallèle

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY Registre_parallele IS
6  PORT (
7      clk: IN std_logic;
8      d_in: IN std_logic_vector(3 downto 0);
9      d_out: OUT std_logic_vector(3 downto 0)
10 );
11 end Registre_parallele;
12 ARCHITECTURE archi OF Registre_parallele is
13
14 BEGIN
15     process(clk)
16     BEGIN
17         IF rising_edge(clk) then
18             d_out <= d_in;
19         END IF;
20     END PROCESS;
21
22 END archi;

```

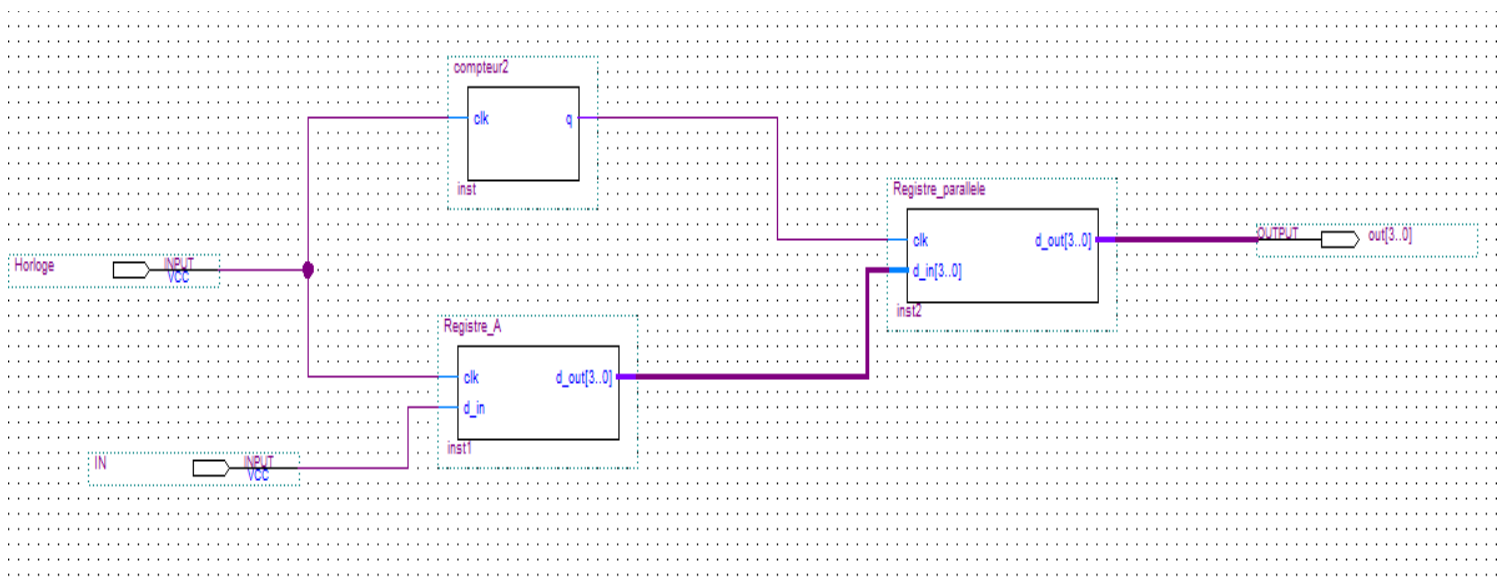
- **Registre de décalage série**

```

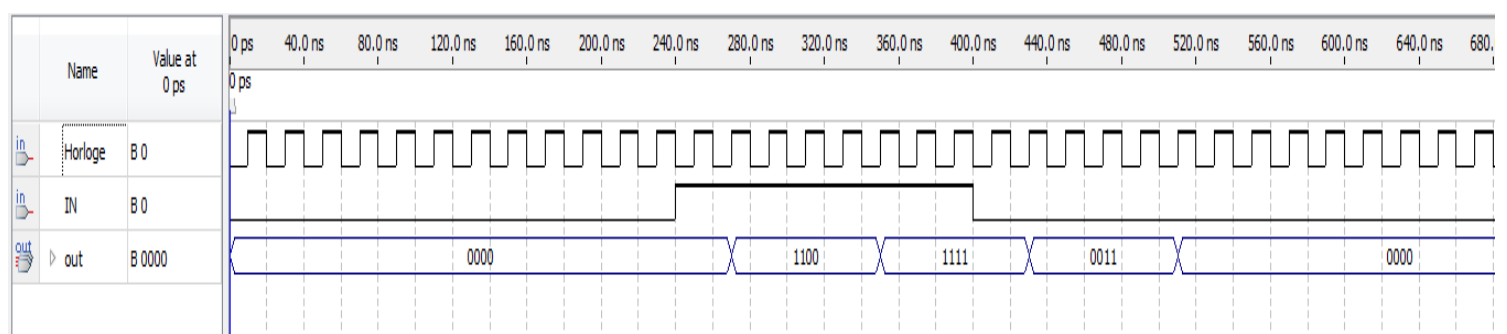
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY Registre_A IS
6  PORT (
7      clk: IN std_logic;
8      d_in: IN std_logic;
9      d_out: OUT std_logic_vector(3 downto 0)
10 );
11 end Registre_A;
12 ARCHITECTURE archi OF Registre_A is
13     signal reg : std_logic_vector(3 downto 0);
14 BEGIN
15     process(clk)
16     BEGIN
17         IF rising_edge(clk) then
18             reg(2 downto 0) <= reg(3 downto 1);
19             reg(3) <= d_in;
20         END IF;
21     END PROCESS;
22     d_out <= reg;
23 END archi;

```

- **La description graphique de l'ensemble**



- **La simulation fonctionnelle de l'ensemble**



## **PROJET 4: Machines d'état synchrone**

Dans cette partie le code permet de détecter le début et la fin d'une impulsion s de durée quelconque par machine Moore.

Le système est synchronisé par une horloge clk. Le début de l'impulsion est indiqué par le signal sm et la fin par le signal sd.

Le code VHDL qui permet la la synthèse du circuit détecteur de fronts à l'aide de :

### **1.) Trois process**

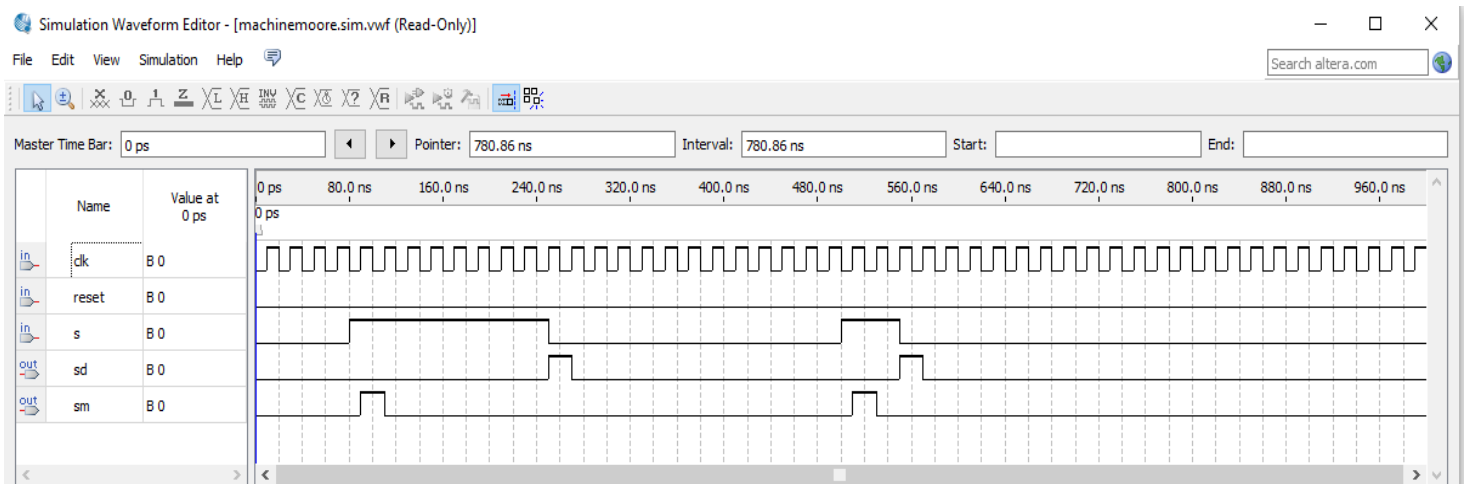
- Son code VHDL :

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  use IEEE.std_logic_unsigned.all;
5
6  ENTITY machinemooore is
7  PORT (
8      clk : in std_logic;
9      reset: in std_logic;
10     s: in std_logic;
11     sm: out std_logic;
12     sd: out std_logic
13 );
14 END machinemooore;
15
16 ARCHITECTURE archi of machinemooore is
17 type Etat is (Etat0,Etat1,Etat2,Etat3);
18 Signal Etat_present,Etat_futur:Etat:=Etat0;
19
20 BEGIN
21
22     REGISTRE : PROCESS(clk,reset)
23     BEGIN
24         IF reset='1' THEN
25             Etat_present<=Etat0;
26
27         ELSIF RISING_EDGE(clk) THEN
28             Etat_present<=Etat_futur;
29         end if;
30     END PROCESS REGISTRE;
31
32     Combinatoire_etats: PROCESS(Etat_present,Etat_futur,s)
33     BEGIN
34         case Etat_present is
35             When Etat0 => if s='1' then
36                 Etat_futur<=Etat1;
37             else
38                 Etat_futur<=Etat0;
39             end if;
40             When Etat1 =>
41                 Etat_futur<=Etat2;
42             When Etat2 => if s='1' then
43                 Etat_futur<=Etat2;
44             else
45                 Etat_futur<=Etat3;
46             end if;
47             When Etat3 =>
48                 Etat_futur<=Etat0;
49         end case;
50
51     END PROCESS Combinatoire_etats;
52
53     Combinatoire_soties: PROCESS(Etat_present,Etat_futur)
54     BEGIN
55         case Etat_present is
56             When Etat0 => sm <= '0' ;sd <= '0';
57             When Etat1 => sm <='1' ;sd <= '0';
58             When Etat2 => sd <= '0' ;sm <= '0';
59             When Etat3 => sd <='1'; sm <= '0';
60         end case;
61     END PROCESS Combinatoire_soties;
62
63 END archi;

```

- **Le résultat de sa simulation :**



## 2.) Deux process

- **Son code VHDL :**

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY machine_moore IS
6  PORT (
7      clk: IN std_logic;
8      s: IN std_logic;
9      reset: in std_logic;
10     sm: out std_logic;
11     sd: out std_logic
12 );
13 end machine_moore;
14
15 ARCHITECTURE archi of machine_moore is
16     type Etat is (Etat0,Etat1,Etat2,Etat3);
17     Signal Etat_present,Etat_futur:Etat:=Etat0;
18 BEGIN
19
20     REGISTRE : PROCESS(clk,reset)
21     BEGIN
22         IF reset='1' THEN
23             Etat_present<=Etat0;
24
25         ELSIF RISING_EDGE(clk) THEN
26             Etat_present<=Etat_futur;
27         end if;
28     END PROCESS REGISTRE;
29
30     Combinatoire_etats: PROCESS(Etat_present,s)
31     BEGIN
32         case Etat_present is
33             When Etat0 => sm <= '0' ;sd <= '0';
34                 if s='1' then
35                     Etat_futur<=Etat1;
36                 else
37                     Etat_futur<=Etat0;
38                 end if;
39             When Etat1 => sm <='1' ;sd <= '0';
40                 Etat_futur<=Etat2;
41
42             When Etat2 => sd <= '0' ;sm <= '0';
43                 if s='1' then
44                     Etat_futur<=Etat2;
45                 else
46                     Etat_futur<=Etat3;
47                 end if;
48             When Etat3 => sd <='1' ; sm <= '0';
49                 Etat_futur<=Etat0;
50
51         end case;
52
53     END PROCESS Combinatoire_etats;
54
55 END archi;
56
57
58
59
60

```





```

41
42
43
44
45
46
47
48
49
50
51
52
53

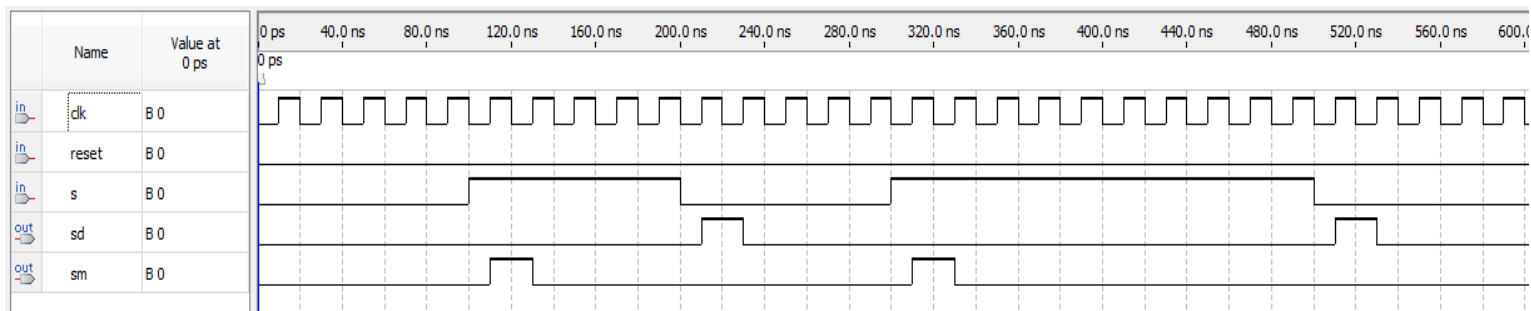
else
    Etat <= Etat3;
    sd <='1'; sm <= '0';
end if;
When Etat3 => Etat <= Etat0;
sm <= '0' ;sd <= '0';

end case;
end if;
END PROCESS;

END archi;

```

- Le résultat de sa simulation



## PROJET 5: Mini-Projet : Convertisseur binaire – décimal 4 bits

Dans cette partie nous voulons concevoir un convertisseur qui affiche directement la valeur décimale d'un nombre binaire V sur 4 bits au moyen de deux afficheurs 7 segments HEX0 (unité) et HEX1(décimal).

### Analyse de l'architecture du convertisseur

#### Circuit A

Le circuit A est un **soustracteur** qui joue un rôle très important dans le convertisseur.

## Circuit B

Le circuit B est un **décodeur 7 segment de 1 bit en entrées et 7 bits en sortie** .

Si l'entrée unitaire A est à **0** la sortie prend la valeur : **S = 1000000** pour afficher 0 sur l'afficheur HEX1, par contre si l'entrée unitaire A est à **1** la sortie prend la valeur : **S = 1111100** pour afficher 0 sur l'afficheur HEX1.

### Le code VHDL du circuit A

```
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  use ieee.std_logic_unsigned.all;
5
6  ENTITY circuitA IS
7  PORT (
8      y: IN std_logic_vector(2 downto 0);
9      s: out std_logic_vector(2 downto 0)
10 );
11 end circuitA;
12
13 ARCHITECTURE archi OF circuitA IS
14 BEGIN
15     s(2 downto 0) <= y(2 downto 0) - "10";
16 end archi;
```

### Le code VHDL du circuit B

```
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY CircuitB IS
6  PORT (
7      q: IN std_logic;
8      seg: out std_logic_vector(6 downto 0)
9  );
10 end CircuitB;
11
12 ARCHITECTURE archi OF CircuitB IS
13 BEGIN
14     process (q)
15     BEGIN
16         CASE q IS
17             WHEN '0' => seg <= "1000000";
18             WHEN '1' => seg <= "1111100";
19         END CASE;
20     END process;
21 end archi;
```

## Le code VHDL du multiplexeur 2 vers 1

```
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY multiplexeur2 IS
6  PORT (
7      E: IN std_logic_vector(1 DOWNTO 0);
8      c: IN std_logic;
9      s: out std_logic
10 );
11 end multiplexeur2 ;
12
13 ARCHITECTURE archi OF multiplexeur2 IS
14 BEGIN
15     s <= E(0) WHEN c='0' ELSE
16         E(1);
17 end archi;
```

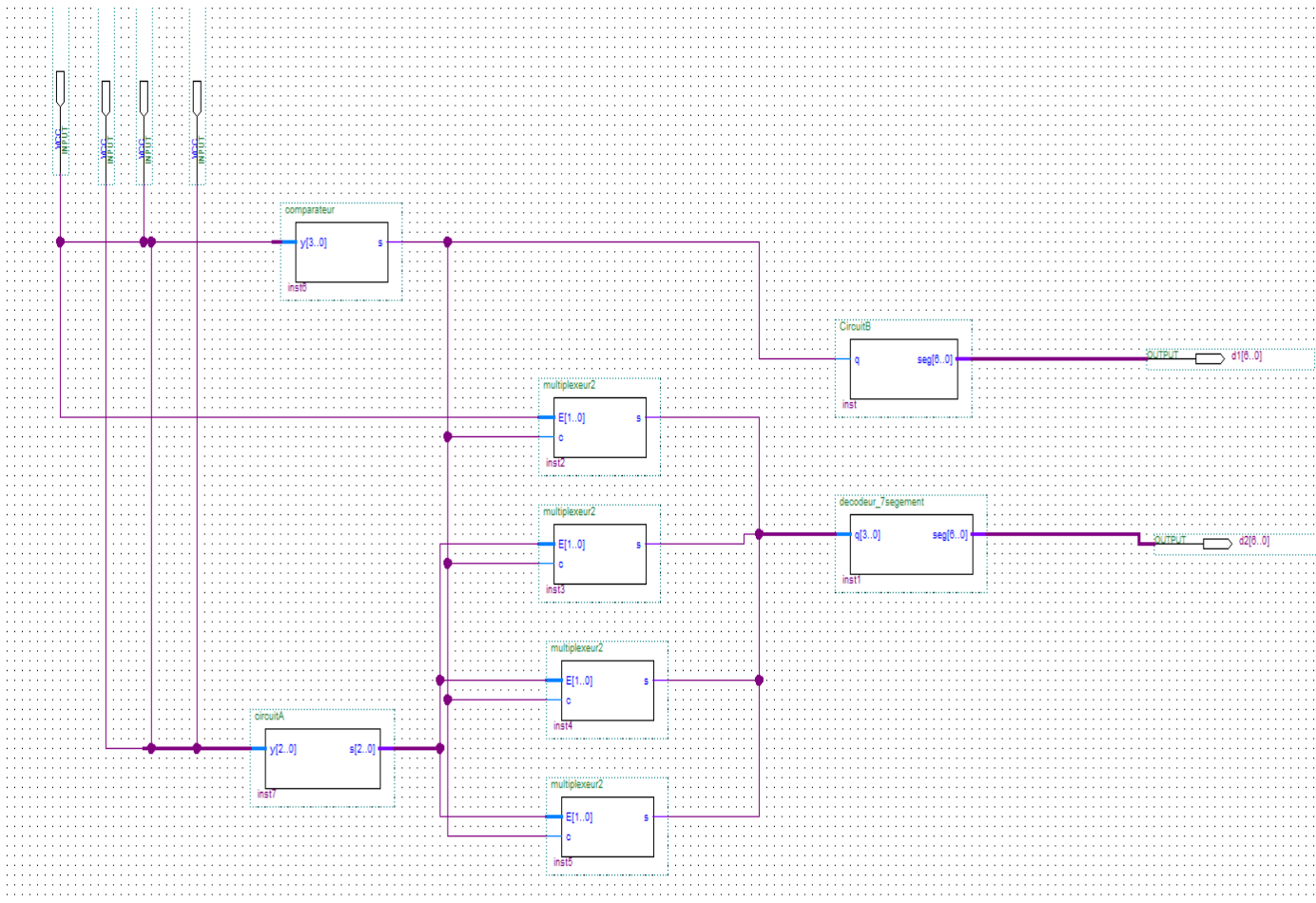
## Le code VHDL du comparateur

```
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  ENTITY compareur IS
6  PORT (
7      y: IN std_logic_vector(3 DOWNTO 0);
8      s: out std_logic
9  );
10 end compareur;
11
12 ARCHITECTURE archi OF compareur IS
13     signal k : std_logic_vector(3 DOWNTO 0);
14     BEGIN
15         k <= "1010";
16
17         s <= '0' WHEN (y > k) ELSE
18             '1';
19     end archi;
```

## Le code VHDL du décodeur

```
1  | LIBRARY IEEE;
2  | use IEEE.std_logic_1164.all;
3  | use IEEE.numeric_std.all;
4
5  | ENTITY decodeur_7segment IS
6  | PORT (
7  |     q: IN std_logic_vector(3 DOWNTO 0);
8  |     seg: out std_logic_vector(6 downto 0)
9  | );
10 | end decodeur_7segment;
11
12 | ARCHITECTURE archi OF decodeur_7segment is
13 | BEGIN
14 |   process (q)
15 |   BEGIN
16 |     CASE q is
17 |       when "0000" => seg <="1000000";
18 |       when "0001" => seg <="1111001";
19 |       when "0010" => seg <="0100100";
20 |       when "0011" => seg <="0110000";
21 |       when "0100" => seg <="0011001";
22 |       when "0101" => seg <="0010010";
23 |       when others => seg <= "XXXXXXX";
24 |     end CASE;
25 |   end process;
26 | end archi;
```

## Le schéma bloc du convertisseur



## Conclusion

En conclusion, ce rapport en technologie des circuits numériques a permis de mettre en évidence l'importance de cette discipline dans le développement de systèmes électronique modernes. Nous avons examiné en détail les différents types de circuits numériques, leurs caractéristiques et leur utilisation dans des application pratiques telles que les processeurs, les mémoires et les circuits de communication.

Enfin, ce rapport a souligné l'importance de la conception de circuits numérique efficaces et robuste pour garantir des performances optimales et une fiabilité à long termes. La technologie des circuits numérique est donc une discipline clé dans l'industrie électronique, et son importance ne fera que croître à mesure que de nouvelles applications émergent.