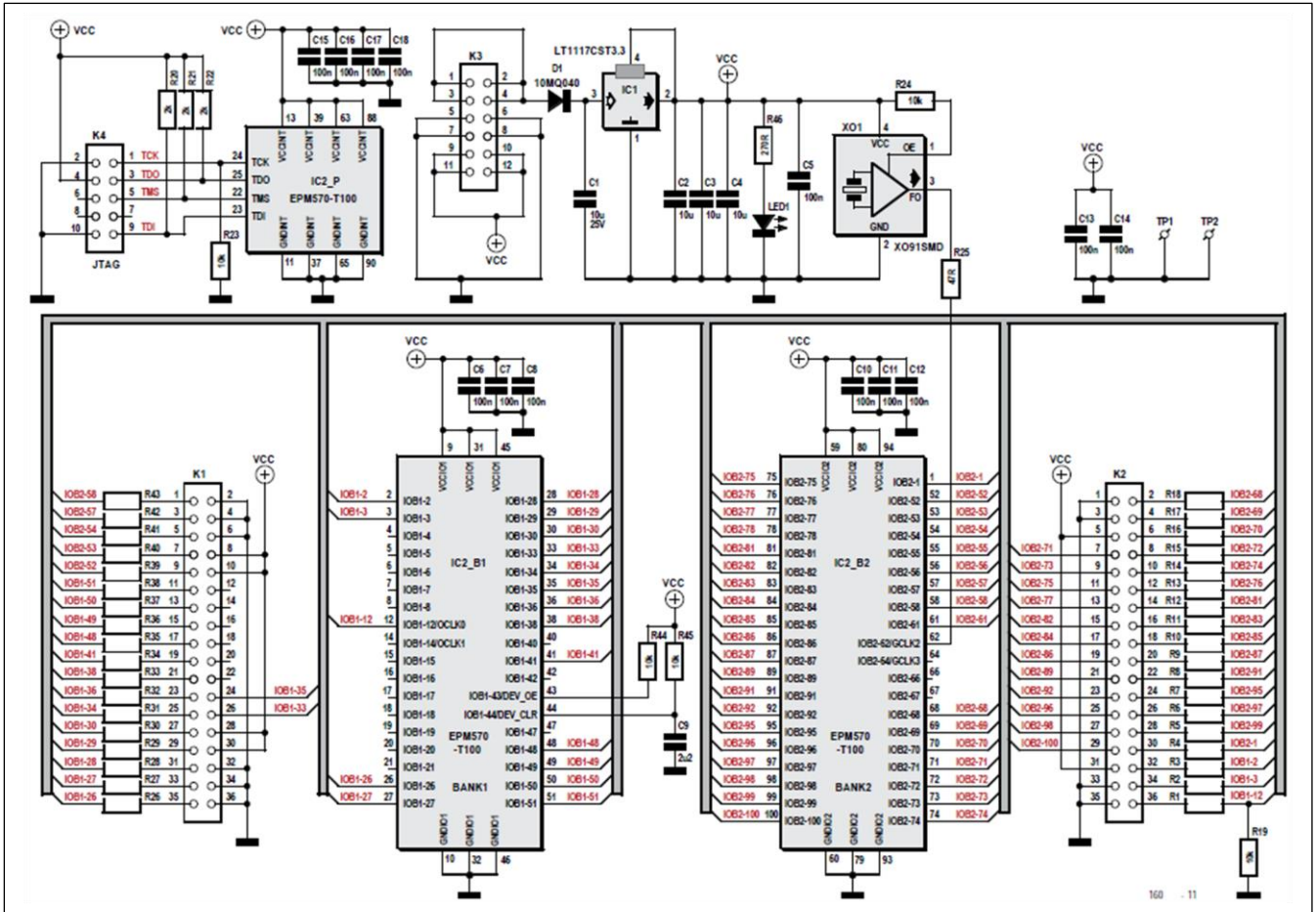
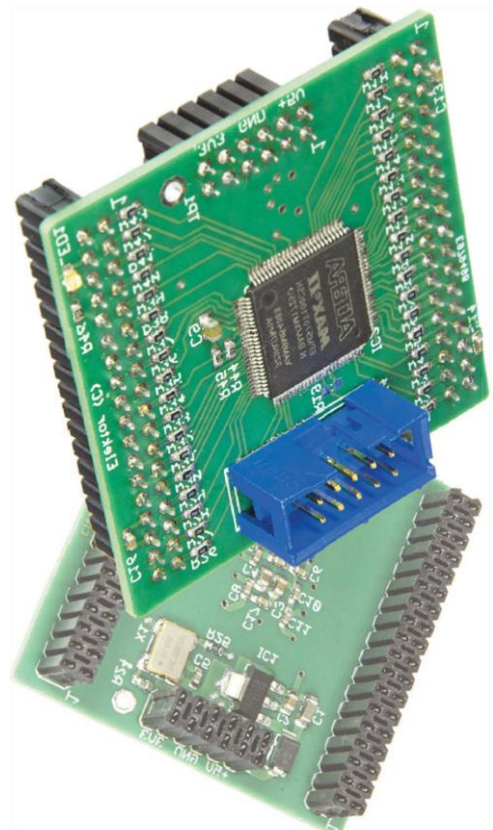
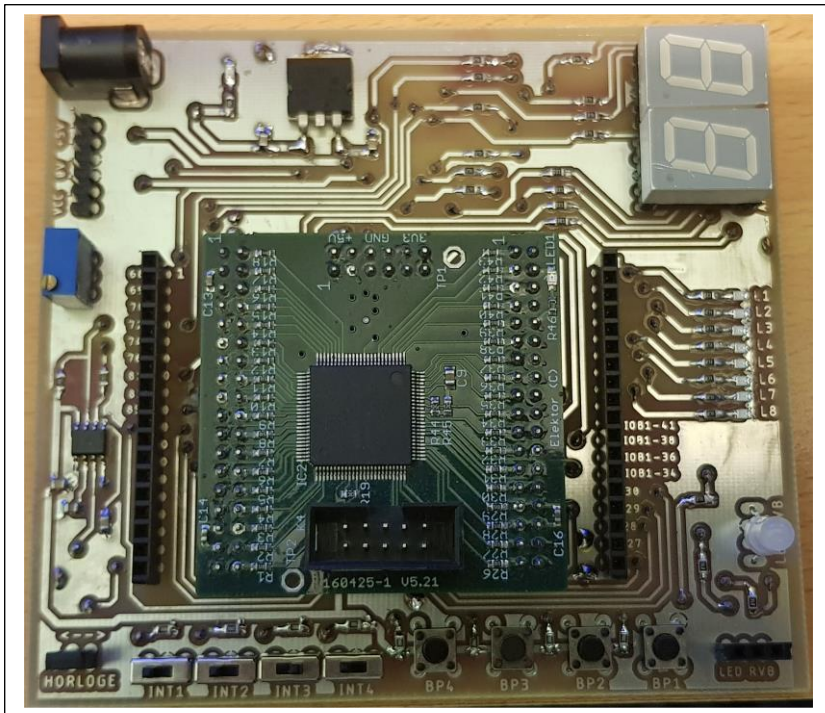


## INTERFACE CPLD ELEKTOR.

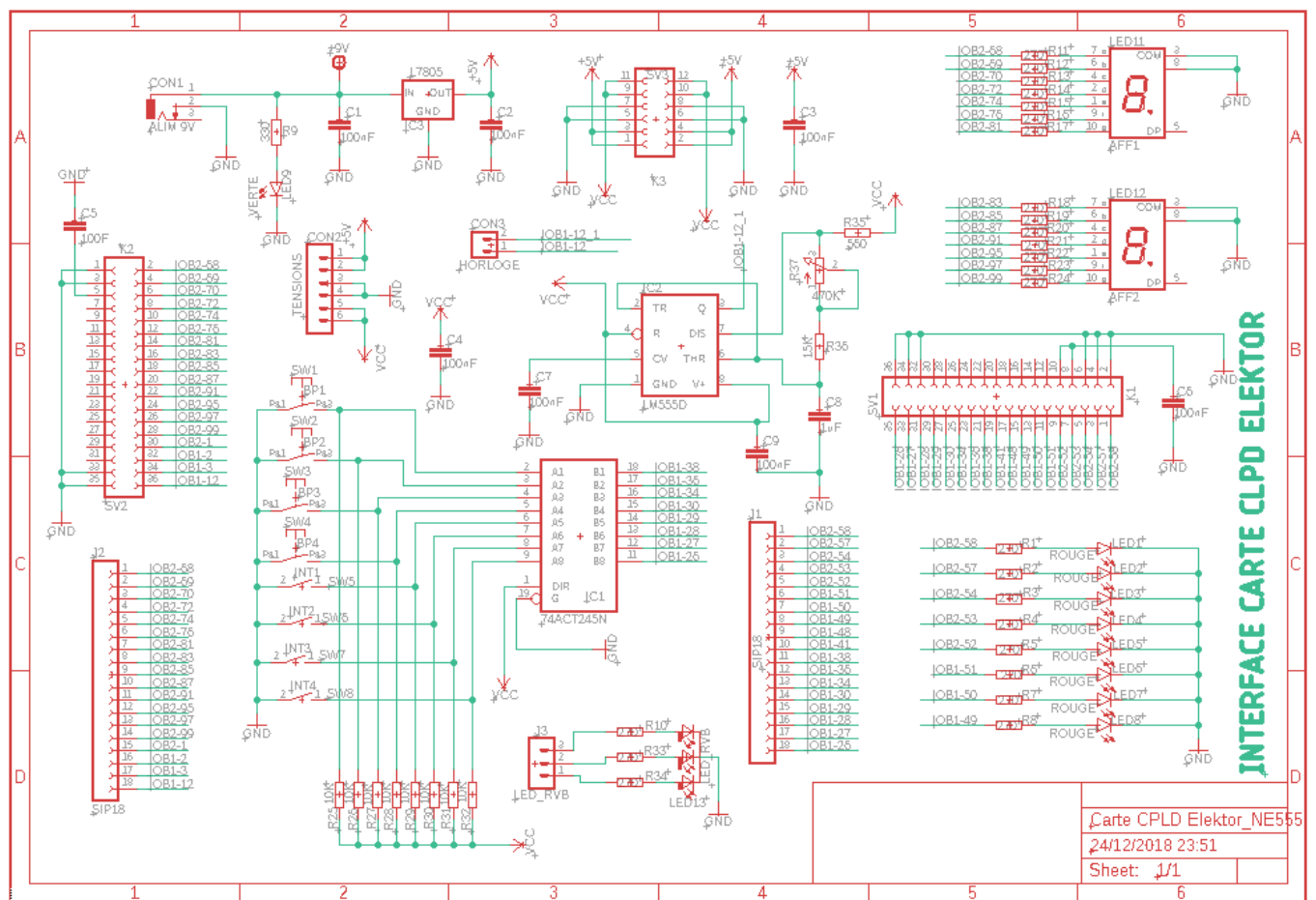
### Carte CPLD ELEKTOR – Schéma structurel.



### INTERFACE CARTE CPLD.



## Schéma structurel.



## Coté composants.

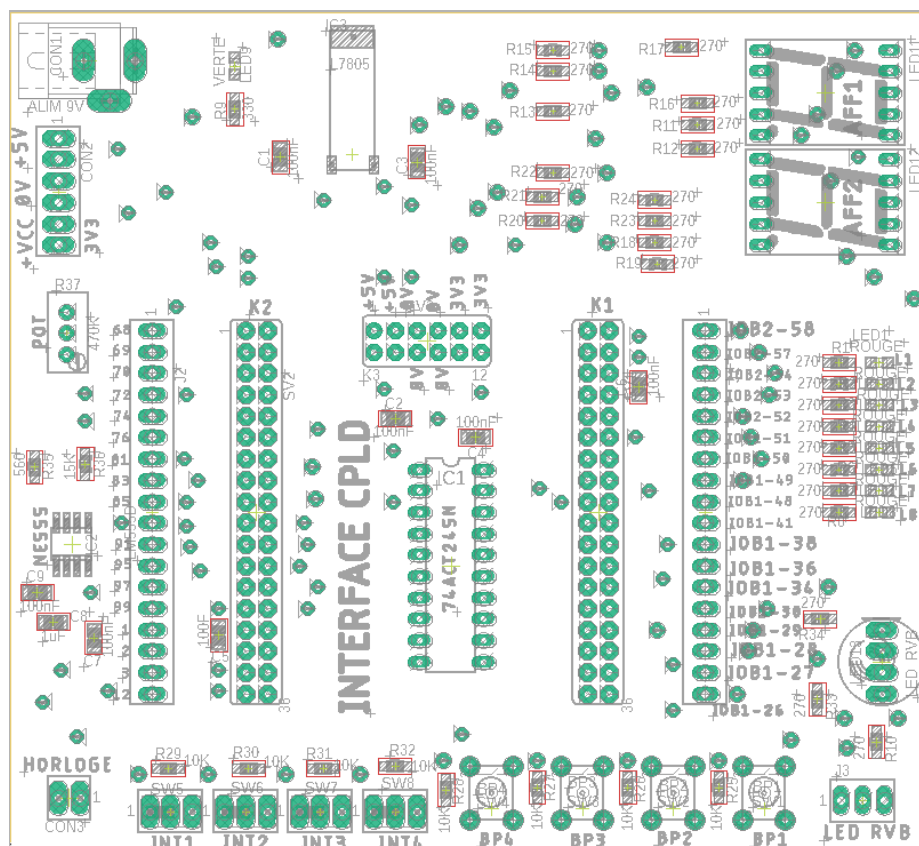
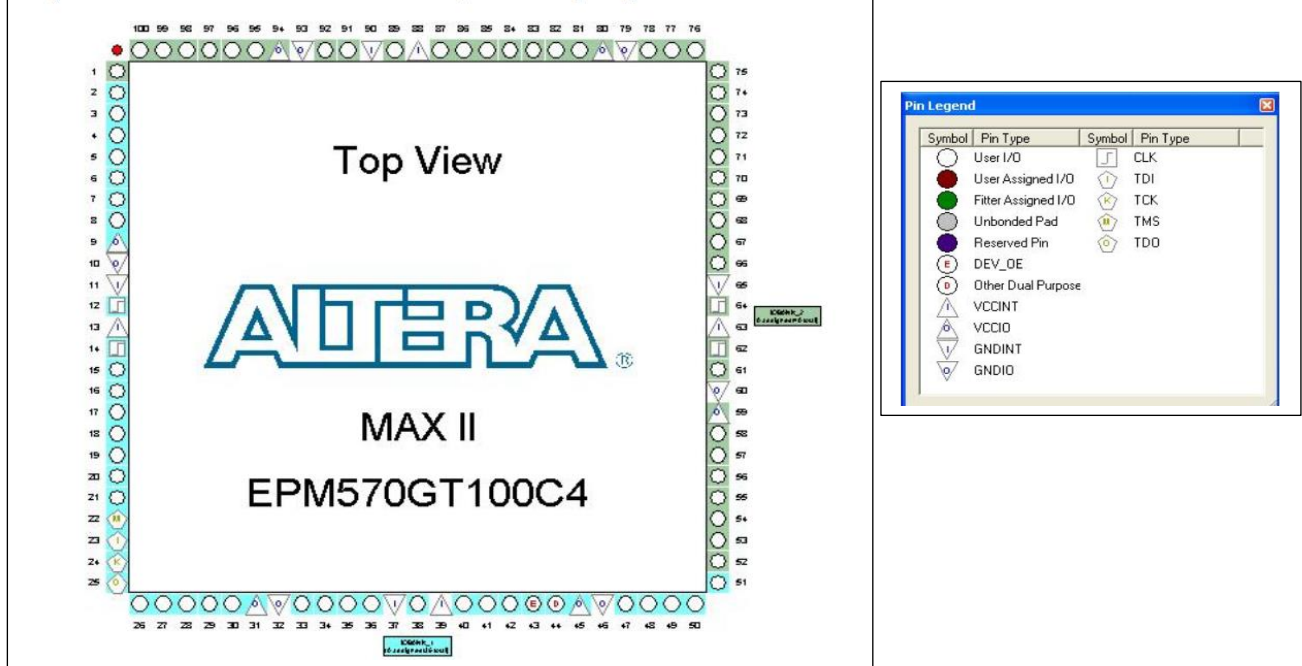


Figure 1. MAX II EPM570 / EPM570G T100 Device Top View Package Diagram and Bank Information



### Affectation des broches.

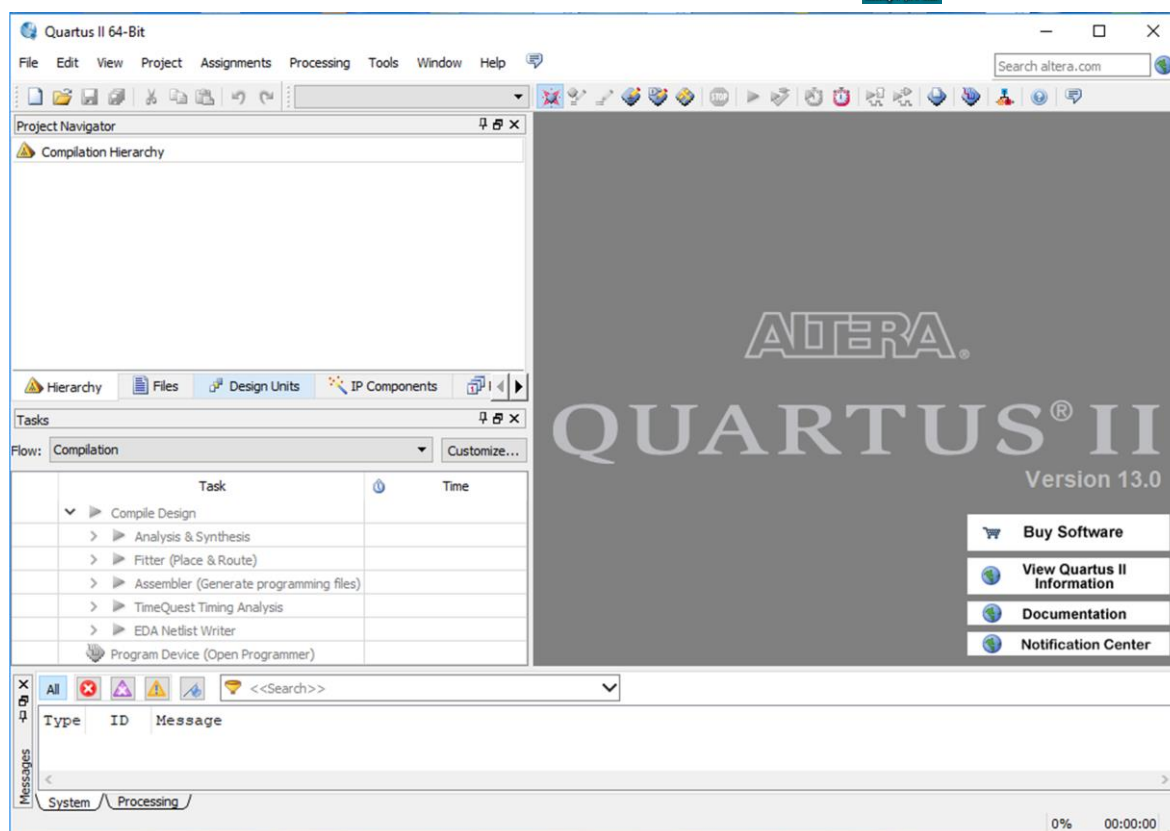
N° Broche	Nom	Affectation	Remarque
<b>Afficheurs 7-Segments : AFF1 et AFF2</b>			
68	IOB2-68	AFF1 – Commande segment "a"	
69	IOB2-69	AFF1 – Commande segment "b"	
70	IOB2-70	AFF1 – Commande segment "c"	
72	IOB2-72	AFF1 – Commande segment "d"	
74	IOB2-74	AFF1 – Commande segment "e"	
76	IOB2-76	AFF1 – Commande segment "f"	
81	IOB2-81	AFF1 – Commande segment "g"	
83	IOB2-83	AFF2 – Commande segment "a"	
85	IOB2-85	AFF2 – Commande segment "b"	
87	IOB2-87	AFF2 – Commande segment "c"	
91	IOB2-91	AFF2 – Commande segment "d"	
95	IOB2-95	AFF2 – Commande segment "e"	
97	IOB2-97	AFF2 – Commande segment "f"	
99	IOB2-99	AFF2 – Commande segment "g"	
<b>Les Leds L1, L2, L3, L4, L5, L6, L7 et L8.</b>			
58	IOB2-58	Commande diode led L1	
57	IOB2-57	Commande diode led L2	
54	IOB2-54	Commande diode led L3	
53	IOB2-53	Commande diode led L4	
52	IOB2-52	Commande diode led L5	
51	IOB2-51	Commande diode led L6	
50	IOB2-50	Commande diode led L7	
49	IOB2-49	Commande diode led L8	



Les boutons poussoirs BP1, BP2, BP3 et BP4.			
38	IOB1-38	Lecture état du bouton poussoir <b>BP1</b>	
36	IOB1-36	Lecture état du bouton poussoir <b>BP2</b>	
34	IOB1-34	Lecture état du bouton poussoir <b>BP3</b>	
30	IOB1-30	Lecture état du bouton poussoir <b>BP4</b>	
Les interrupteurs INT1, INT2, INT3 et INT4.			
29	IOB1-29	Lecture état de l'interrupteur <b>INT1</b>	
28	IOB1-28	Lecture état de l'interrupteur <b>INT2</b>	
27	IOB1-27	Lecture état de l'interrupteur <b>INT3</b>	
26	IOB1-26	Lecture état de l'interrupteur <b>INT4</b>	
Les broches non affectées.			
1	IOB2-1		
2	IOB1-2		
3	IOB1-3		
12	IOB1-12	Disponible pour une horloge externe	IOB1-12/OCLK0
41	IOB1-41		
48	IOB1-48		
Les signaux d'horloge			
62	IOB2-62/GCLK2	Horloge CPLD 40 MHz	Oscillateur à quartz XO1
12	IOB1-12	Disponible pour une horloge externe	

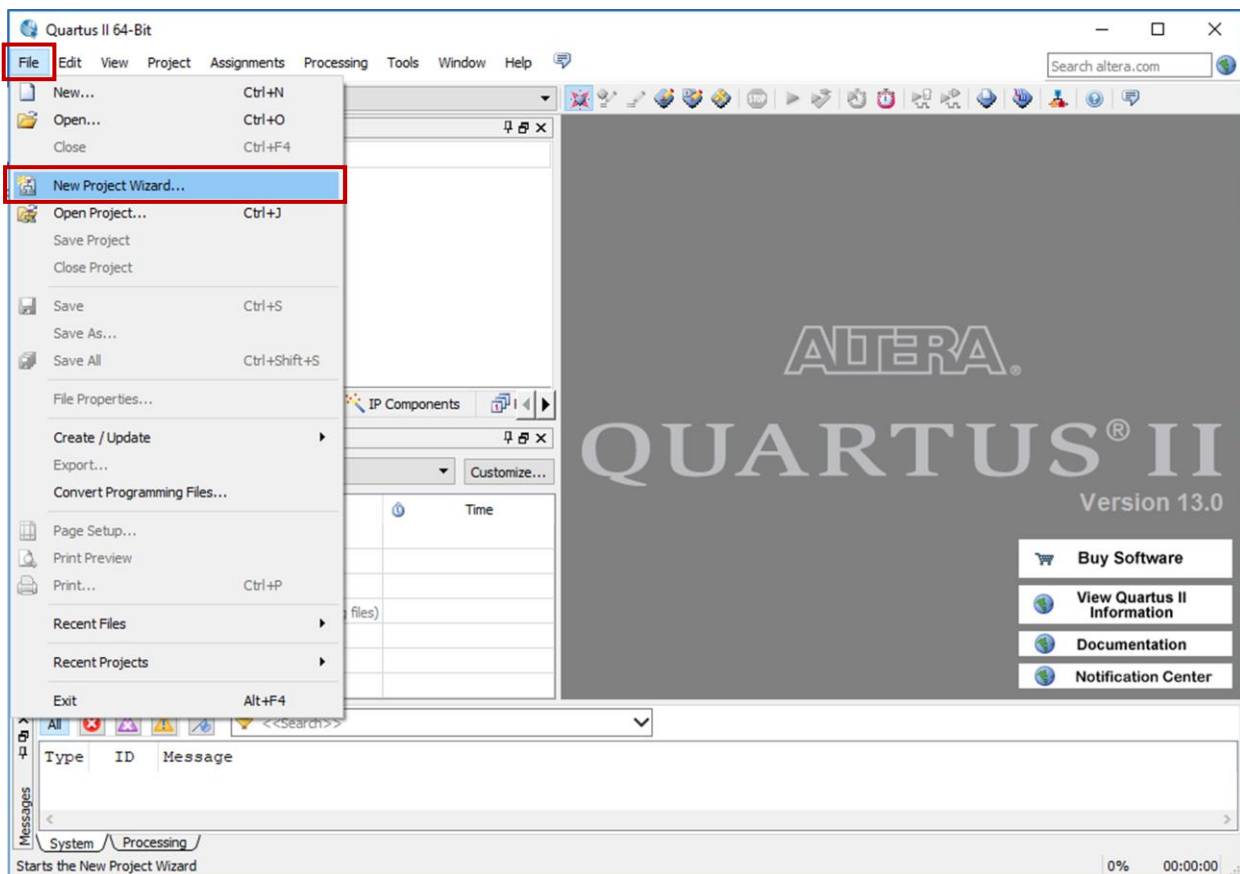
## **QUARTUS : Outil de développement et de téléversement de programmes pour les puces Altera.**

☞ Lancer le logiciel en cliquant sur l'icône "QUARTUS II"

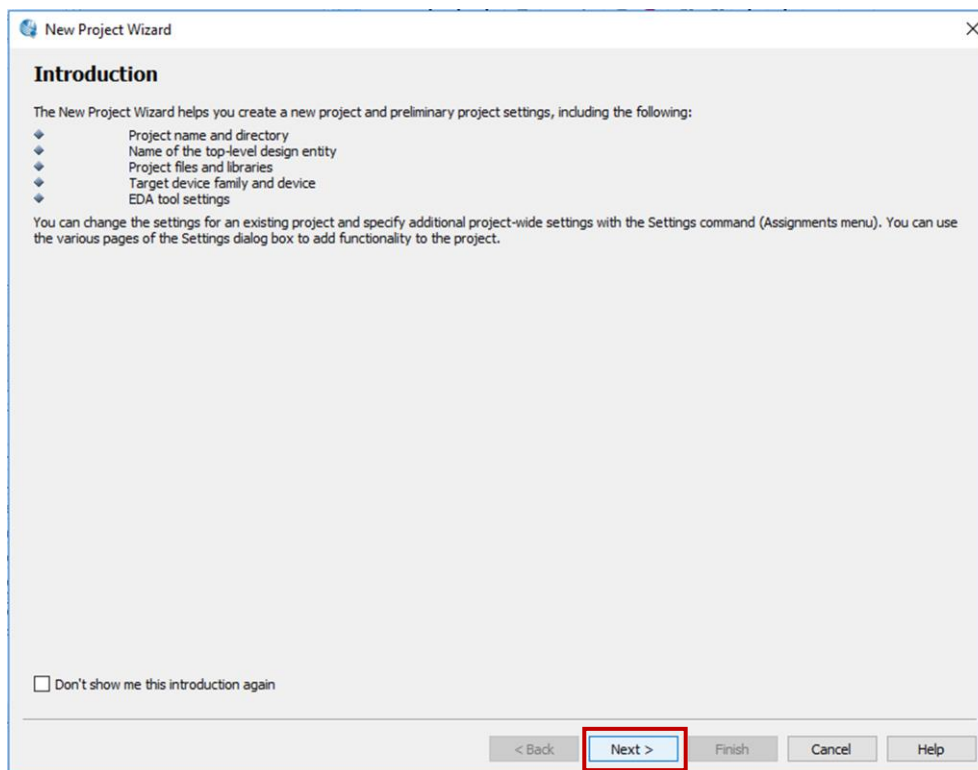


*... Choisir ensuite l'assistant pour commencer un nouveau programme.*





☞ *Suivre les cinq étapes proposées par l'assistant.*



New Project Wizard

### Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

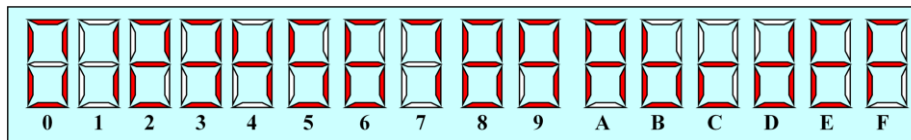
What is the name of this project?

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

*... Nom et emplacement du projet.*

**Exemple : Programmation en VHDL de la logique de décodage d'un afficheur 7-Segments à cathode commune. Affichage des 16 chiffres hexadécimaux.**

#### Affichage des chiffres décimaux et hexadécimaux.



New Project Wizard

### Add Files [page 2 of 5]

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.  
 Note: you can always add design files to the project later.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version

*Aucune librairie à ajouter au projet !*

Specify the path names of any non-default libraries.

☞ Choisir le circuit logique à programmer : **EPM570T100C5**.

New Project Wizard

### Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.  
You can install additional device support with the Install Devices command on the Tools menu.

Device family

Family: MAX II

Devices: All

Show in 'Available devices' list

Package: Any

Pin count: 100

Speed grade: Any

Name filter:

☒ Show advanced devices ☐ HardCopy compatible only

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Available devices:

Name	Core Voltage	LEs	UFM blocks
EPM570T100C3	3.3V	570	1
EPM570T100C4	3.3V	570	1
<b>EPM570T100C5</b>	<b>3.3V</b>	<b>570</b>	<b>1</b>
EPM570T100I5	3.3V	570	1
EPM570ZM100C6	1.8V	570	1
EPM570ZM100C7	1.8V	570	1
EPM570ZM100I8	1.8V	570	1

Companion device

HardCopy:

☐ Limit DSP & RAM to HardCopy device resources

< Back Next > Finish Cancel Help

☞ Choisir le module de simulation : **ModelSim-Altera VHDL**.

New Project Wizard

### EDA Tool Settings [page 4 of 5]

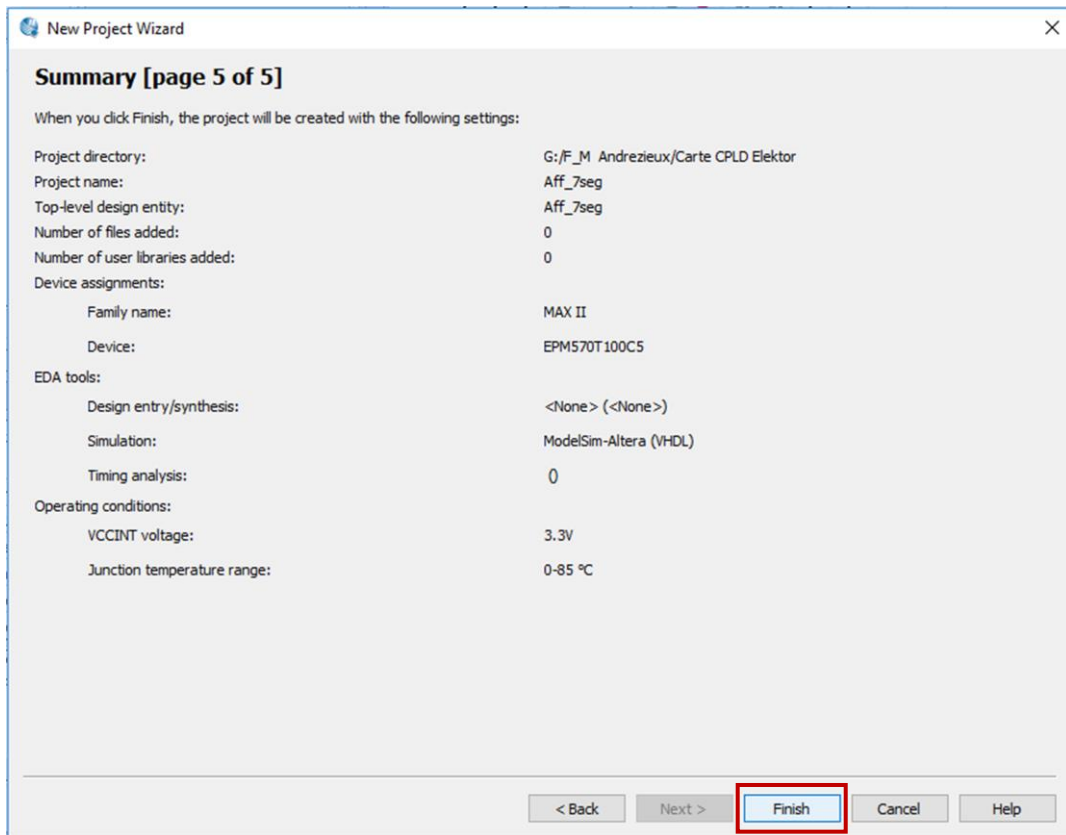
Specify the other EDA tools used with the Quartus II software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synthesis	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
<b>Simulation</b>	<b>ModelSim-Altera</b>	<b>VHDL</b>	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Formal Verification	<None>	<None>	
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

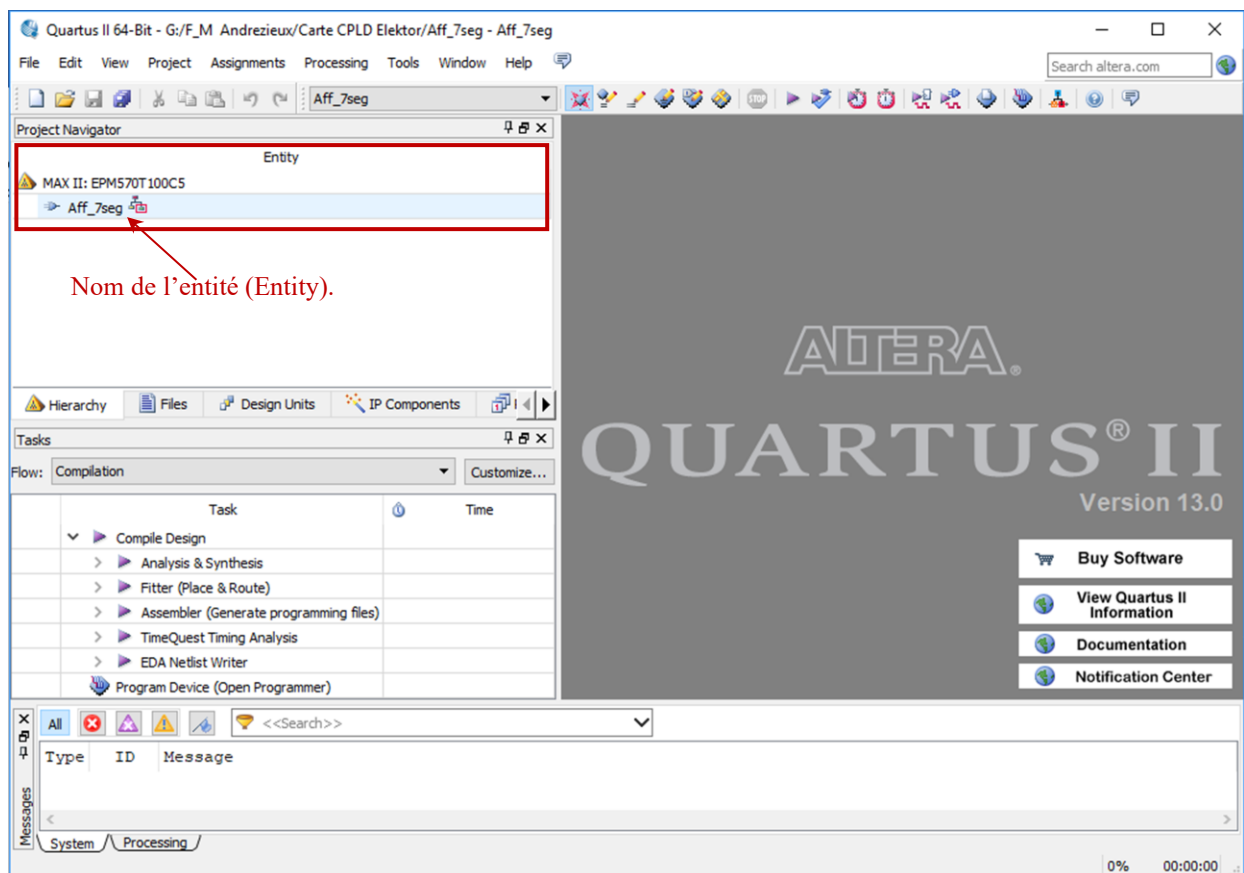
< Back Next > Finish Cancel Help



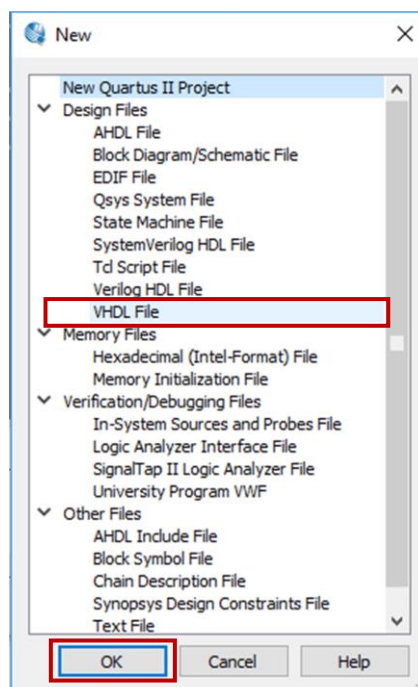
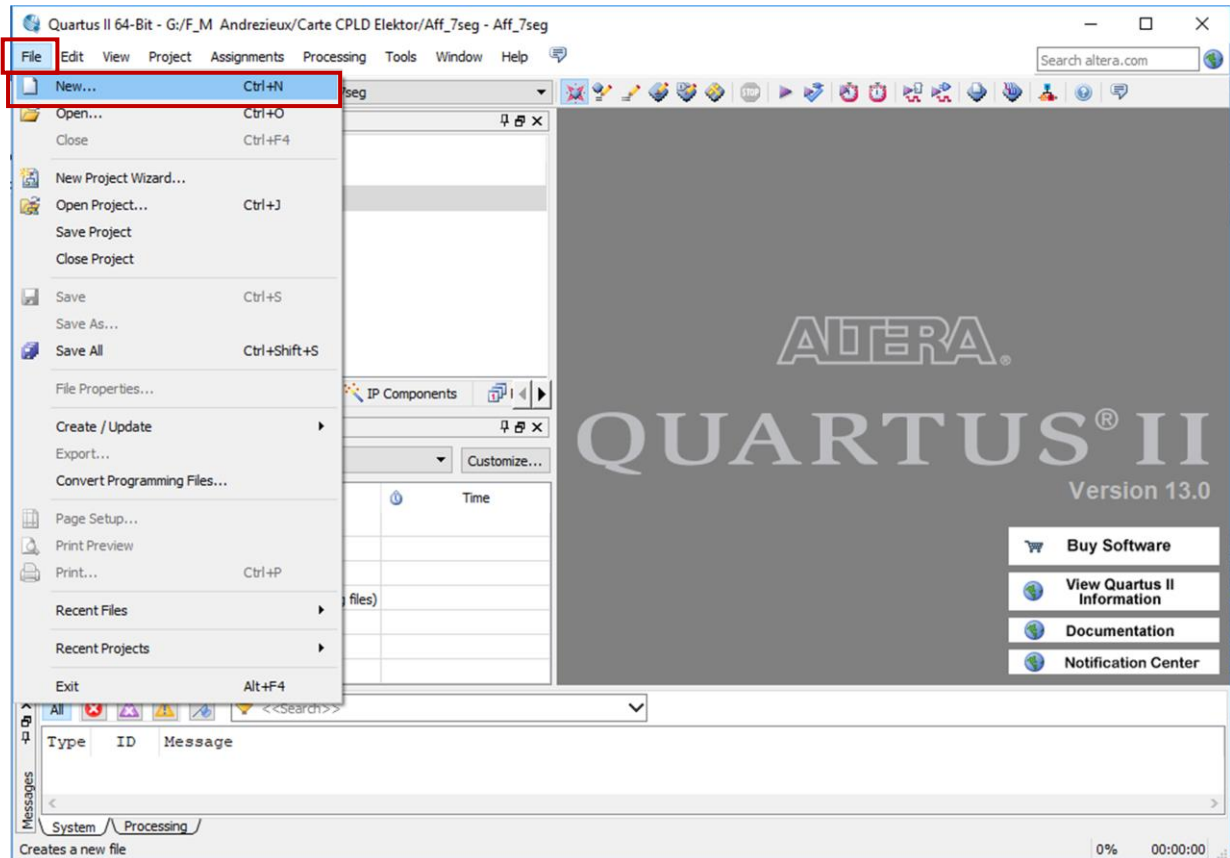


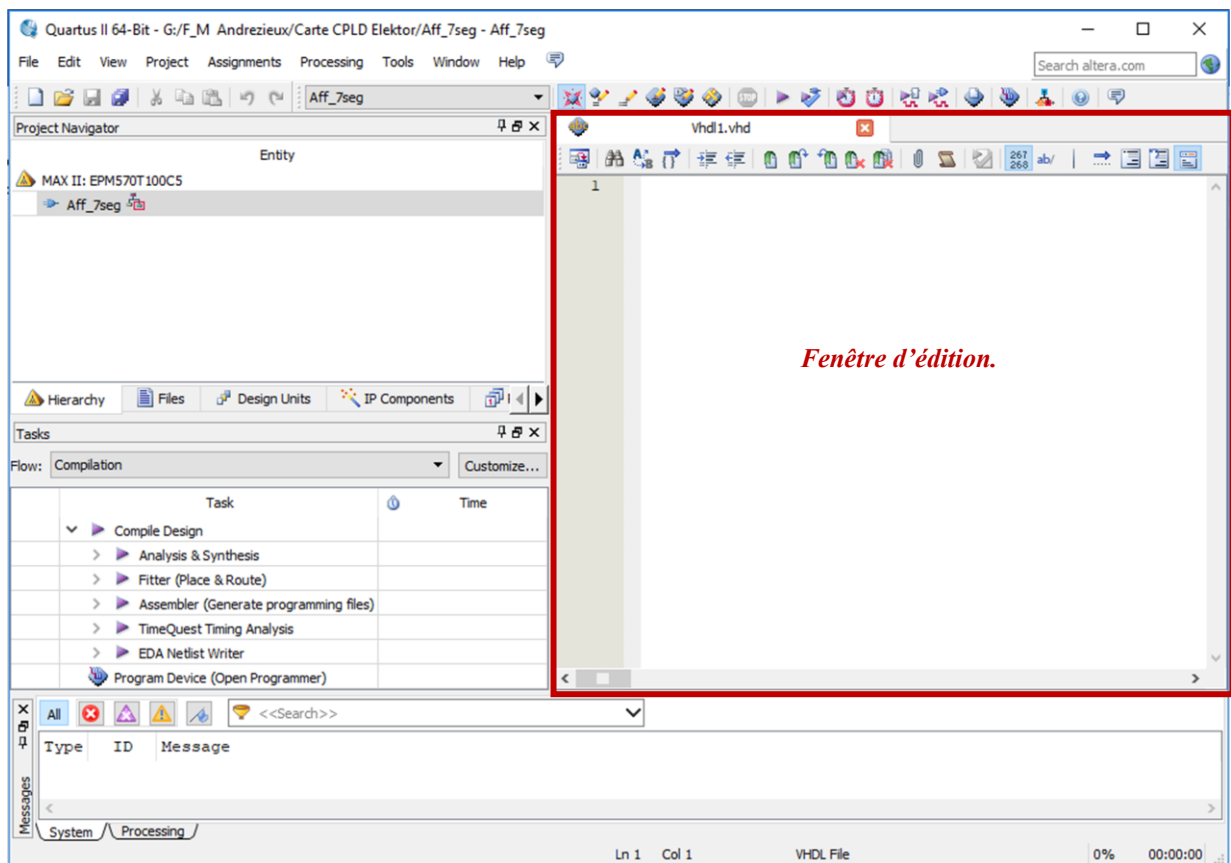
☞ *Cliquer sur "Finish" pour fermer l'assistant.*

## ✂ Environnement de développement "QUARTUS".



☞ *Cliquer sur "File" → "New..." afin de choisir le langage de programmation : VHDL.*





☺ **Programme du décodeur BCD-7Segments. Les équations simplifiées des segments sont données !**

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

Entity Aff_7seg is
    PORT(A, B, C, D : IN STD_LOGIC;
         Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC);
END Aff_7seg;

ARCHITECTURE Afficheur OF Aff_7seg is
BEGIN

    Sa <= (NOT(A OR C)) OR (B AND C) OR (B AND NOT(D)) OR (A AND C AND NOT(D)) OR (NOT(B)
AND NOT(C) AND D);

    Sb <= (NOT(C OR D)) OR (NOT(A OR C)) OR (A AND NOT(B) AND D) OR (A AND B AND NOT(D)) OR
(NOT(A OR B OR D));

    Sc <= (NOT(C) AND D) OR (C AND NOT(D)) OR (A AND NOT(B)) OR (A AND NOT(C)) OR (NOT(B OR
C));

    Sd <= (NOT(A OR B OR C)) OR (NOT(A) AND NOT(B) AND D) OR (B AND NOT(C) AND NOT(D)) OR
(A AND NOT(C) AND D) OR (NOT(A) AND B AND C) OR (A AND NOT(B) AND C);

    Se <= (NOT(A) AND B) OR (C AND D) OR (NOT(A OR C)) OR (B AND D);

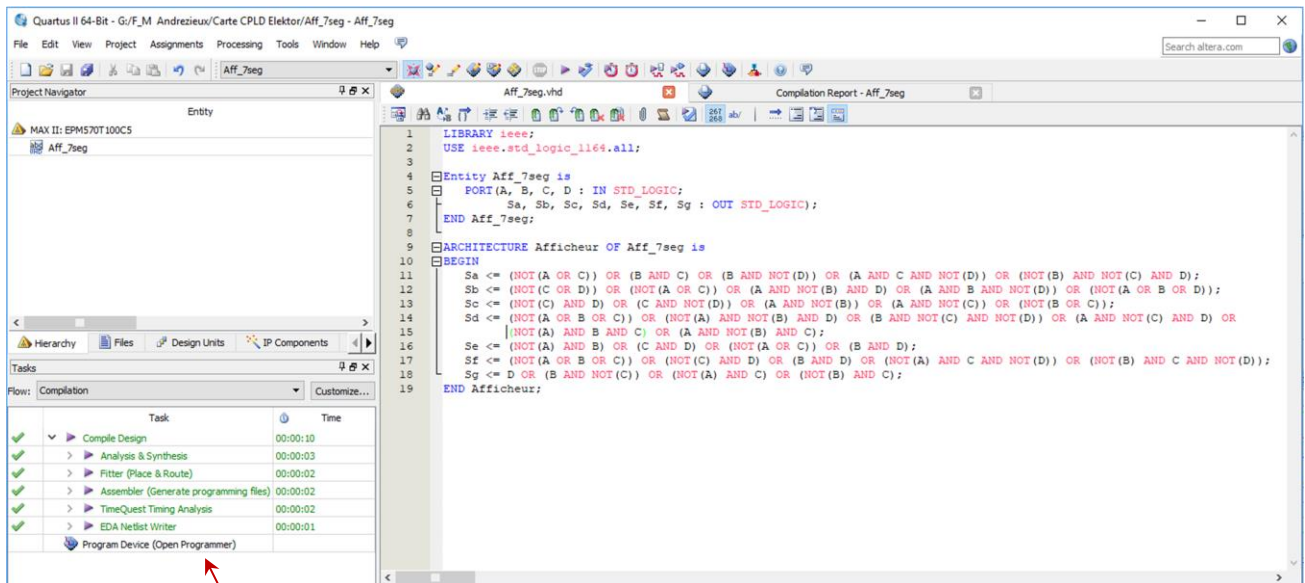
    Sf <= (NOT(A OR B OR C)) OR (NOT(C) AND D) OR (B AND D) OR (NOT(A) AND C AND NOT(D)) OR
(NOT(B) AND C AND NOT(D));

    Sg <= D OR (B AND NOT(C)) OR (NOT(A) AND C) OR (NOT(B) AND C);

END Afficheur;

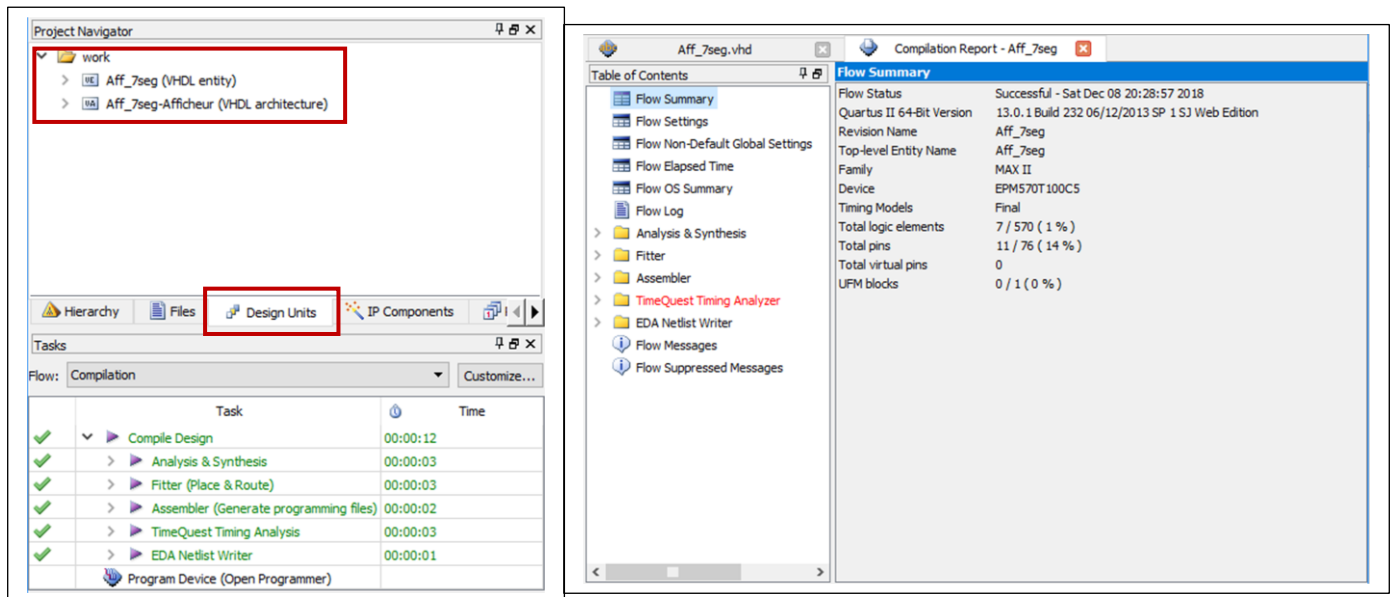
```





Résultat de la compilation.

- ☺ A la fin de la compilation, un dossier "**work**" est créé ...
- ☺ La fenêtre "**Flow Summary**" résume les éléments du projet et l'utilisation de la puce CPLD.



### Affectation des broches.

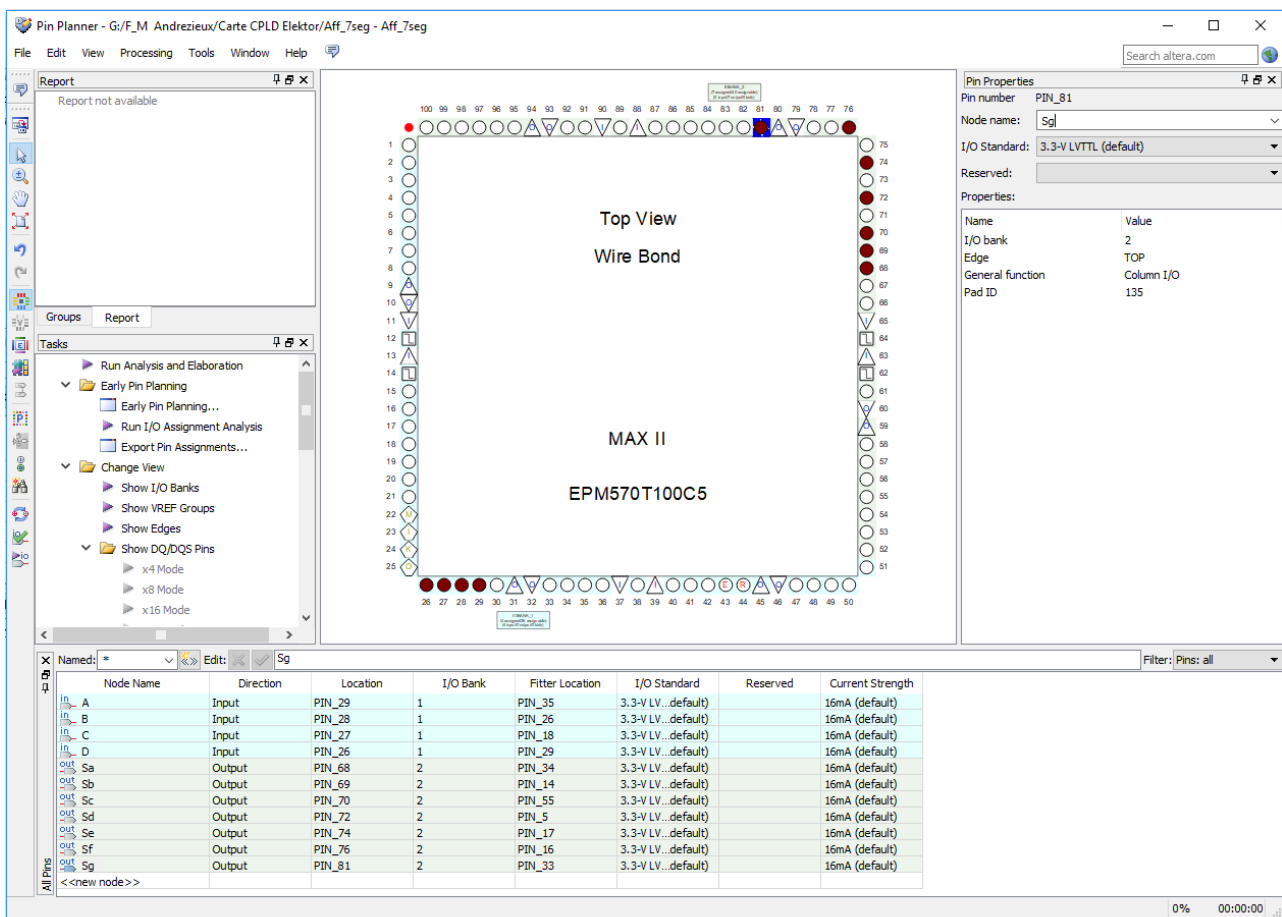
Les interrupteurs INT1, INT2, INT3 et INT4.			
29	IOB1-29	Lecture état de l'interrupteur <b>INT1</b>	
28	IOB1-28	Lecture état de l'interrupteur <b>INT2</b>	
27	IOB1-27	Lecture état de l'interrupteur <b>INT3</b>	
26	IOB1-26	Lecture état de l'interrupteur <b>INT4</b>	

Afficheurs 7-Segments : <b>AFF1</b> et <b>AFF2</b>			
68	IOB2-68	<b>AFF1</b> – Commande segment <b>"a"</b>	
69	IOB2-69	<b>AFF1</b> – Commande segment <b>"b"</b>	
70	IOB2-70	<b>AFF1</b> – Commande segment <b>"c"</b>	
72	IOB2-72	<b>AFF1</b> – Commande segment <b>"d"</b>	
74	IOB2-74	<b>AFF1</b> – Commande segment <b>"e"</b>	
76	IOB2-76	<b>AFF1</b> – Commande segment <b>"f"</b>	
81	IOB2-81	<b>AFF1</b> – Commande segment <b>"g"</b>	

### Broches utilisées.

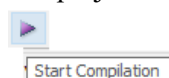
Cliquer sur l'icône **"Pin Planer"**  afin de choisir les broches à utiliser.

Un double clic sur la broche choisie ouvre une fenêtre décrivant ses propriétés. Les affectations des broches de la puce EPM570T100C5 aux broches de l'afficheur et aux interrupteurs se fera conformément au tableau donné ci-dessus.



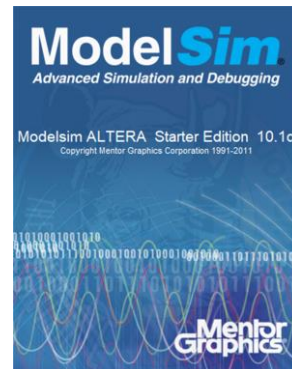
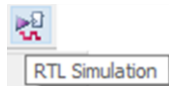
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
A	Input	PIN_29	1	PIN_35	3.3-V LV...default		16mA (default)
B	Input	PIN_28	1	PIN_26	3.3-V LV...default		16mA (default)
C	Input	PIN_27	1	PIN_18	3.3-V LV...default		16mA (default)
D	Input	PIN_26	1	PIN_29	3.3-V LV...default		16mA (default)
Sa	Output	PIN_68	2	PIN_34	3.3-V LV...default		16mA (default)
Sb	Output	PIN_69	2	PIN_14	3.3-V LV...default		16mA (default)
Sc	Output	PIN_70	2	PIN_55	3.3-V LV...default		16mA (default)
Sd	Output	PIN_72	2	PIN_5	3.3-V LV...default		16mA (default)
Se	Output	PIN_74	2	PIN_17	3.3-V LV...default		16mA (default)
Sf	Output	PIN_76	2	PIN_16	3.3-V LV...default		16mA (default)
Sg	Output	PIN_81	2	PIN_33	3.3-V LV...default		16mA (default)

☺ Une fois l'affectation des broches réalisée, recompiler le projet en cliquant sur l'icône **"Start Compilation"**.

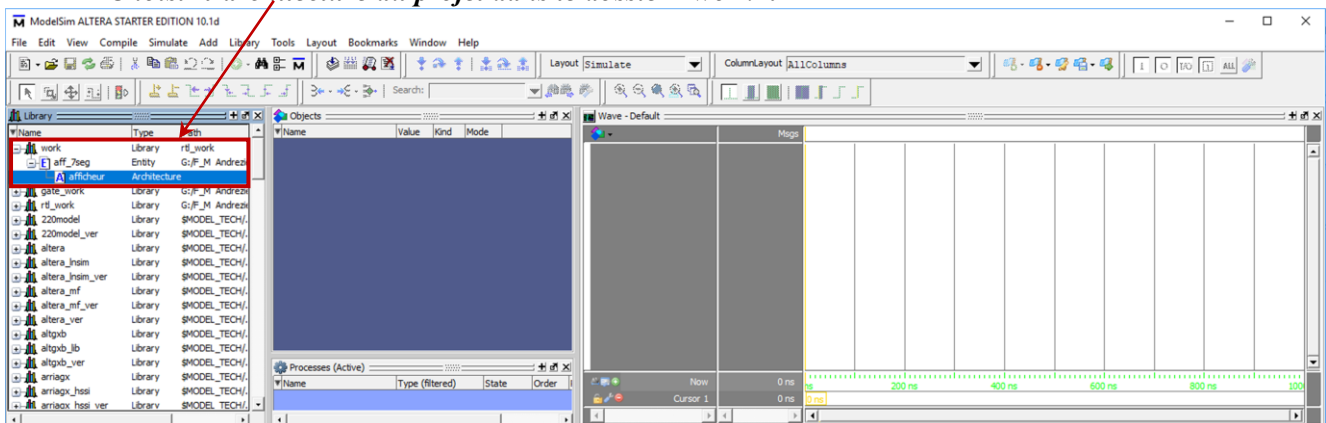


## SIMULATION.

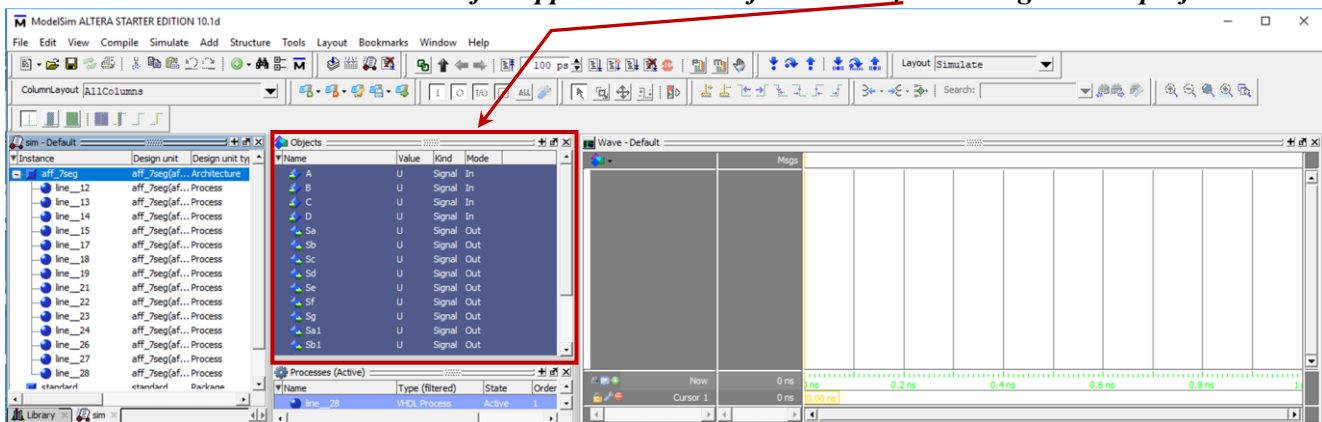
☺ Lancer le module de simulation en cliquant sur l'icône "RTL Simulation" dans la barre d'outils QUARTUS.



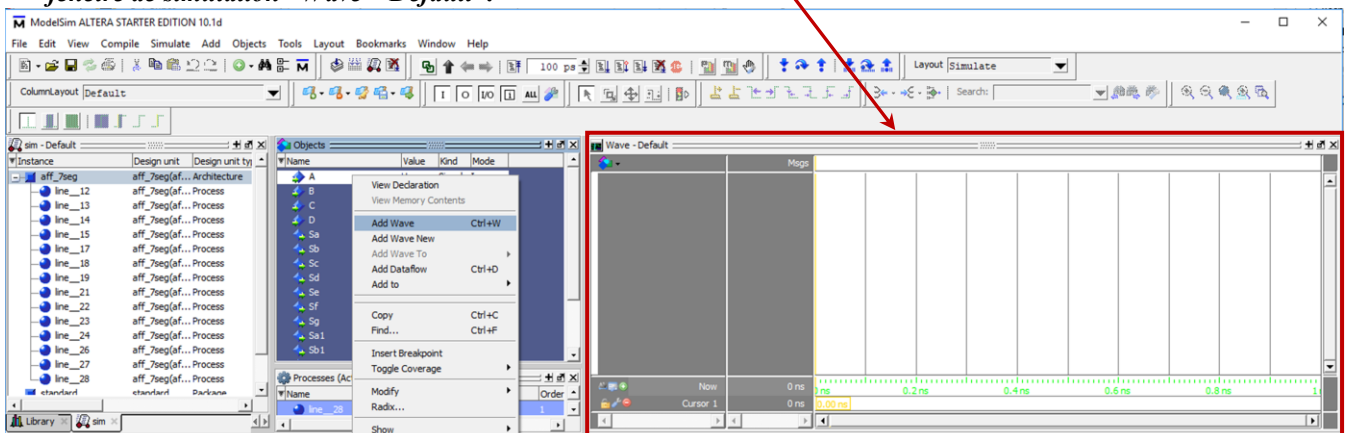
☞ Choisir l'architecture du projet dans le dossier "work".



... Un double clic sur "Architecture" fait apparaître dans la fenêtre "Objets" les signaux du projet.

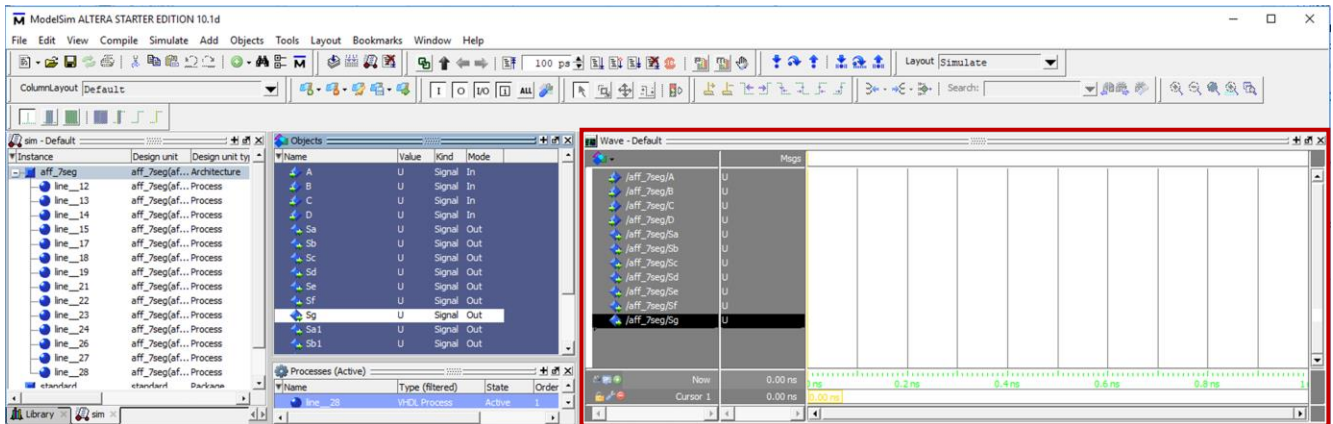


... Faire un clic droit sur chacun des signaux et choisir "Add Wave". Les signaux sont ainsi transférés dans la fenêtre de simulation "Wave - Default".

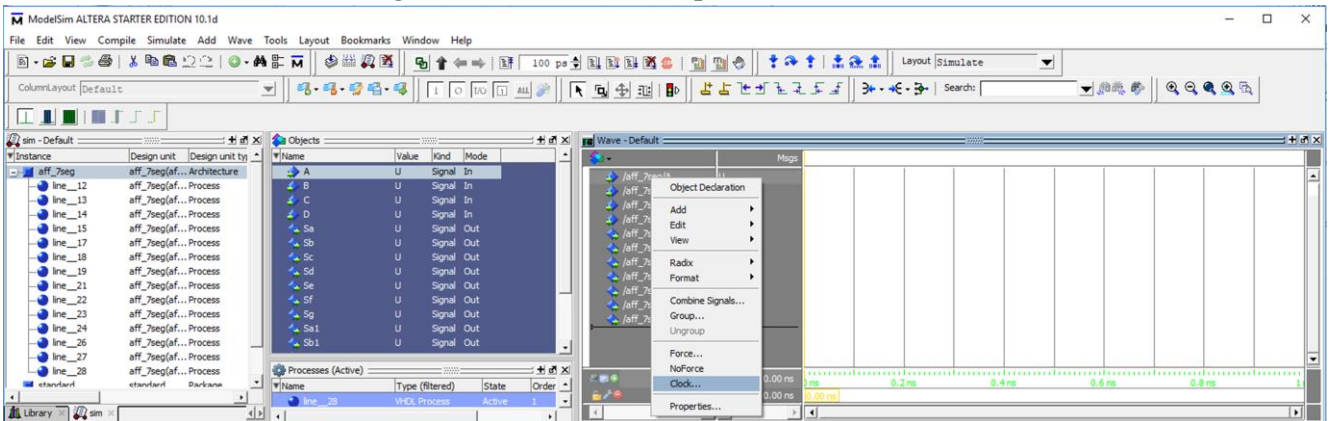




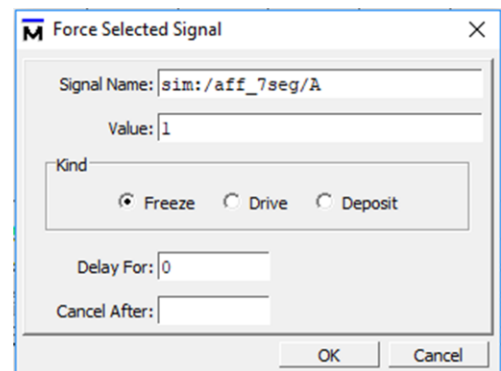
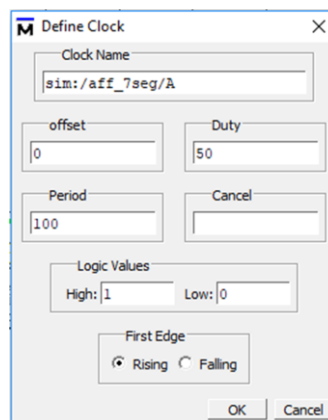
## Paramétrage des signaux "A", "B", "C" et "D".



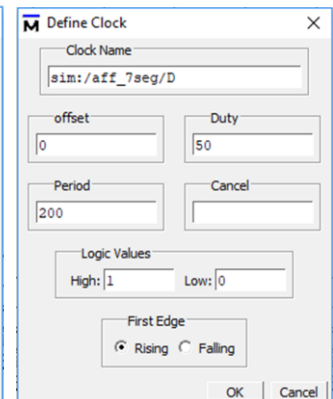
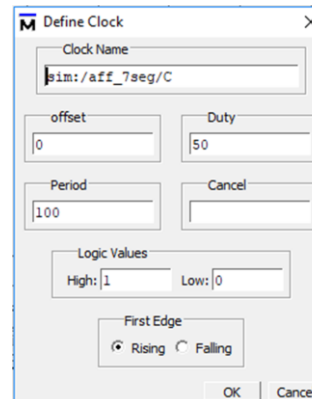
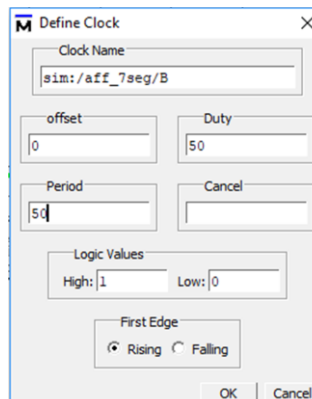
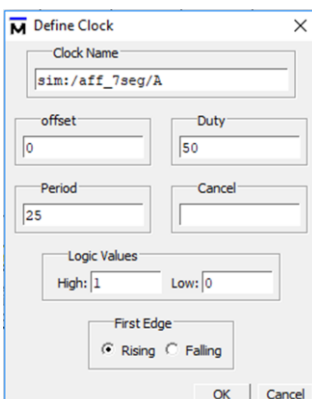
Faire un clic droit sur un signal et choisir l'une des options "Force...", "NoForce" ou "Clock..."



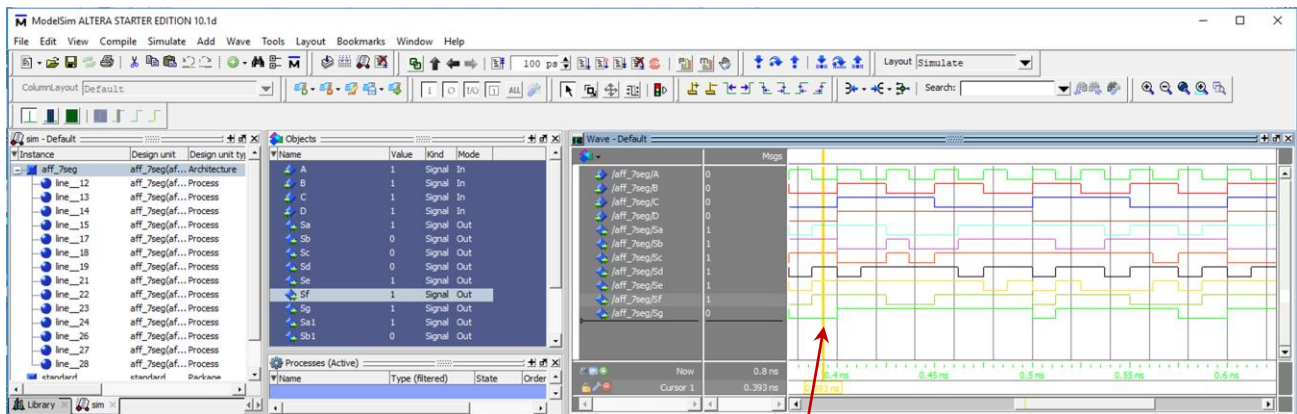
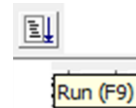
➤ Chacune des options "Clock..." et "Force..." sera paramétrée en fonction du résultat souhaité de la simulation.



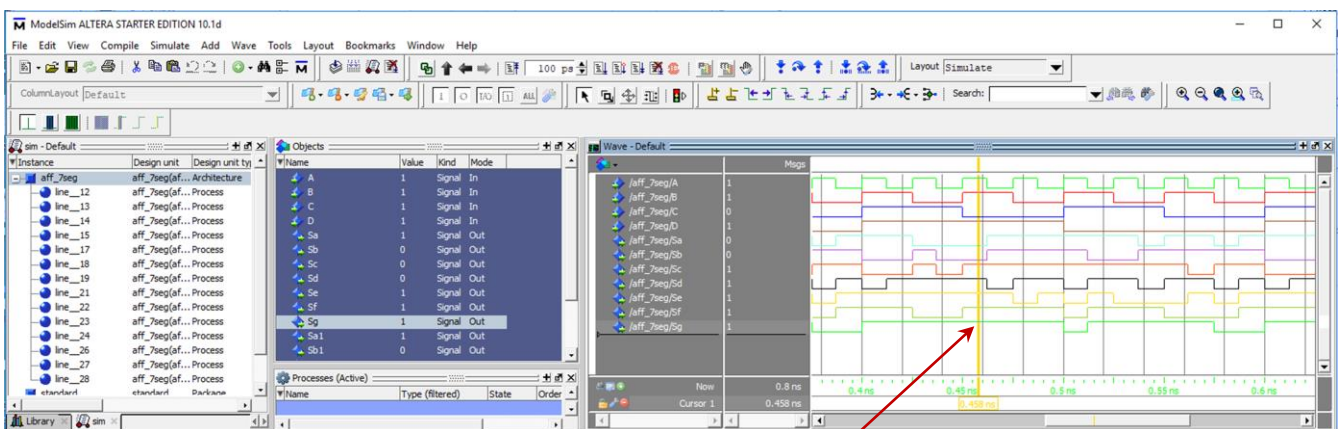
Pour les signaux A, B, C et D du décodeur BCD-7Segments, on choisit l'option "Clock" avec le paramétrage suivant :



☞ Lancer la simulation en cliquant successivement sur l'icône "Run".



Affichage du "0" : A = 0 ; B = 0 ; C = 0 et D = 0. Tous les segments sont allumés sauf le "g".



Affichage du "b" : A = 1 ; B = 1 ; C = 0 et D = 1 (0xb soit 11 en valeur décimale).

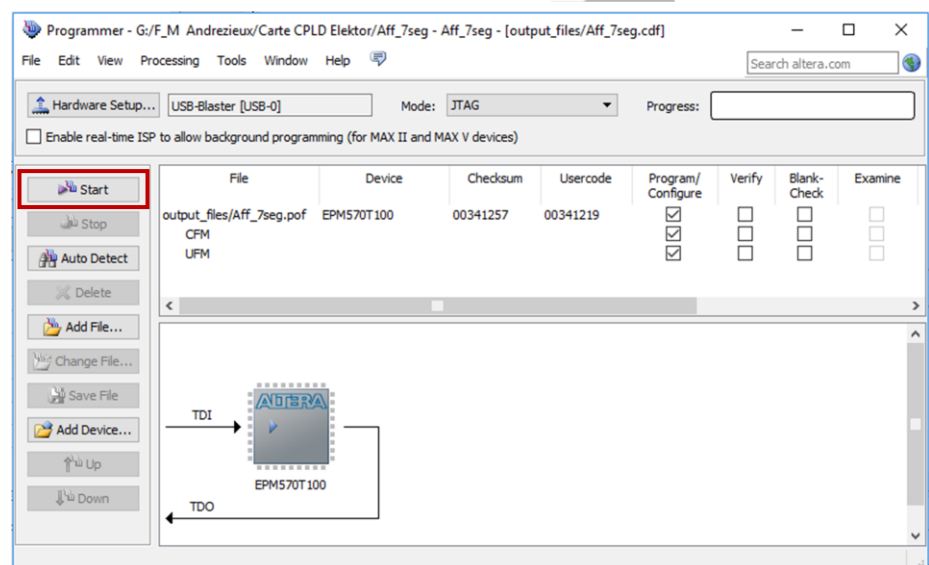
Tous les segments sont allumés sauf le "a" et le "b".

## PROGRAMMATION DE LA PUCE CPLD : EPM570T100C5.

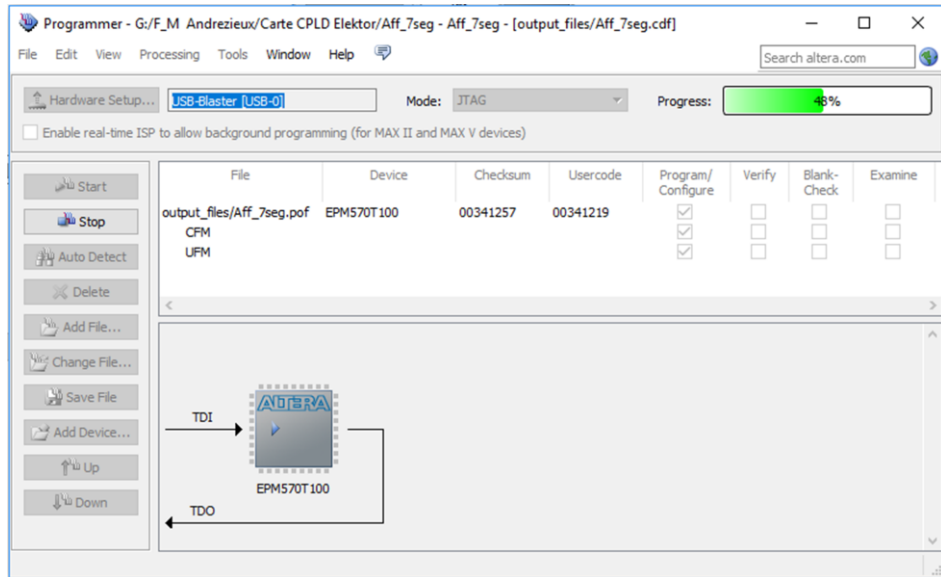
➤ Cliquer sur l'icône "Programmer" de la barre d'outils QUARTUS.



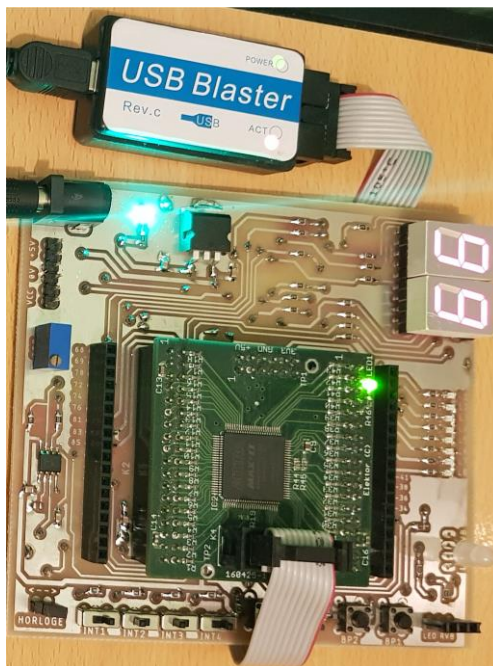
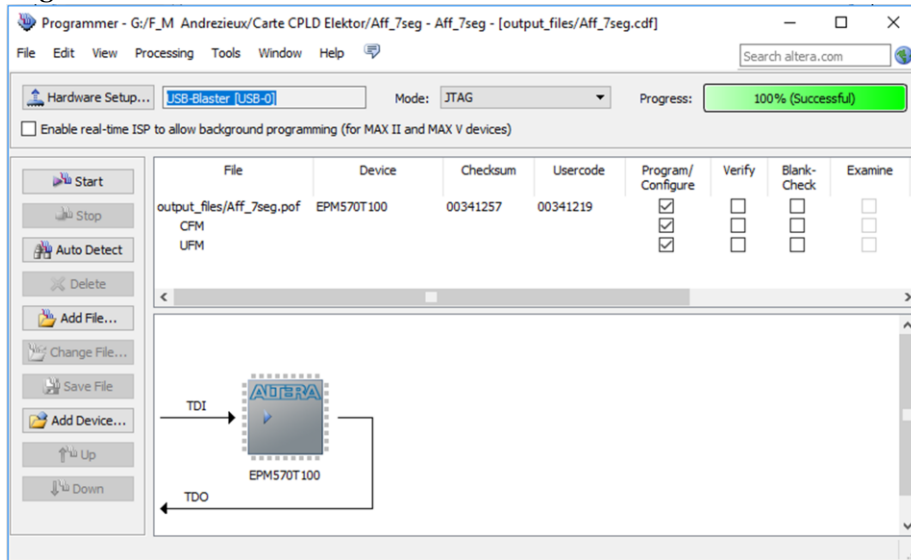
Cliquer ensuite sur "Start" pour lancer la programmation de la puce.



### ... Téléchargement



*Fin du téléchargement.*



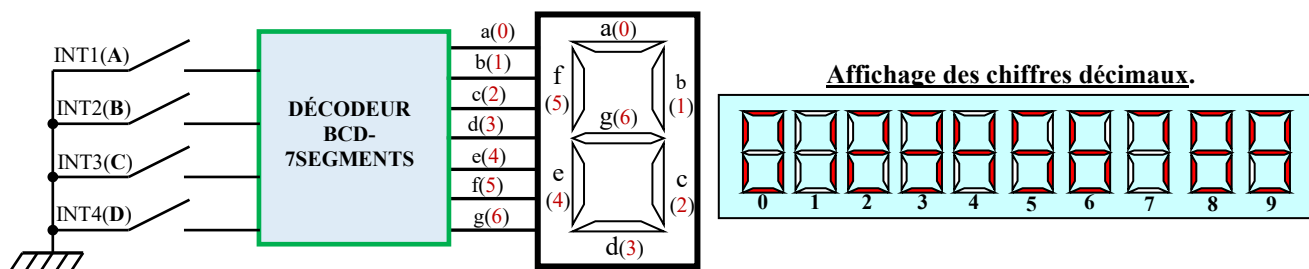


✎ Programme VHDL du décodeur utilisant l'instruction "when...else". Les équations simplifiées des segments de l'afficheur ne sont plus utilisées !

La syntaxe de l'instruction "when...else" est la suivante :

```
cible <= source1 when condition_booléenne1 else
          source2 when condition_booléenne2 else
          source3 when condition_booléenne3 else
          .....
          .....
          sourceN;
```

**DÉCODEUR BCD 7-SEGMENTS. Affichage des 10 chiffres décimaux.**



## Équations logiques des segments.

Ces équations, simplifiées par les diagrammes de KARNAUGH, sont issues de la table de vérité suivante :

Déc.	D	C	B	A	Sa	Sb	Sc	Sd	Se	Sf	Sg
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
	1	0	1	0	0	0	0	0	0	0	0
	1	0	1	1	0	0	0	0	0	0	0
	1	1	0	0	0	0	0	0	0	0	0
	1	1	0	1	0	0	0	0	0	0	0
	1	1	1	0	0	0	0	0	0	0	0
	1	1	1	1	0	0	0	0	0	0	0

Segments éteints

## Équations simplifiées des segments :

$$Sa = (\overline{B+C}) \cdot (\overline{A} + D) + B \cdot (\overline{C+D}) + C \cdot \overline{D} \cdot (A+B)$$

$$Sb = (\overline{C+D}) + (\overline{B+C}) + \overline{D} \cdot (\overline{A \oplus B})$$

$$Sc = (\overline{B+C}) + \overline{D} \cdot (C+A \cdot B)$$

$$Sd = (\overline{A+B+C}) + \overline{D} \cdot (B \cdot \overline{C} + \overline{A} \cdot B + A \cdot \overline{B} \cdot C)$$

$$Se = \overline{A} \cdot ((\overline{C+B \cdot D}) + B \cdot \overline{D})$$

$$Sf = (\overline{A+B+C}) + (\overline{B+C}) \cdot D + (\overline{A \cdot B}) \cdot C \cdot \overline{D}$$

$$Sg = B \cdot (\overline{C+D}) + \overline{B} \cdot (C \oplus D) + C \cdot (\overline{A+D})$$

## Programme VHDL.

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  -- Décodeur BCD 7-Segments
5  Entity AffichageDec3 is
6  PORT (A, B, C, D : IN STD_LOGIC; -- Les entrées du décodeur
7       Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC; -- Les segments de l'afficheur 1
8       Sd1, Sb1, Sc1, Sd1, Se1, Sf1, Sg1 : OUT STD_LOGIC); -- Les segments de l'afficheur 3
9  END AffichageDec3;
10
11 ARCHITECTURE Afficheur OF AffichageDec3 is
12 BEGIN
13
14 -- Equations des 7-segments de l'afficheur : Premier digit.
15 Sa <= ((NOT(B OR C)) AND (NOT(A OR D)) OR ((C AND NOT(D)) AND (A OR B)));
16 Sb <= ((NOT(C OR D)) OR (NOT(B OR C)) OR (NOT(D) AND NOT(A XOR B)));
17 Sc <= ((NOT(B OR C)) OR ((NOT(D)) AND (C OR (A AND B))));
18 Sd <= ((NOT(A OR B OR C)) OR ((NOT(D)) AND ((B AND (NOT(C))) OR ((NOT(A)) AND B) OR (A AND (NOT(B)) AND C))));
19 Se <= ((NOT(A)) AND ((NOT(C OR (B AND D))) OR (B AND (NOT(D)))));
20 Sf <= ((NOT(A OR B OR C)) OR ((NOT(B OR C)) AND D) OR ((NOT(A AND B)) AND (C AND (NOT(D)))));
21 Sg <= ((B AND (NOT(C OR D))) OR ((NOT(B)) AND (C XOR D)) OR (C AND (NOT(A OR D))));
22
23 -- Equations des 7-segments de l'afficheur : Deuxième digit.
24 Sa1 <= ((NOT(B OR C)) AND (NOT(A OR D)) OR (B AND (NOT(C OR D)) OR ((C AND NOT(D)) AND (A OR B)));
25 Sb1 <= ((NOT(C OR D)) OR (NOT(B OR C)) OR (NOT(D) AND NOT(A XOR B)));
26 Sc1 <= ((NOT(B OR C)) OR ((NOT(D)) AND (C OR (A AND B))));
27 Sd1 <= ((NOT(A OR B OR C)) OR ((NOT(D)) AND ((B AND (NOT(C))) OR ((NOT(A)) AND B) OR (A AND (NOT(B)) AND C))));
28 Se1 <= ((NOT(A)) AND ((NOT(C OR (B AND D))) OR (B AND (NOT(D)))));
29 Sf1 <= ((NOT(A OR B OR C)) OR ((NOT(B OR C)) AND D) OR ((NOT(A AND B)) AND (C AND (NOT(D)))));
30 Sg1 <= ((B AND (NOT(C OR D)) OR ((NOT(B)) AND (C XOR D)) OR (C AND (NOT(A OR D))));
31
32 END Afficheur;

```

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in_A	Input	PIN_29	1	PIN_29	3.3-V LV...default		16mA (default)
in_B	Input	PIN_28	1	PIN_28	3.3-V LV...default		16mA (default)
in_C	Input	PIN_27	1	PIN_27	3.3-V LV...default		16mA (default)
in_D	Input	PIN_26	1	PIN_26	3.3-V LV...default		16mA (default)
out_Ga	Output	PIN_68	2	PIN_68	3.3-V LV...default		16mA (default)
out_Sa1	Output	PIN_83	2	PIN_83	3.3-V LV...default	As Signa...e output	16mA (default)
out_Sb1	Output	PIN_69	2	PIN_69	3.3-V LV...default		16mA (default)
out_Sc1	Output	PIN_85	2	PIN_85	3.3-V LV...default		16mA (default)
out_Sd1	Output	PIN_70	2	PIN_70	3.3-V LV...default		16mA (default)
out_Se1	Output	PIN_87	2	PIN_87	3.3-V LV...default		16mA (default)
out_Sf1	Output	PIN_72	2	PIN_72	3.3-V LV...default		16mA (default)
out_Sg1	Output	PIN_91	2	PIN_91	3.3-V LV...default		16mA (default)
out_Sd1	Output	PIN_74	2	PIN_74	3.3-V LV...default		16mA (default)
out_Se1	Output	PIN_95	2	PIN_95	3.3-V LV...default		16mA (default)
out_Sf1	Output	PIN_76	2	PIN_76	3.3-V LV...default		16mA (default)
out_Sg1	Output	PIN_97	2	PIN_97	3.3-V LV...default		16mA (default)
out_Sd1	Output	PIN_81	2	PIN_81	3.3-V LV...default		16mA (default)
out_Sg1	Output	PIN_99	2	PIN_99	3.3-V LV...default		16mA (default)

## Programme VHDL du décodeur utilisant l'instruction "when...else"

The screenshot displays the Quartus II 64-bit AffichageDec2 project. The main window shows the VHDL code for the AffichageDec2.vhd file. The code defines an entity with two 6-bit input ports (AF1 and AF2) and two 6-bit output ports (AF1 and AF2). The architecture implements a BCD 7-Segments decoder using a 'when...else' statement to map 4-bit BCD inputs to 7-segment outputs. The code is as follows:

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  -- Décodeur BCD 7-Segments
5  Entity AffichageDec2 is
6  Port ( A : IN std_logic_vector(3 downto 0); -- Les entrées du décodeur
7        AF1 : OUT std_logic_vector(6 downto 0); -- Les segments de l'afficheur 1
8        AF2 : OUT std_logic_vector(6 downto 0); -- Les segments de l'afficheur 2
9  );
10 END AffichageDec2;
11
12 ARCHITECTURE Afficheur OF AffichageDec2 is
13 BEGIN
14     -- Codage 1'afficheur 1.
15     AF1 <= "0111111" when A = "0000" else --Affichage du 0
16            "0000110" when A = "0001" else --Affichage du 1
17            "1011011" when A = "0010" else --Affichage du 2
18            "1001111" when A = "0011" else --Affichage du 3
19            "1100110" when A = "0100" else --Affichage du 4
20            "1101101" when A = "0101" else --Affichage du 5
21            "1111101" when A = "0110" else --Affichage du 6
22            "0000111" when A = "0111" else --Affichage du 7
23            "1111111" when A = "1000" else --Affichage du 8
24            "1100111" when A = "1001" else --Affichage du 9
25            "0000000"; -- Afficheur éteint.
26
27     -- Codage de 1'afficheur 2.
28     AF2 <= "0111111" when A = "0000" else --Affichage du 0
29            "0000110" when A = "0001" else --Affichage du 1
30            "1011011" when A = "0010" else --Affichage du 2
31            "1001111" when A = "0011" else --Affichage du 3
32            "1100110" when A = "0100" else --Affichage du 4
33            "1101101" when A = "0101" else --Affichage du 5
34            "1111101" when A = "0110" else --Affichage du 6
35            "0000111" when A = "0111" else --Affichage du 7
36            "1111111" when A = "1000" else --Affichage du 8
37            "1100111" when A = "1001" else --Affichage du 9
38            "0000000"; -- Afficheur éteint.
39 END Afficheur;

```

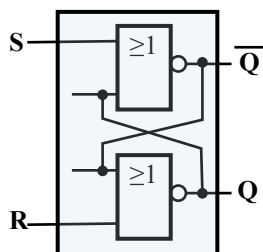
Below the code, a pin assignment table is shown, detailing the connections for the decoder circuit. The table includes columns for Node Name, Direction, Location, I/O Bank, Fitter Location, I/O Standard, Reserved, and Current Strength.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
A[3]	Input	PIN_26	1	PIN_26	3.3-V LV..default		16mA (default)
A[2]	Input	PIN_27	1	PIN_27	3.3-V LV..default		16mA (default)
A[1]	Input	PIN_28	1	PIN_28	3.3-V LV..default		16mA (default)
A[0]	Input	PIN_29	1	PIN_29	3.3-V LV..default		16mA (default)
AF1[6]	Output	PIN_81	2	PIN_81	3.3-V LV..default		16mA (default)
AF1[5]	Output	PIN_76	2	PIN_76	3.3-V LV..default		16mA (default)
AF1[4]	Output	PIN_74	2	PIN_74	3.3-V LV..default		16mA (default)
AF1[3]	Output	PIN_72	2	PIN_72	3.3-V LV..default		16mA (default)
AF1[2]	Output	PIN_70	2	PIN_70	3.3-V LV..default		16mA (default)
AF1[1]	Output	PIN_69	2	PIN_69	3.3-V LV..default		16mA (default)
AF1[0]	Output	PIN_68	2	PIN_68	3.3-V LV..default		16mA (default)
AF2[6]	Output	PIN_99	2	PIN_99	3.3-V LV..default		16mA (default)
AF2[5]	Output	PIN_97	2	PIN_97	3.3-V LV..default		16mA (default)
AF2[4]	Output	PIN_95	2	PIN_95	3.3-V LV..default		16mA (default)
AF2[3]	Output	PIN_91	2	PIN_91	3.3-V LV..default		16mA (default)
AF2[2]	Output	PIN_87	2	PIN_87	3.3-V LV..default		16mA (default)
AF2[1]	Output	PIN_85	2	PIN_85	3.3-V LV..default		16mA (default)
AF2[0]	Output	PIN_83	2	PIN_83	3.3-V LV..default		16mA (default)

- 1- Après avoir programmer le circuit **FPGA** et vérifier le bon fonctionnement du décodeur BCD 7-Segments, **faire valider le fonctionnement par le professeur.**
- 2- Rédiger un compte rendu du TP précisant la programmation d'un circuit "Décodeur BCD-7Segments".
- 3- Conclure.



### 1- LA BASCULE R-S : Bascule asynchrone.

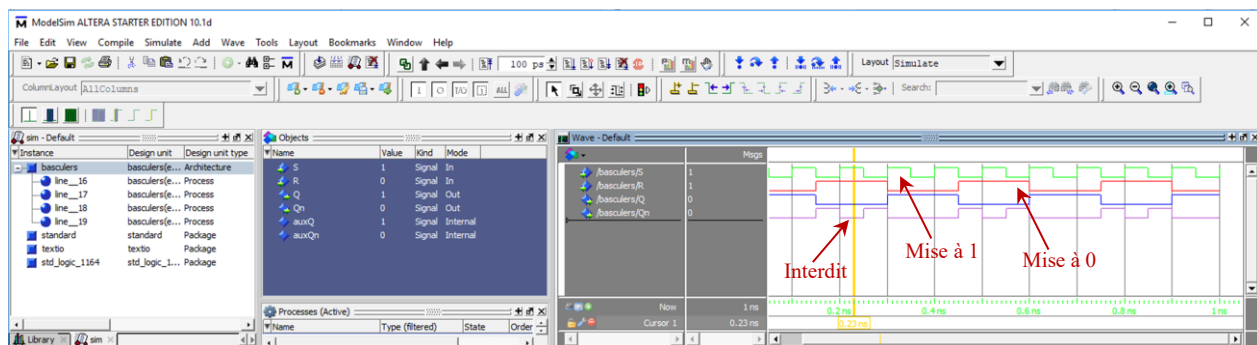
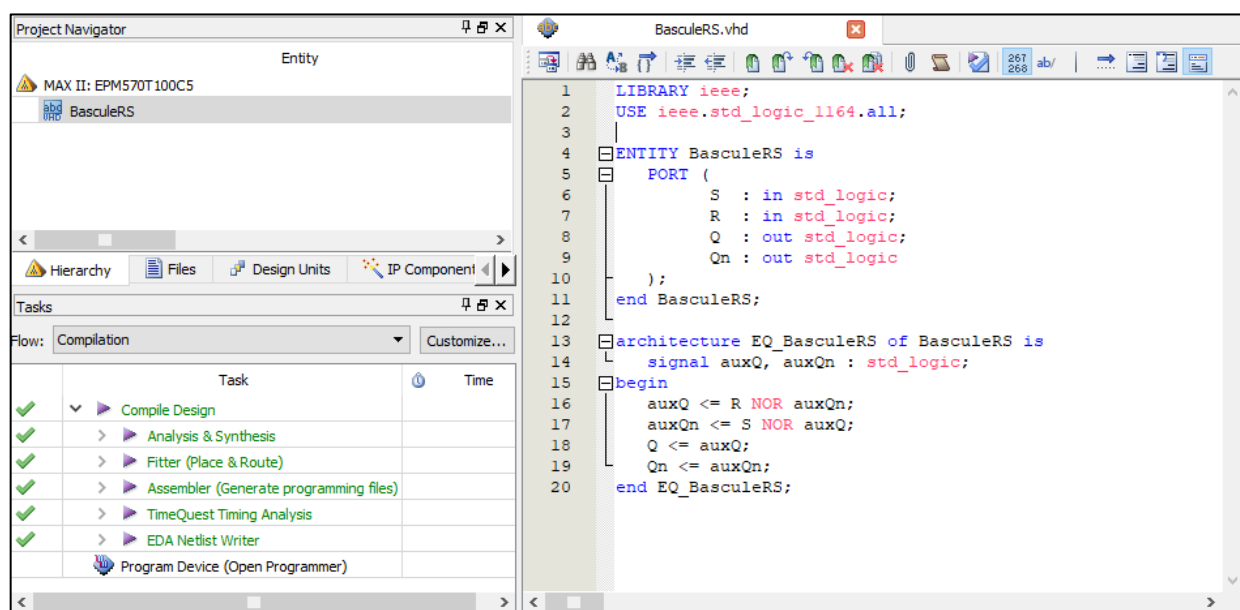


➤ Équations des sorties Q et  $\bar{Q}$  en fonction des entrées S et R.

- $Q = \overline{R + \bar{Q}}$  (1)
- $\bar{Q} = \overline{S + Q}$  (2)
- (2) dans (1) :  $Q = \overline{R + \overline{S + Q}} = \bar{R} \cdot (S + Q)$
- (1) dans (2) :  $\bar{Q} = \overline{S + \overline{R + \bar{Q}}} = \bar{S} \cdot (R + \bar{Q})$

**Table de vérité.**

S	R	Q	$\bar{Q}$	Fonction
0	0	Q	$\bar{Q}$	Mémoire
0	1	0	1	Mise à 0
1	0	1	0	Mise à 1
1	1	0	0	Interdit



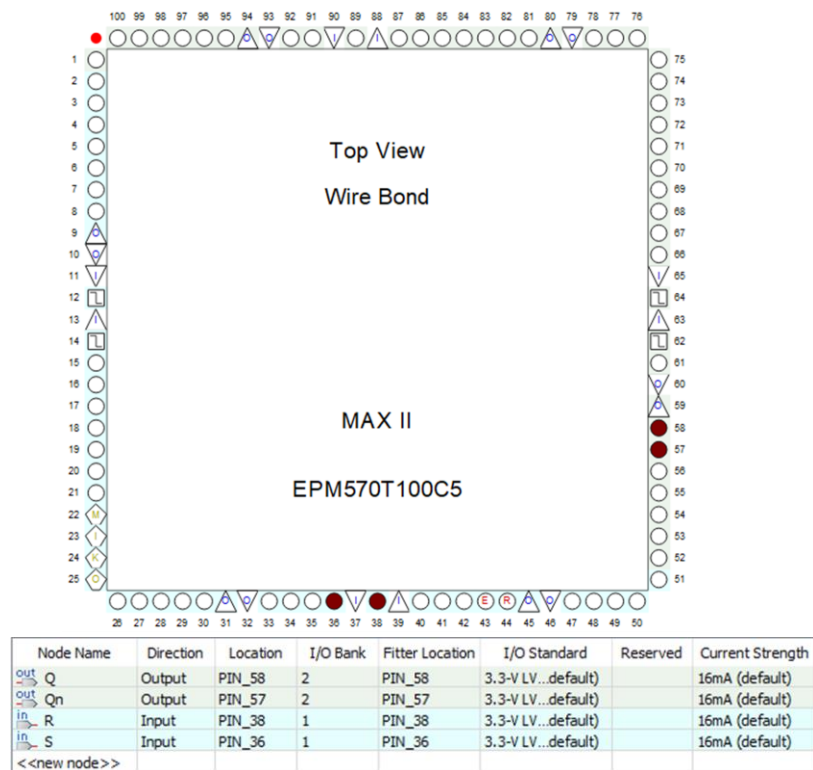
#### Affectation des broches.

##### Les entrées R (reset) et S (set) de la bascule.

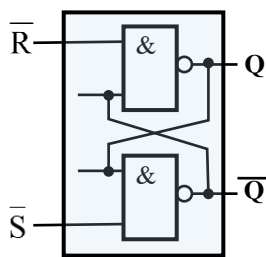
Broche N°38	IOB1-38	Lecture état du bouton poussoir <b>BP1</b>	R (Reset)
Broche N°36	IOB1-36	Lecture état du bouton poussoir <b>BP2</b>	S (Set)

##### Les sorties Q et Qn de la bascule.

Broche N°58	IOB2-58	Commande diode led <b>L1</b>	Sortie Q
Broche N°57	IOB2-57	Commande diode led <b>L2</b>	Sortie inversée Qn



## 2- LA BASCULE $\bar{R}-\bar{S}$ : Bascule asynchrone.



➤ Équations des sorties Q et  $\bar{Q}$  en fonction des entrées S et R.

- $Q = \bar{R} \cdot \bar{Q}$  (1)
- $\bar{Q} = \bar{S} \cdot Q$  (2)

Table de vérité.

$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	Fonction
0	0	1	1	Interdit
0	1	0	1	Mise à 1
1	0	1	0	Mise à 0
1	1	Q	$\bar{Q}$	Mémoire

Project Navigator

Entity

MAX II: EPM570T100C5

BasculeNRNS

Hierarchy Files Design Units IP Components

Tasks

Flow: Compilation Customize...

Task	Time
Compile Design	00:00:11
Analysis & Synthesis	00:00:03
Fitter (Place & Route)	00:00:02
Assembler (Generate programming files)	00:00:02
TimeQuest Timing Analysis	00:00:03
EDA Netlist Writer	00:00:01
Program Device (Open Programmer)	

BasculeNRNS.vhd

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY BasculeNRNS is
5  PORT (
6      Sn : in std_logic;
7      Rn : in std_logic;
8      Q  : out std_logic;
9      Qn : out std_logic
10 );
11 end BasculeNRNS;
12
13 architecture EQ_BasculeNRNS of BasculeNRNS is
14     signal auxQ, auxQn : std_logic;
15 begin
16     auxQ <= Rn NAND auxQn;
17     auxQn <= Sn NAND auxQ;
18     Q <= auxQ;
19     Qn <= auxQn;
20 end EQ_BasculeNRNS;
21

```

Compilation Report - BasculeNRNS

267 ab/



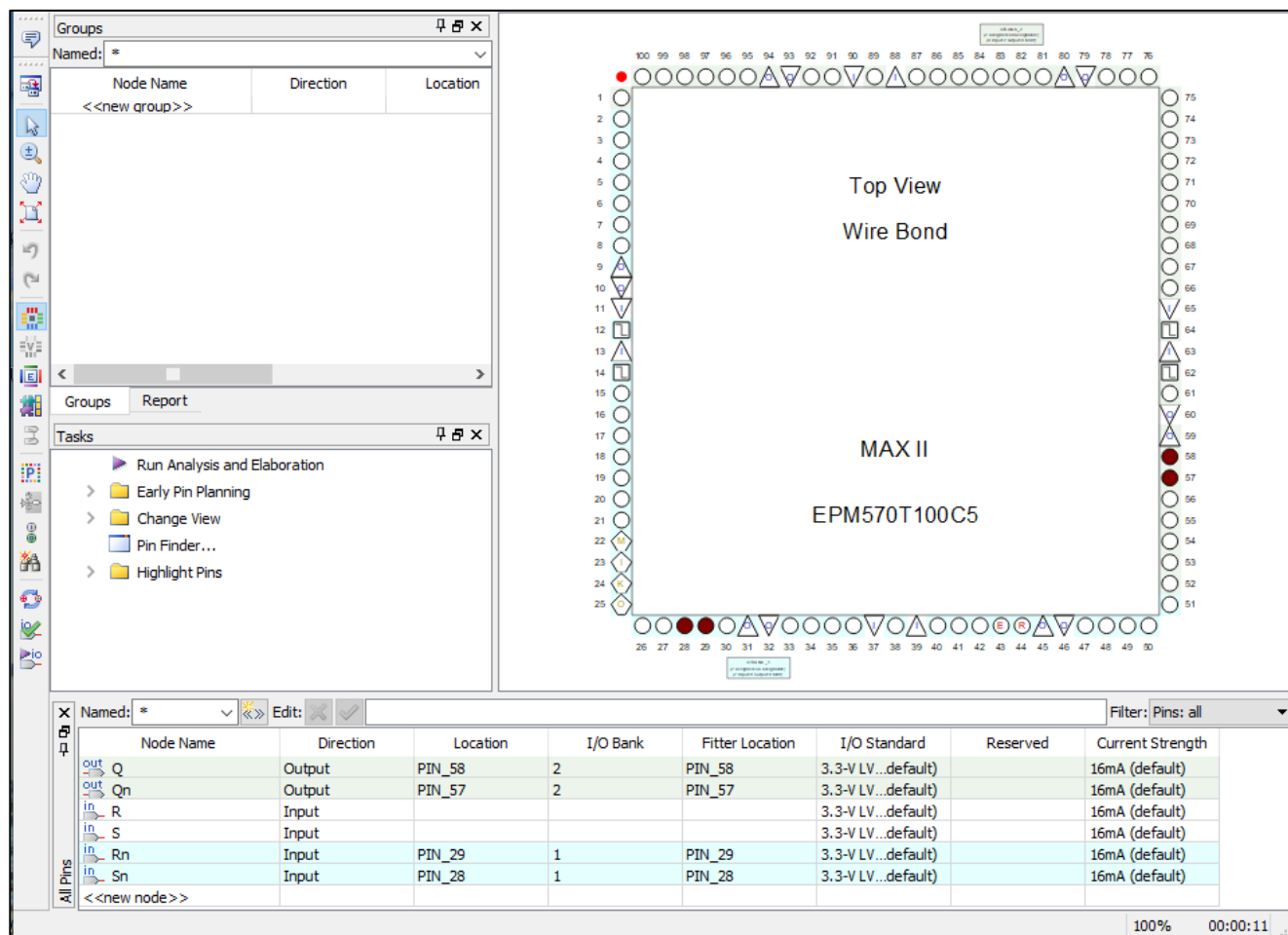
## Affectation des broches.

### Les entrées R (reset) et S (set) de la bascule.

Broche N°29	IOB1-29	Lecture état de l'interrupteur <b>INT1</b>	<b>Rn (Reset)</b>
Broche N°28	IOB1-28	Lecture état de l'interrupteur <b>INT2</b>	<b>Sn (Set)</b>

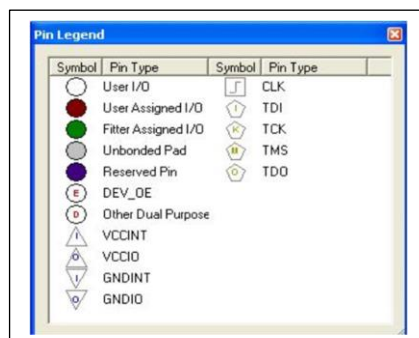
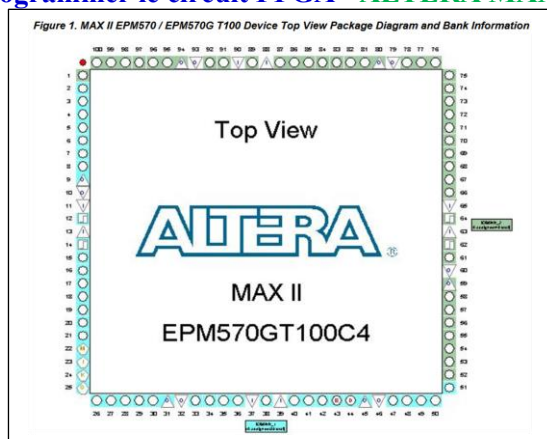
### Les sorties Q et Qn de la bascule.

Broche N°58	IOB2-58	Commande diode led <b>L1</b>	Sortie <b>Q</b>
Broche N°57	IOB2-57	Commande diode led <b>L2</b>	Sortie inversée <b>Qn</b>



Donner le résultat commenté de la simulation de cette bascule.

Programmer le circuit FPGA "ALTERA MAX II – EPM570T100C4" de la carte CPLD Elektor.



Vérifier le fonctionnement de cette bascule : faire valider le fonctionnement par le professeur.

**Remarque.** Dans les deux exemples présentés ci-dessus (**Décodeur** d'un afficheur 7-segments et **Bascule asynchrone R-S**) on a utilisé des **structures combinatoires** qui nous ont données des équations logiques décrivant entièrement le fonctionnement du décodeur et de la bascule asynchrone.

Les exemples qui suivent utilisent des **structures séquentielles** où la présence d'un signal d'horloge rythme les différentes opérations de la structure. On parle de **systèmes synchrones**.

## PROGRAMMATION EN VHDL D'UNE STRUCTURE SÉQUENTIELLE.

☺ Dans la description de l'architecture, on utilise souvent le mot clé **"process"**. Il s'agit des différentes tâches d'un programmes VHDL. Ces tâches s'exécutent en parallèle et sont appelées des **processus**. Dans un **"process"**, les instructions sont séquentielles : elles s'exécutent les unes après les autres.

☺ **Règle de fonctionnement d'un processus :**

**1-** Un processus est une boucle infinie, lorsqu'il arrive à la fin du code, il reprend automatiquement au début.  
**2-** Un processus doit être sensible des points d'arrêt de façon à le synchroniser. La synchronisation est indiquée par un point d'arrêt. Il existe deux types de points d'arrêts :

- Le processus est associé à une **"liste de sensibilité"** qui contient une **liste de signaux** qui réveille le processus lors d'un changement d'état d'un des signaux. Sa syntaxe est **"process (liste de signaux)"**

- Le processus a des instructions d'arrêt **"wait"** dans sa description interne. Le wait est sensible soit à un signal soit à un temps physique.

- Le processus fait souvent appel à la structure **"if then else"**, structure interdit en dehors du processus.

**3-** Les variables sont internes au processus et sont affectées immédiatement, contrairement aux signaux qui eux ne sont pas affectés directement mais par le biais de leur échancier qui est mis à jour en fin de processus avec la nouvelle valeur et le temps d'affectation qui correspond à un delta-cycle après le signal ayant réveillé le processus.

☞ L'instruction **"wait"** permet de mettre des points d'arrêt dans le corps du processus.

**Syntaxe : wait [S1, S2, ...] [until CONDITION] [for DUREE];**

**S1** et **S2** sont des signaux, **CONDITION** est une expression générant un booléen et **DUREE** est le temps physique d'attente.

**Remarque.**

*Ne pas oublier qu'un signal correspondra physiquement à un fil. On ne peut pas utiliser un signal comme une variable dont l'affectation serait immédiate.*

### 3- LA BASCULE D. (Bascule synchrone)

#### ➤ Symbole et table de vérité.

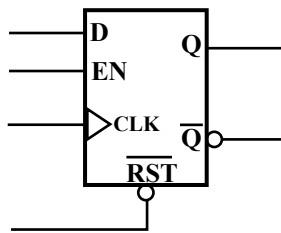
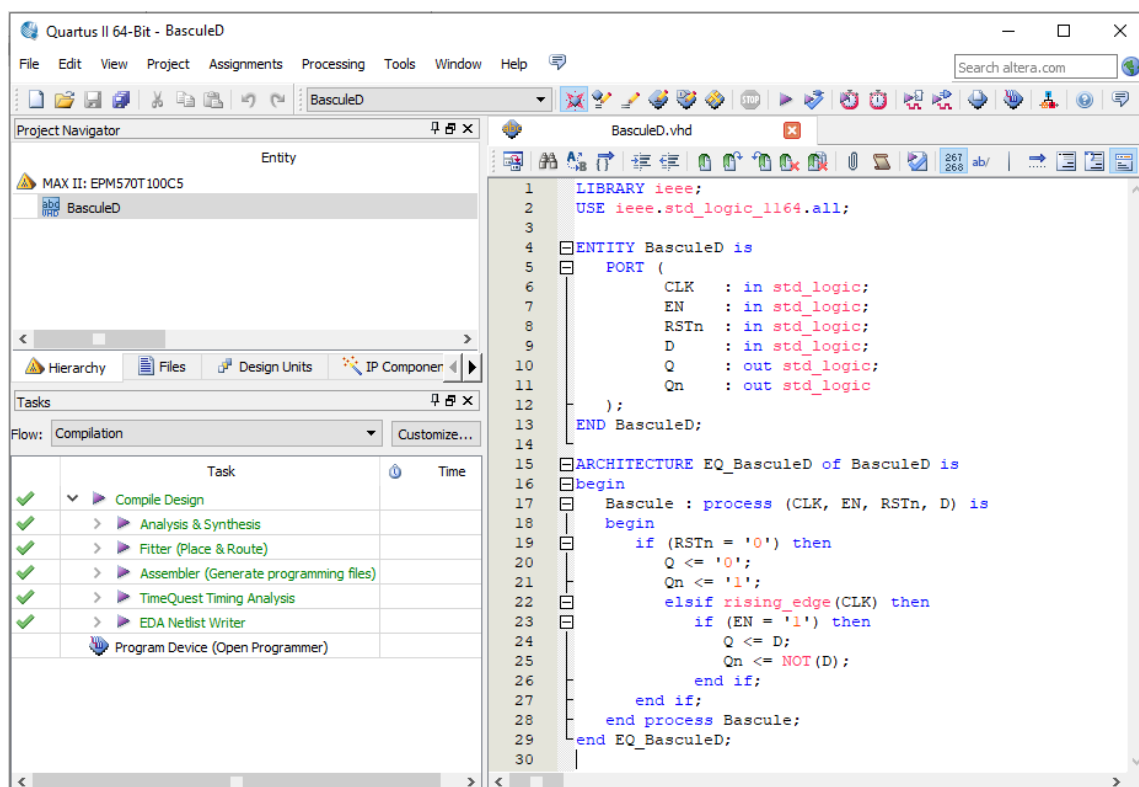


Table de vérité					
CLK	EN	RST	D	Q	$\bar{Q}$
X	0	0	X	0	1
X	1	0	X	0	1
X	0	1	X	$Q_n$	$\bar{Q}_n$
↑	1	1	0	0	1
↑	1	1	1	1	0

Équation de la sortie Q : (EN = 1 et au front montant de CLK),  $Q = D$  ;  $\bar{Q} = \bar{D}$

#### 🔗 Programme VHDL.



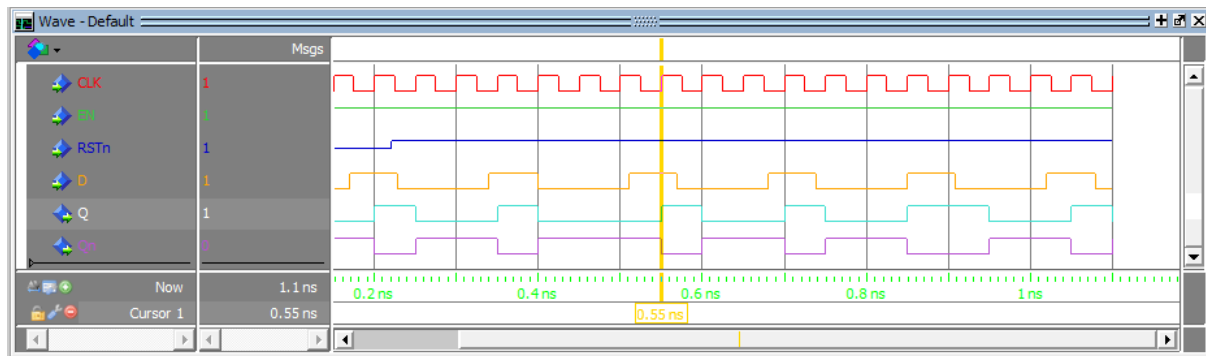
#### ➤ Affectation des broches.

Les sorties de la bascule D			
Broche N°58	IOB2-58	Commande diode led <b>L1</b>	Sortie Q
Broche N°57	IOB2-57	Commande diode led <b>L2</b>	Sortie inverse Qn

Les entrées de la bascule D.			
Broche N°29	IOB1-29	Lecture état de l'interrupteur <b>INT1</b>	Entrée EN de la bascule
Broche N°28	IOB1-28	Lecture état de l'interrupteur <b>INT2</b>	Entrée $\bar{RST}$ de la bascule
Broche N°38	IOB1-38	Lecture état du bouton poussoir <b>BP1</b>	Entrée D de la bascule
Broche N°62	IOB2-62/GCLK2	Horloge CPLD <b>40 MHz</b>	Entrée CLK de la bascule

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in CLK	Input	PIN_62	2	PIN_62	3.3-V LV...default		16mA (default)
in D	Input	PIN_38	1	PIN_38	3.3-V LV...default		16mA (default)
in EN	Input	PIN_29	1	PIN_29	3.3-V LV...default		16mA (default)
out Q	Output	PIN_58	2	PIN_58	3.3-V LV...default		16mA (default)
out Qn	Output	PIN_57	2	PIN_57	3.3-V LV...default		16mA (default)
in RSTn	Input	PIN_28	1	PIN_28	3.3-V LV...default		16mA (default)
<<new node>>							

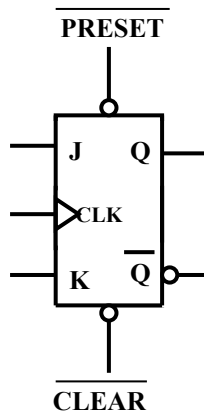
## Résultat de la simulation.



On constate que : avec  $EN = RSTn = 1$ , à chaque front montant du signal d'horloge CLK, la sortie Q recopie l'entrée de données D ( $Q = D$ ).

### 4- La bascule J-K.

#### ➤ Symbole et table de vérité.



CLK	J	K	$\overline{\text{PRESET}}$	$\overline{\text{CLEAR}}$	Q	$\overline{Q}$	Fonction
X	X	X	0	1	1	0	Mise à 1
X	X	X	1	0	0	1	Mise à 0
X	X	X	0	0	Instable		
H	X	X	1	1	Q	$\overline{Q}$	Mémoire
L	X	X	1	1	Q	$\overline{Q}$	Mémoire
↑	0	0	1	1	Q	$\overline{Q}$	Mémoire
↑	1	0	1	1	1	0	Mise à 1
↑	0	1	1	1	0	1	Mise à 0
↑	1	1	1	1	$\overline{Q}$	Q	Basculement

**Bascule synchrone** avec une entrée **J** de mise à "1" et une entrée **K** de mise à "0". Elle dispose également de deux entrées asynchrones de forçage : une entrée  $\overline{\text{PRESET}}$  active à l'état bas pour un forçage à "1" et une entrée  $\overline{\text{CLEAR}}$  active à l'état bas pour un forçage à "0".

## ☺ Le programme VHDL.

Quartus II 64-bit - BasculeJK

File Edit View Project Assignments Processing Tools Window Help

Project Navigator: Entity BasculeJK

Tasks: Compilation

Task	Time
Compile Design	00:00:11
Analysis & Synthesis	00:00:02
Fitter (Place & Route)	00:00:03
Assembler (Generate programming files)	00:00:02
TimeQuest Timing Analysis	00:00:02
EDA Netlist Writer	00:00:02
Program Device (Open Programmer)	

```

1  LIBRARY IEEE;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY BasculeJK IS
5  PORT ( CLK : IN std_logic;
6        PRE : IN std_logic;
7        CLR : IN std_logic;
8        J   : IN std_logic;
9        K   : IN std_logic;
10       Q    : OUT std_logic;
11       Qn   : OUT std_logic
12 );
13 END BasculeJK;
14
15 ARCHITECTURE EQ_BasculeJK of BasculeJK is
16     signal auxQ : std_logic;
17 begin
18     LesSignaux : process (CLK, PRE, CLR, J, K) is
19     begin
20         if ((PRE = '1') and (CLR = '0')) then
21             Q <= '1';
22             auxQ <= '1';
23             Qn <= '0';
24         elsif ((PRE = '0') and (CLR = '1')) then
25             Q <= '0';
26             auxQ <= '0';
27             Qn <= '1';
28         elsif ((PRE = '1') and (CLR = '1')) then
29             if (rising_edge (CLK)) then
30                 if ((J = '1') and (K = '0')) then
31                     Q <= '1';
32                     auxQ <= '1';
33                     Qn <= '0';
34                 elsif ((J = '0') and (K = '1')) then
35                     Q <= '0';
36                     auxQ <= '0';
37                     Qn <= '1';
38                 elsif ((J = '1') and (K = '1')) then
39                     Q <= not auxQ;
40                     Qn <= auxQ;
41                 end if;
42             end if;
43         end if;
44     end process;
45 end EQ_BasculeJK;
46

```

### ☞ Affectation des broches.

#### Les sorties de la bascule J-K

Broche N°58	IOB2-58	Commande diode led <b>L1</b>	Sortie Q
Broche N°57	IOB2-57	Commande diode led <b>L2</b>	Sortie inverse Qn

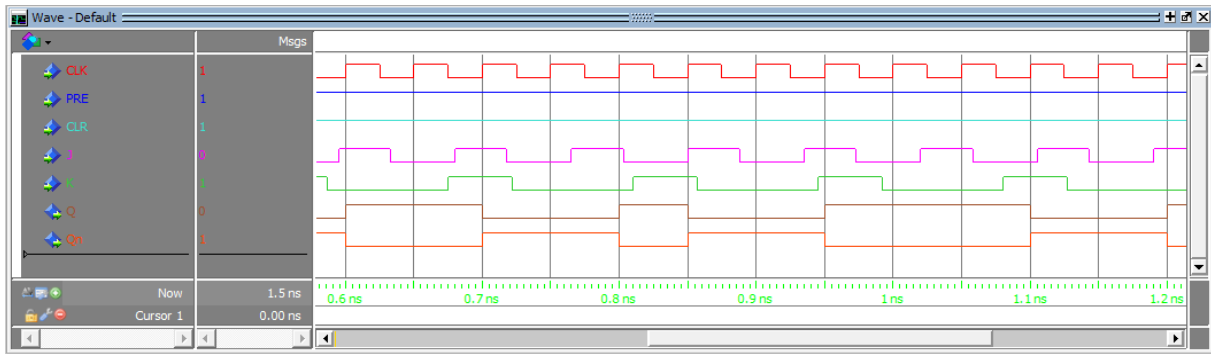
#### Les entrées de la bascule J-K.

Broche N°29	IOB1-29	Lecture état de l'interrupteur <b>INT1</b>	Entrée $\overline{\text{PRE}}$ de la bascule
Broche N°28	IOB1-28	Lecture état de l'interrupteur <b>INT2</b>	Entrée CLR de la bascule
Broche N°38	IOB1-38	Lecture état du bouton poussoir <b>BP1</b>	Entrée J de la bascule
Broche N°36	IOB1-36	Lecture état du bouton poussoir <b>BP2</b>	Entrée K de la bascule
Broche N°62	IOB2-62/GCLK2	Horloge CPLD 40 MHz	Entrée CLK de la bascule

Named: * Edit: Filter: Pins: all							
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in CLK	Input	PIN_62	2	PIN_62	3.3-V LV...default		16mA (default)
in CLR	Input	PIN_28	1	PIN_28	3.3-V LV...default		16mA (default)
in J	Input	PIN_38	1	PIN_38	3.3-V LV...default		16mA (default)
in K	Input	PIN_36	1	PIN_36	3.3-V LV...default		16mA (default)
in PRE	Input	PIN_29	1	PIN_29	3.3-V LV...default		16mA (default)
out Q	Output	PIN_58	2	PIN_58	3.3-V LV...default		16mA (default)
out Qn	Output	PIN_57	2	PIN_57	3.3-V LV...default		16mA (default)
<<new node>>							



## Résultat de la simulation.



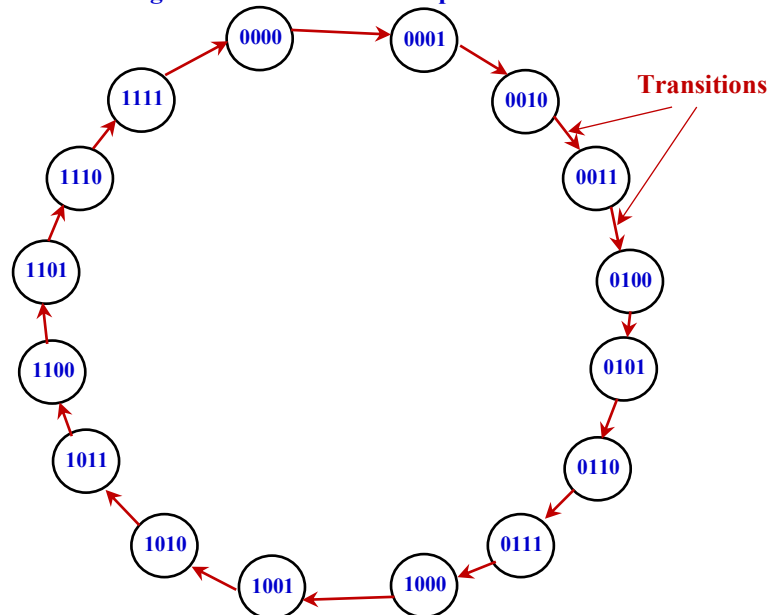
## LES COMPTEURS.

### 1- Compteur synchrone à cycle complet Modulo-16.

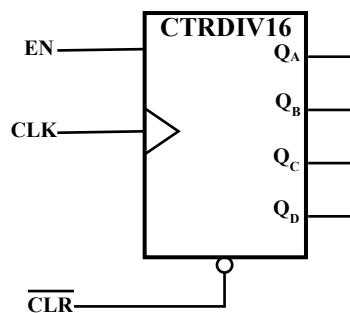
Il s'agit d'un compteur à  $16 = 2^4$  états distincts. Il utilise 4 bascules synchrones A, B, C et D câblés en diviseur de fréquence par 2. Les sorties des 4 bascules sont  $Q_A$ ,  $Q_B$ ,  $Q_C$  et  $Q_D$ .

N°	$Q_D$	$Q_C$	$Q_B$	$Q_A$	États
0	0	0	0	0	0000
1	0	0	0	1	0001
2	0	0	1	0	0010
3	0	0	1	1	0011
4	0	1	0	0	0100
5	0	1	0	1	0101
6	0	1	1	0	0110
7	0	1	1	1	0111
8	1	0	0	0	1000
9	1	0	0	1	1001
10	1	0	1	0	1010
11	1	0	1	1	1011
12	1	1	0	0	1100
13	1	1	0	1	1101
14	1	1	1	0	1110
15	1	1	1	1	1111

Diagramme des états du compteur.



☞ Le passage d'un état au suivant, appelé **transition**, se fait sur une **transition valide** (front montant ou descendant) du signal d'horloge.



- **EN** = 1 : Comptage
- **EN** = 0 : Arrêt comptage
- **CLR** = 0 : Mise à 0 du compteur  $Q_A = Q_B = Q_C = Q_D = 0$

## 😊 Programme VHDL.

```

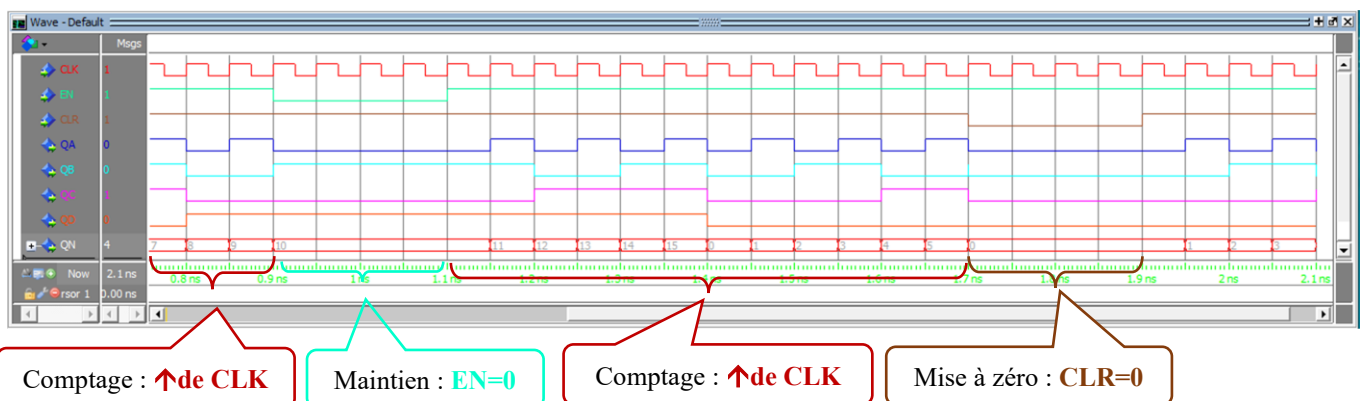
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_arith.all;
4  USE ieee.std_logic_unsigned.all;
5
6  ENTITY CompteurMod16 IS
7  PORT ( CLK : IN std_logic; -- Signal d'horloge
8        EN : IN std_logic; -- Validation du comptage
9        CLR : IN std_logic; -- Mise à zéro du compteur
10       QN : OUT std_logic_vector(3 downto 0) -- Les sorties du compteur
11       );
12  END CompteurMod16;
13
14  ARCHITECTURE Comptage of CompteurMod16 is
15  signal aux : std_logic_vector(3 downto 0); -- Signaux intermédiaires
16  BEGIN
17  process (CLK, CLR, EN)
18  BEGIN
19      if ((CLR = '0') and (EN = '1')) then
20          aux <= (others => '0'); -- Mise à zéro du compteur
21      elsif ((CLR = '1') and (EN = '0')) then
22          aux <= aux; -- Mémoire
23      elsif (rising_edge (CLK)) then
24          aux <= aux + 1; -- Incréméntation du compteur
25      end if;
26  end process;
27  QN <= aux;
28  END Comptage;
29
30

```

## 🔧 Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default		16mA (default)
in CLR	Input	PIN_28	1	PIN_28	3.3-V LV...default		16mA (default)
in EN	Input	PIN_29	1	PIN_29	3.3-V LV...default		16mA (default)
out QN[3]	Output	PIN_53	2	PIN_53	3.3-V LV...default		16mA (default)
out QN[2]	Output	PIN_54	2	PIN_54	3.3-V LV...default		16mA (default)
out QN[1]	Output	PIN_57	2	PIN_57	3.3-V LV...default		16mA (default)
out QN[0]	Output	PIN_58	2	PIN_58	3.3-V LV...default		16mA (default)

## Résultat de la simulation.

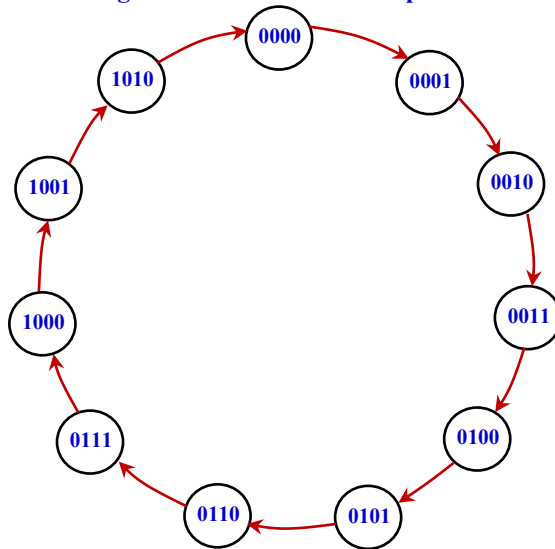


## 2- Compteur synchrone à cycle incomplet : compteur Modulo 11.

☞ Compteur à 11 états distincts.

Diagramme des états du compteur.

N°	Q <sub>D</sub>	Q <sub>C</sub>	Q <sub>B</sub>	Q <sub>A</sub>	États
0	0	0	0	0	0000
1	0	0	0	1	0001
2	0	0	1	0	0010
3	0	0	1	1	0011
4	0	1	0	0	0100
5	0	1	0	1	0101
6	0	1	1	0	0110
7	0	1	1	1	0111
8	1	0	0	0	1000
9	1	0	0	1	1001
10	1	0	1	0	1010
11	1	0	1	1	1011

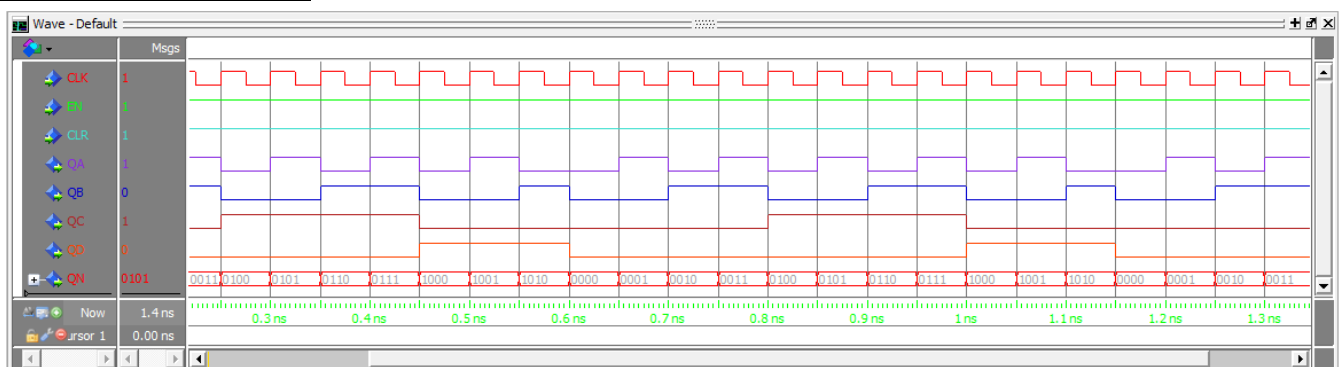


### Programme VHDL.

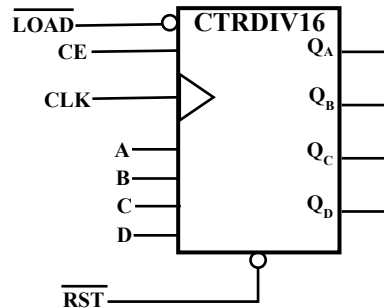
```

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 CompteurMod11 IS
7  PORT (
8    CLK : IN std_logic; -- Signal d'horloge
9    EN : IN std_logic; -- Validation du comptage
10   CLR : IN std_logic; -- Mise à zéro du compteur
11   QN : OUT std_logic_vector(3 downto 0) -- Les sorties du compteur
12 );
13 END CompteurMod11;
14
15 ARCHITECTURE Comptage of CompteurMod11 is
16   signal aux : std_logic_vector(3 downto 0); -- Signaux intermédiaires
17   begin
18     process (CLK, CLR, EN)
19     begin
20       if ((CLR = '0') and (EN = '1')) then
21         aux <= (others => '0'); -- Mise à zéro du compteur
22       elsif ((CLR = '1') and (EN = '0')) then
23         aux <= aux; -- Mémoire
24       elsif (rising_edge (CLK)) then
25         aux <= aux + 1; -- Incréméntation du compteur
26         if (aux = "1010") then -- Si valeur 10 atteinte
27           aux <= (others => '0'); -- alors mise à zéro
28         end if;
29       end if;
30       QN <= aux;
31     end process;
32   end Comptage;
  
```

### Résultat de la simulation.



### 3- Compteur synchrone Modulo-16 avec chargement parallèle.



- **CLK** : Signal d'horloge
- **LOAD** = 0 : Chargement.  $Q_A = A$  ;  $Q_B = B$  ;  $Q_C = C$  ;  $Q_D = D$
- **LOAD** = 1 : Comptage
- **RST** = 0 : Mise à 0 du compteur  $Q_A = Q_B = Q_C = Q_D = 0$

#### Programme VHDL

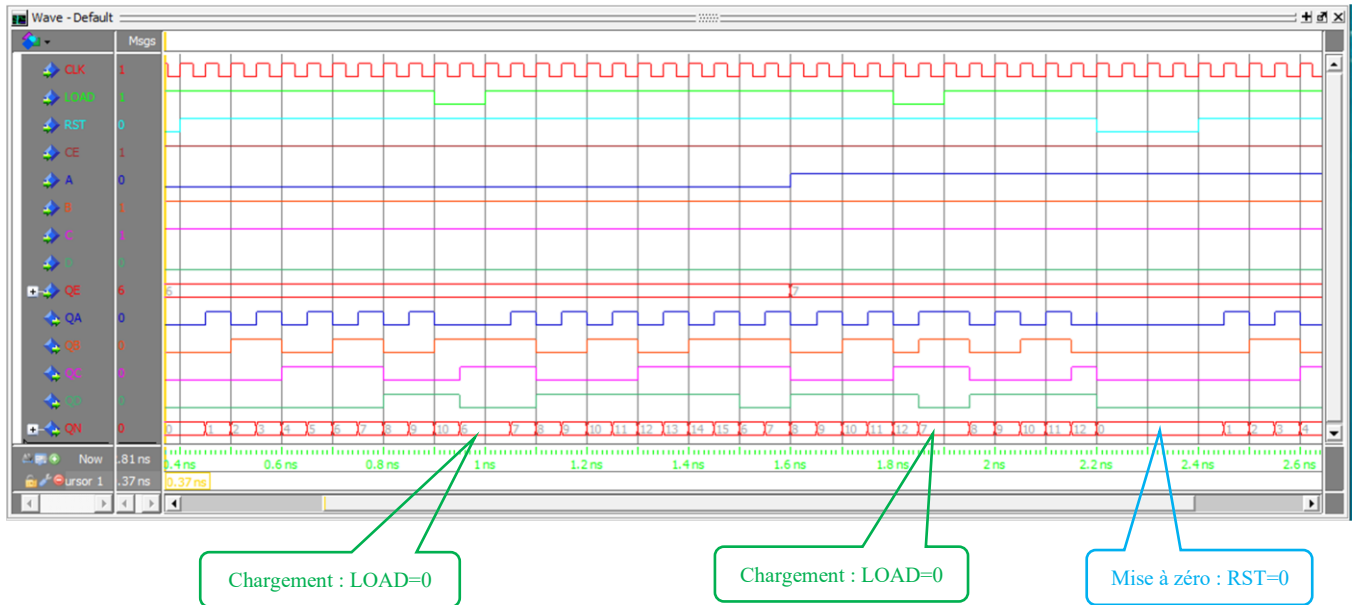
```

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 CompteurChargement is
7  PORT (
8      CLK : IN    std_logic; -- Signal d'horloge
9      LOAD : IN   std_logic; -- Signal de chargement
10     RST : IN   std_logic; -- Mise à zéro du compteur
11     CE : IN   std_logic; -- Signal de validation du compteur
12     QN : OUT  std_logic_vector(3 downto 0); -- Données de chargement
13 );
14 END CompteurChargement;
15
16 ARCHITECTURE Comptage of CompteurChargement is
17     signal aux : std_logic_vector(3 downto 0); -- Signal intermédiaire
18 BEGIN
19     process(CLK, RST, LOAD)
20     BEGIN
21         if (RST = '0') then
22             aux <= (others => '0');
23         elsif (rising_edge (CLK)) then
24             if (LOAD = '0') then -- Si LOAD = 0 on charge le compteur
25                 aux <= QE; -- Chargement du compteur
26             elsif (CE = '1') then
27                 -- if (aux = "1111") then aux <= (others => '0');
28                 if (aux = "1111") then -- Si fin du cycle on recharge
29                     aux <= QE; -- Rechargement du compteur en fin de cycle
30                 else aux <= aux + 1; -- Si pas fin de cycle on incrémente
31                 end if;
32             end if;
33         end process;
34         QN <= aux;
35     END Comptage;
36 
```

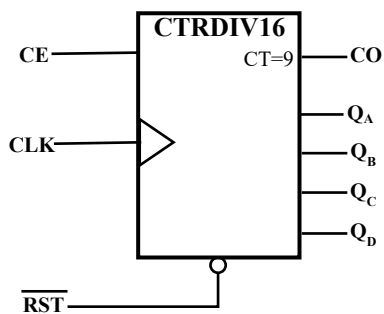
#### Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
CE	Input	PIN_36	1	PIN_36	3.3-V LV...default		16mA (default)
CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default		16mA (default)
LOAD	Input	PIN_38	1	PIN_38	3.3-V LV...default		16mA (default)
QE[3]	Input	PIN_26	1	PIN_26	3.3-V LV...default		16mA (default)
QE[2]	Input	PIN_27	1	PIN_27	3.3-V LV...default		16mA (default)
QE[1]	Input	PIN_28	1	PIN_28	3.3-V LV...default		16mA (default)
QE[0]	Input	PIN_29	1	PIN_29	3.3-V LV...default		16mA (default)
QN[3]	Output	PIN_53	2	PIN_53	3.3-V LV...default		16mA (default)
QN[2]	Output	PIN_54	2	PIN_54	3.3-V LV...default		16mA (default)
QN[1]	Output	PIN_57	2	PIN_57	3.3-V LV...default		16mA (default)
QN[0]	Output	PIN_58	2	PIN_58	3.3-V LV...default		16mA (default)
RST	Input	PIN_34	1	PIN_34	3.3-V LV...default		16mA (default)
<<new node>>							

## Résultat de la simulation.



## 4- COMPTEUR SYNCHRONES À DÉCADE (Modulo-10) : Compteur Binaire Codé Décimal BCD.



- **CLK** : Signal d'horloge
- **CO** : Sortie retenue = 1 si le compteur est en fin de cycle. C'est-à-dire si  $Q_A = 1$  ;  $Q_B = 0$  ;  $Q_C = 0$  ;  $Q_D = 1$ .
- **CO** : Sortie retenue = 0 si  $Q_A = Q_B = Q_C = Q_D = 0$ .
- **$\overline{RST}$**  = 0 : Mise à 0 du compteur  $Q_A = Q_B = Q_C = Q_D = 0$ .



## Programme VHDL.

```

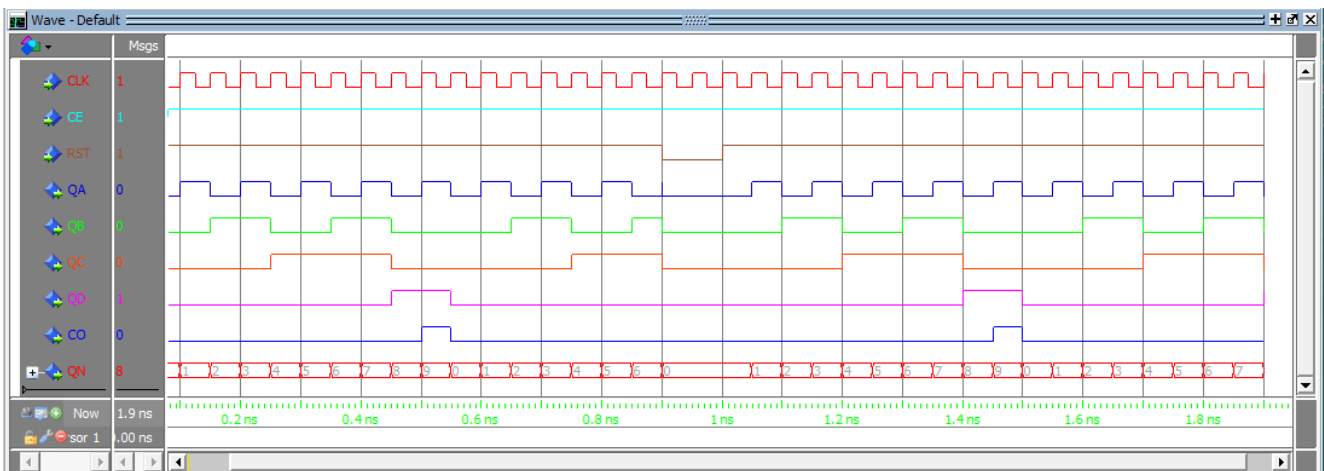
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_arith.all;
4  USE ieee.std_logic_unsigned.all;
5
6  ENTITY CompteurMod10RCO is
7  PORT ( CLK : IN    std_logic; --Signal d'horloge
8        CE  : IN    std_logic; --Entrée de validation du compteur
9        RST : IN    std_logic; -- Entrée de mise à zéro
10       CO  : OUT   std_logic; -- Sortie retenue: fin comptage
11       QN : OUT   std_logic_vector(3 downto 0) --Les sorties du compteur
12     );
13 END CompteurMod10RCO;
14
15 ARCHITECTURE Comptage of CompteurMod10RCO is
16     signal aux : std_logic_vector(3 downto 0); --Sinaux intermédiaires
17 BEGIN
18
19     process (CLK, CE, RST) -- Cycle de comptage
20     begin
21         if (RST = '0') then
22             aux <= "0000";
23         elsif (rising_edge (CLK)) then
24             if (CE = '1') then
25                 if (aux = "1001") then
26                     aux <= "0000";
27                 else
28                     aux <= aux + 1;
29                 end if;
30             end if;
31         end process; -- Fin cycle de comptage
32         QN <= aux;
33         process (aux, CE) -- Ce processus génère le signal de fin de cycle: Carry Out (CO)
34         begin
35             if (aux = "1001") then -- Si Q0=0; Q1=0; Q2=0; Q3=1
36                 CO <= '1'; -- CO = 1
37             else -- Sinon
38                 CO <= '0'; -- CO = 0
39             end if;
40         end process;
41     end Comptage;
42

```

## Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
CE	Input	PIN_38	1	PIN_38	3.3-V LV...default		16mA (default)
CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default		16mA (default)
CO	Output	PIN_49	1	PIN_49	3.3-V LV...default		16mA (default)
QN[3]	Output	PIN_53	2	PIN_53	3.3-V LV...default		16mA (default)
QN[2]	Output	PIN_54	2	PIN_54	3.3-V LV...default		16mA (default)
QN[1]	Output	PIN_57	2	PIN_57	3.3-V LV...default		16mA (default)
QN[0]	Output	PIN_58	2	PIN_58	3.3-V LV...default		16mA (default)
RST	Input	PIN_36	1	PIN_36	3.3-V LV...default		16mA (default)
<<new node>>							

## Résultat de la simulation.

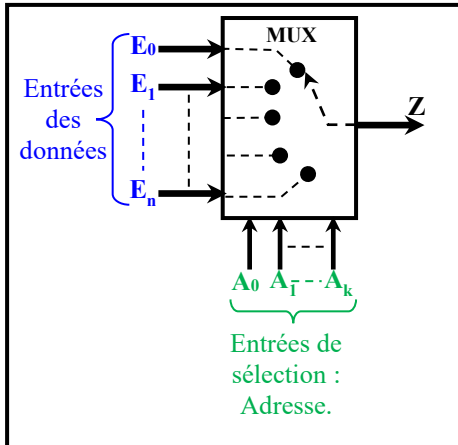


## LES MULTIPLEXEURS – Exemples.

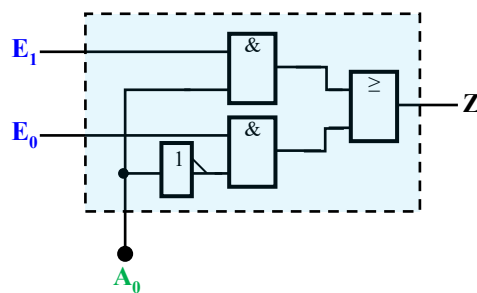
Un multiplexeur (MUX) ou sélecteur de données est un circuit logique ayant :

- Plusieurs entrées de données **E0, E1, E2, E3 ...**,
- Une seule sortie **Z** qui communique les données.

L'aiguillage de l'entrée de données qui nous intéresse sur la sortie est commandé par les entrées de sélections, appelées parfois entrées d'Adresse.



**Exemple :** Multiplexeur élémentaire à deux entrées **E0, E1** ou sélecteur de données 1 parmi 2.



$$Z = E_1 \cdot A_0 + E_0 \cdot \overline{A_0}$$

Table de vérité

A <sub>0</sub>	Z
0	E <sub>0</sub>
1	E <sub>1</sub>

**Autre exemple :** Multiplexeur à 4 entrées **E0, E1, E2, E3** ou sélecteur de données 1 parmi 4.

Pour sélectionner une entrée parmi 4, on a besoin de 2 entrées d'adresse **A0** et **A1**.

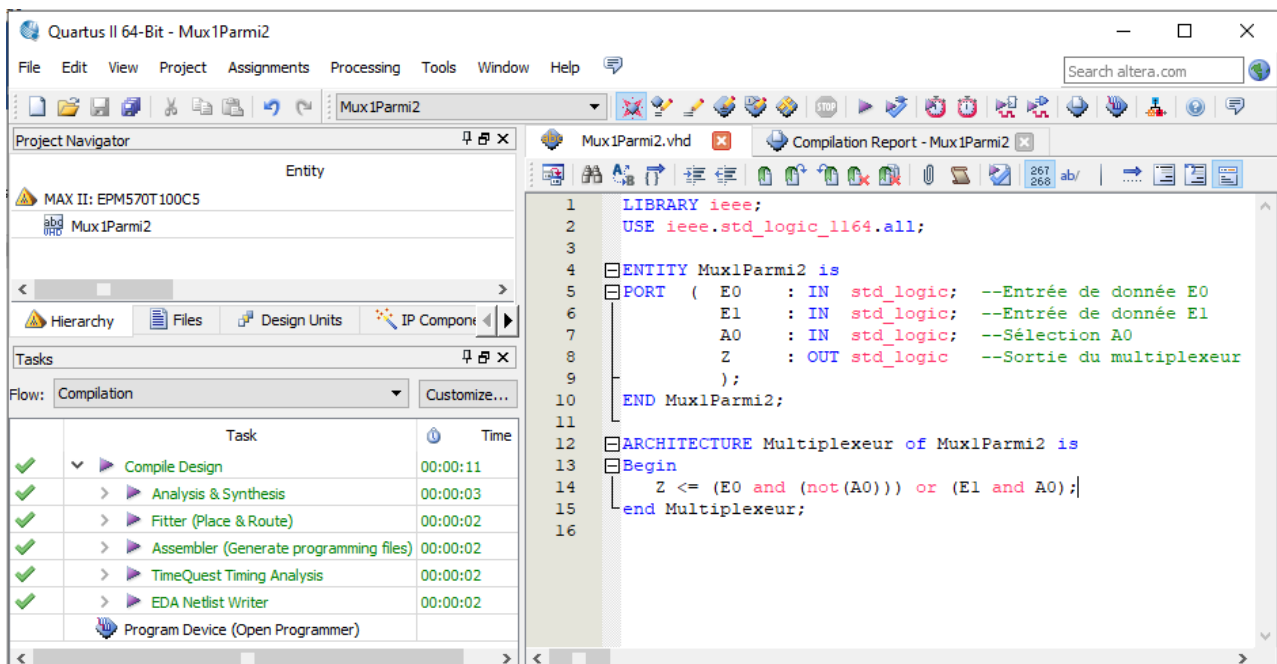
Table de vérité

A <sub>1</sub>	A <sub>0</sub>	Z
0	0	E <sub>0</sub>
0	1	E <sub>1</sub>
1	0	E <sub>2</sub>
1	1	E <sub>3</sub>

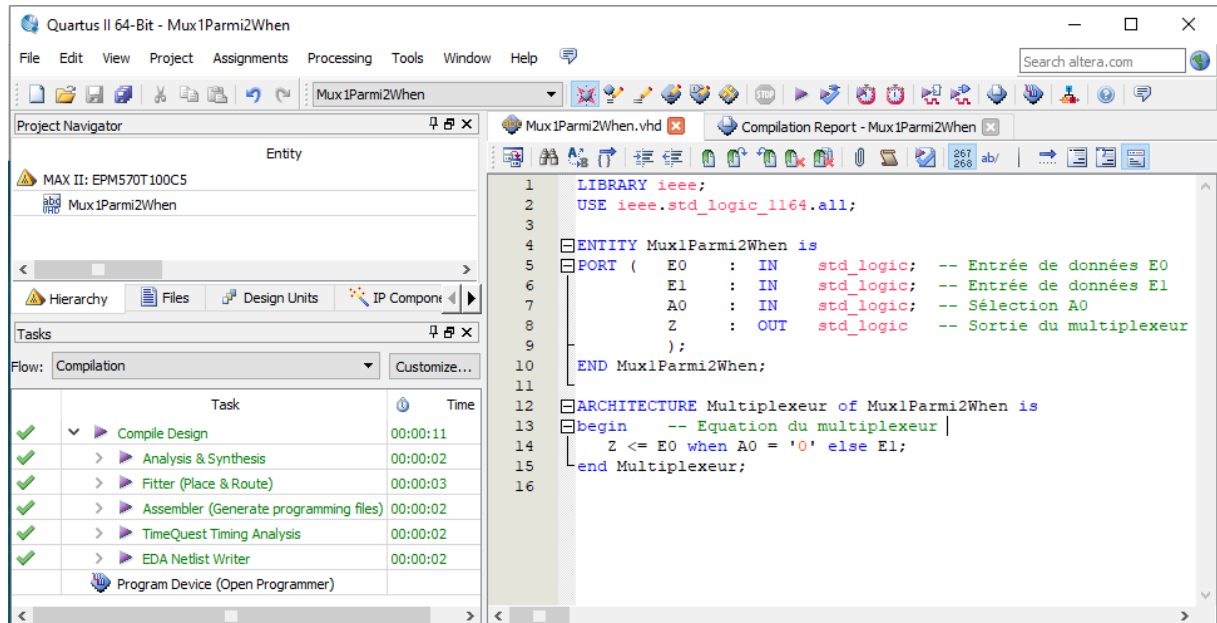
Équation de la sortie Z du multiplexeur.

$$Z = E_0 \cdot \overline{A_0} \cdot \overline{A_1} + E_1 \cdot A_0 \cdot \overline{A_1} + E_2 \cdot \overline{A_0} \cdot A_1 + E_3 \cdot A_0 \cdot A_1$$

### Programme VHDL : Multiplexeur à 2 entrées.



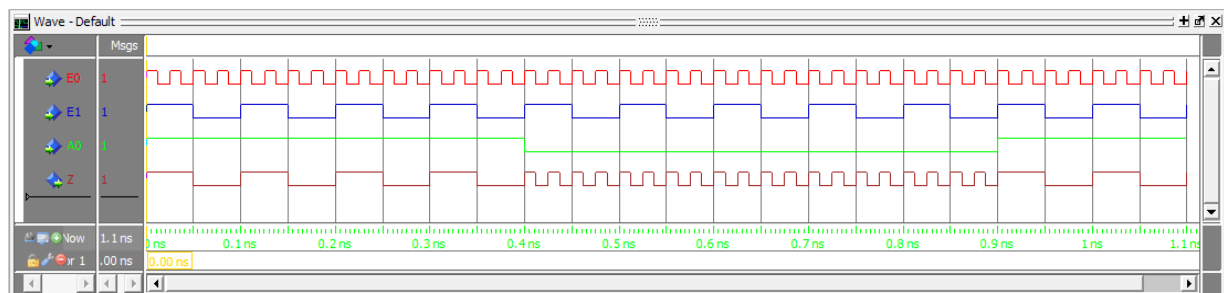
## ☺ Autre version utilisant l'instruction "when ... else".



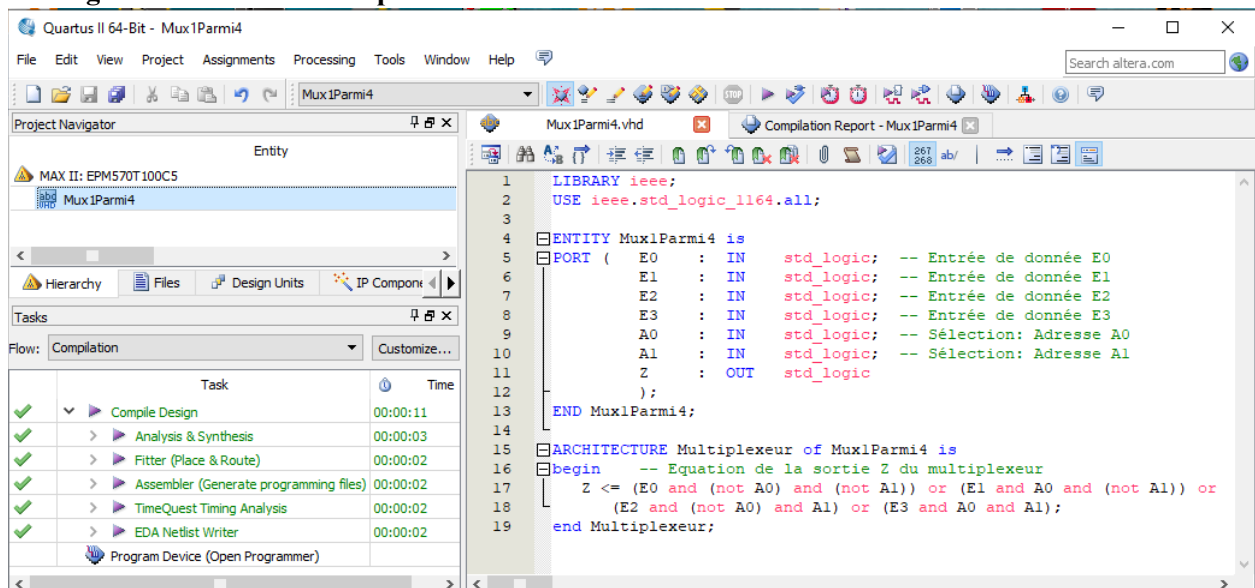
## Affectation des broches.

Named: *	Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
	A0	Input	PIN_29	1	PIN_29	3.3-V LV..default)		16mA (default)
	E0	Input	PIN_38	1	PIN_38	3.3-V LV..default)		16mA (default)
	E1	Input	PIN_12	1	PIN_12	3.3-V LV..default)		16mA (default)
	Z	Output	PIN_58	2	PIN_58	3.3-V LV..default)		16mA (default)
	<<new node>>							

## Résultat de la simulation.



## 🔗 Programme VHDL : Multiplexeur à 4 entrées de données.



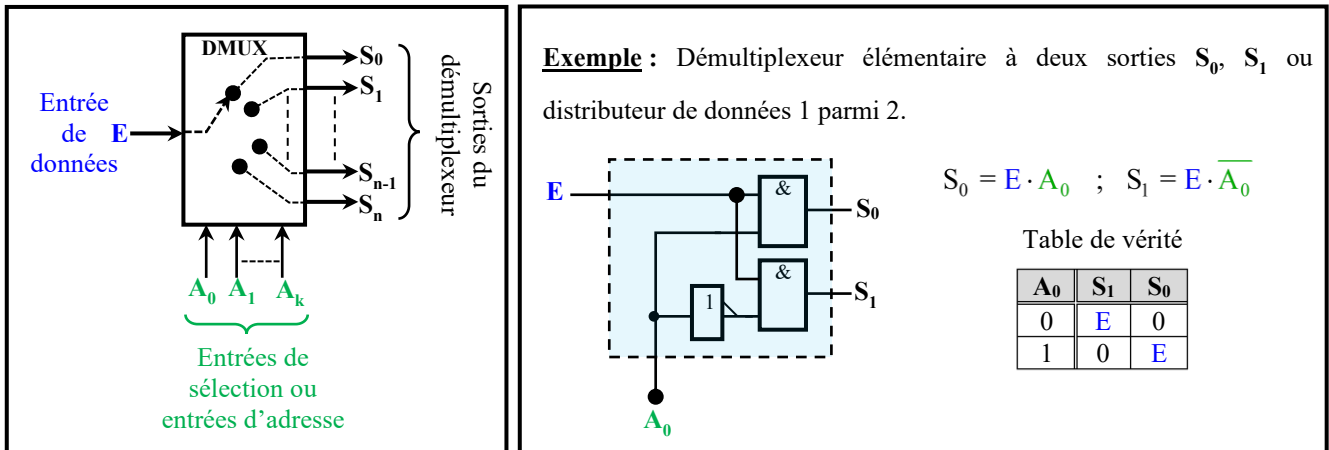




## LES DÉMULTIPLEXEURS – Exemples.

Un démultiplexeur (DMUX) ou distributeur de données est un circuit n'ayant qu'une entrée de donnée **E** et plusieurs sorties **S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub>, S<sub>3</sub> ....**

L'aiguillage de la donnée d'entrée vers la sortie désirée se fait grâce à la valeur du mot binaire présente sur les entrées de sélection encore appelées entrées d'adresse **A<sub>0</sub>, A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub> ....**



**Autre exemple :** Démultiplexeur à 4 sorties **S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub>, S<sub>3</sub>** ou distributeur de données 1 parmi 4.

Pour sélectionner une sortie parmi 4, on a besoin de 2 entrées d'adresse **A<sub>0</sub>** et **A<sub>1</sub>**.

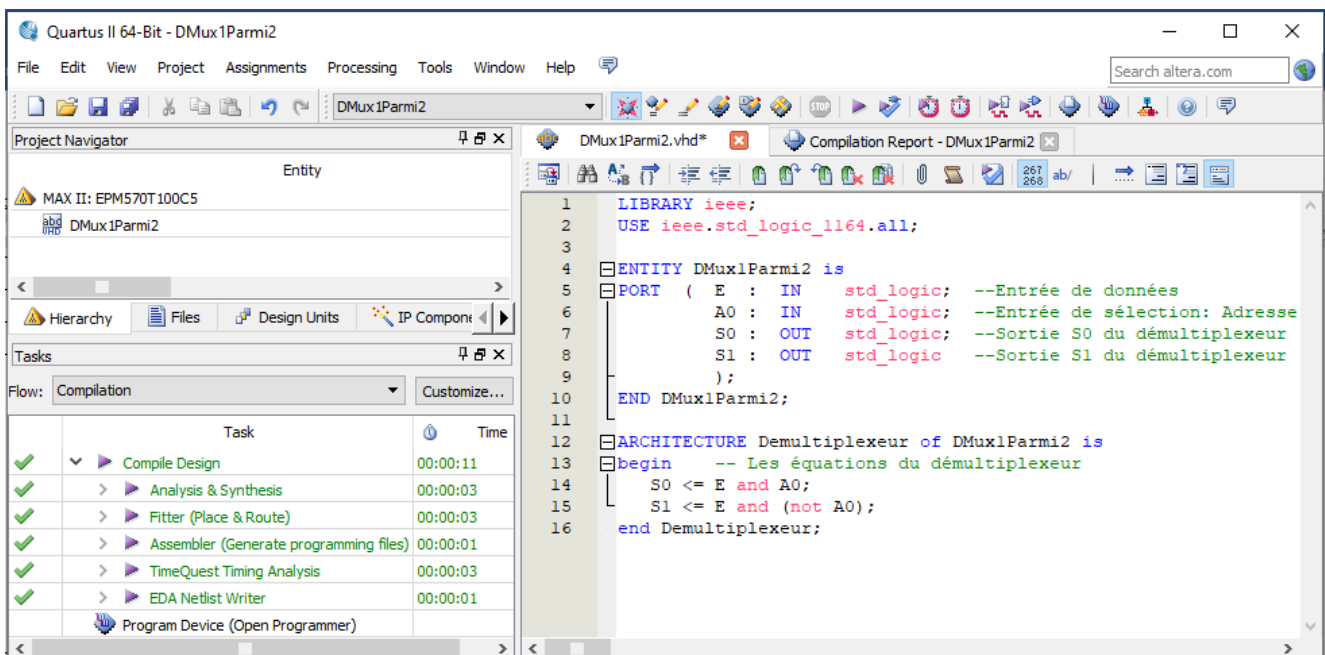
Table de vérité

A <sub>1</sub>	A <sub>0</sub>	S <sub>3</sub>	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>
0	0	0	0	0	E
0	1	0	0	E	0
1	0	0	E	0	0
1	1	E	0	0	0

Équations des sorties du démultiplexeur.

$$S_0 = E \cdot \overline{A_0} \cdot \overline{A_1} \quad ; \quad S_1 = E \cdot A_0 \cdot \overline{A_1} \quad ; \quad S_2 = E \cdot \overline{A_0} \cdot A_1 \quad ; \quad S_3 = E \cdot A_0 \cdot A_1$$

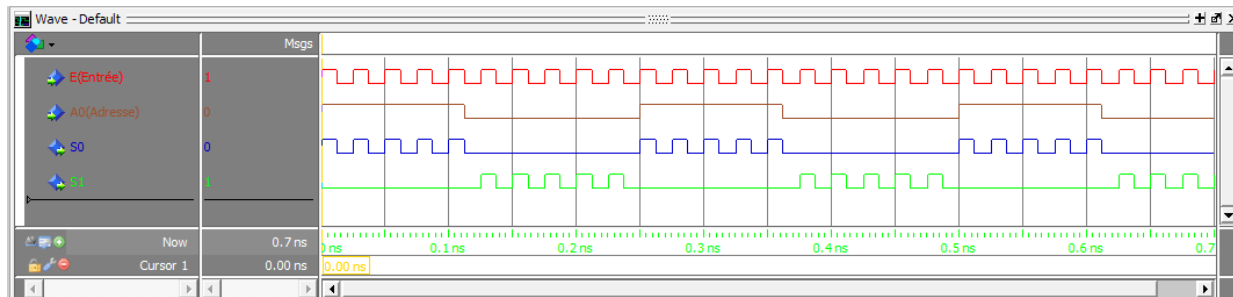
### Programme VHDL : Démultiplexeur à 2 sorties.



## Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in A0	Input	PIN_29	1	PIN_29	3.3-V LV...default)		16mA (default)
in E	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
out S0	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)
out S1	Output	PIN_57	2	PIN_57	3.3-V LV...default)		16mA (default)
<<new node>>							

## Résultat de la simulation.



## Programme VHDL : Démultiplexeur à 4 sorties.

```

Quartus II 64-Bit - DMux1Parmi4
File Edit View Project Assignments Processing Tools Window Help
DMux1Parmi4.vhd
Compilation Report - DMux1Parmi4

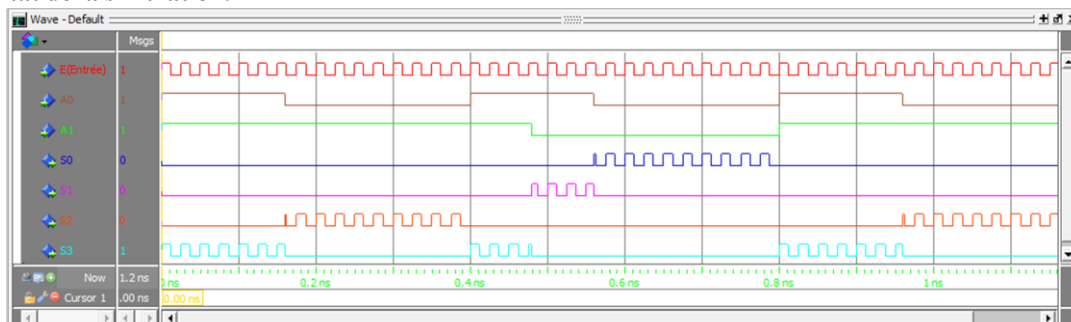
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY DMux1Parmi4 is
5  PORT (
6      E : IN    std_logic; -- Entrée de données
7      A0 : IN    std_logic; -- Entrée d'adresse: poids faible A0
8      A1 : IN    std_logic; -- Entrée d'adresse: poids fort A1
9      S0 : OUT   std_logic; -- Première sortie du démultiplexeur
10     S1 : OUT   std_logic; -- Deuxième sortie du démultiplexeur
11     S2 : OUT   std_logic; -- Troisième sortie du démultiplexeur
12     S3 : OUT   std_logic; -- Dernière sortie du démultiplexeur
13 );
14 END DMux1Parmi4;
15
16 ARCHITECTURE Demultiplexeur of DMux1Parmi4 is
17 BEGIN
18     -- Les équations des sorties du démultiplexeur
19     S0 <= (E and (not A0) and (not A1));
20     S1 <= (E and A0 and (not A1));
21     S2 <= (E and (not A0) and A1);
22     S3 <= (E and A0 and A1);
23 end Demultiplexeur;

```

## Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in A0	Input	PIN_29	1	PIN_29	3.3-V LV...default)		16mA (default)
in A1	Input	PIN_28	1	PIN_28	3.3-V LV...default)		16mA (default)
in E	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
out S0	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)
out S1	Output	PIN_57	2	PIN_57	3.3-V LV...default)		16mA (default)
out S2	Output	PIN_54	2	PIN_54	3.3-V LV...default)		16mA (default)
out S3	Output	PIN_53	2	PIN_53	3.3-V LV...default)		16mA (default)
<<new node>>							

## Résultat de la simulation.



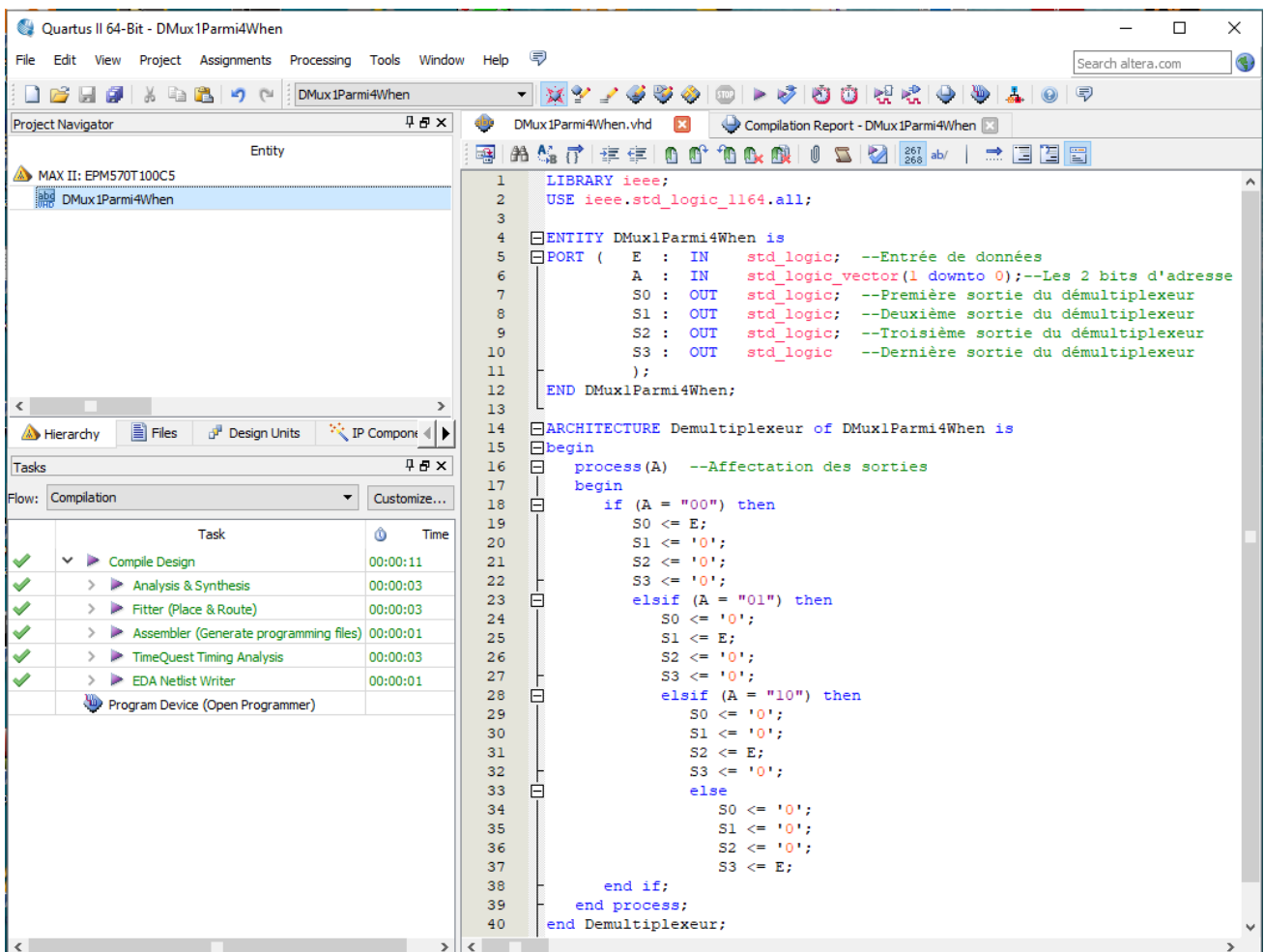
☺ Autre version utilisant l'instruction "if ... then ... else ... end if".

Il s'agit d'une instruction séquentielle. Elle est interne à un processus (process), à une procédure et à une fonction.

**Syntaxe :**

```

if expression_logique then
    instructions séquentielles;
    .....;
[ elsif expression_logique then ]
    instructions séquentielles;
[ else ]
    instructions séquentielles;
    .....;
end if;
  
```



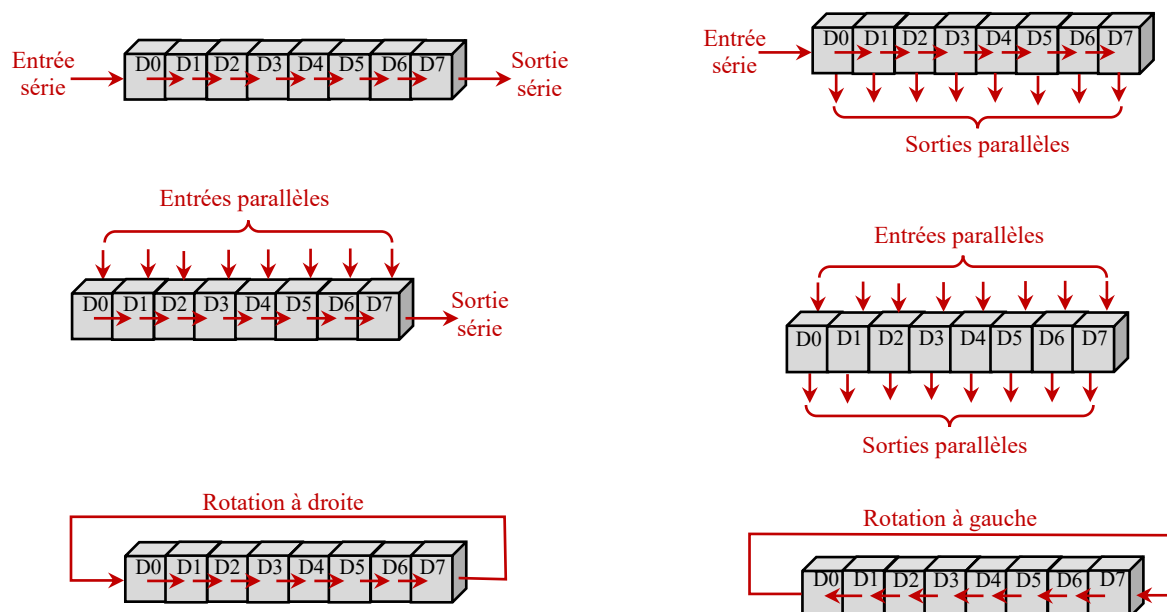
## LE REGISTRE A DÉCALAGE.

Un registre à décalage est une associations série de bascules ayant toutes le même signal d'horloge. Le registre à décalage est un élément important dans les applications de stockage et de transfert de données.

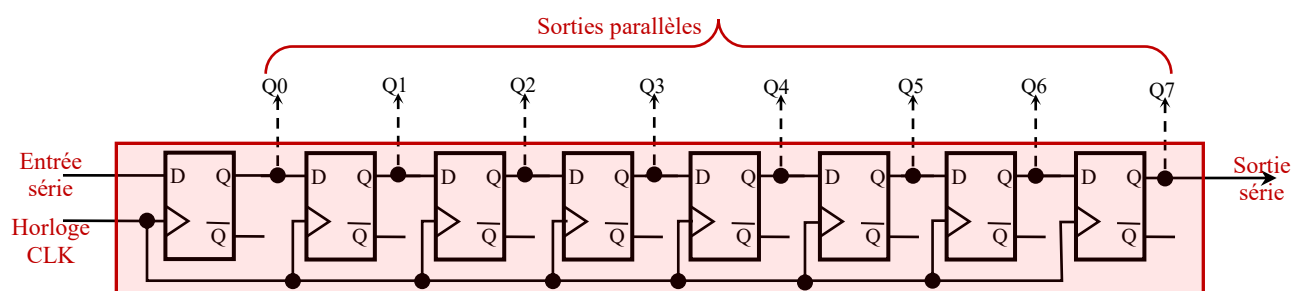
La capacité de stockage d'un registre est le nombre total de bits de données qu'il peut emmagasiner. Chaque étage (ou bascule) représente une capacité de stockage de 1 bit. Un registre à décalage 8 bits est donc une association série de 8 bascules.

Le stockage ou le transfert de données dans un registre peut se faire en série ou en parallèle.

### Déplacement des données dans les registres à décalage.



**Exemple :** Registre à décalage 8 bits – Entrée série et Sortie série ou parallèle.



## ☺ Programme VHDL. Entrée série – Sorties parallèles.

```

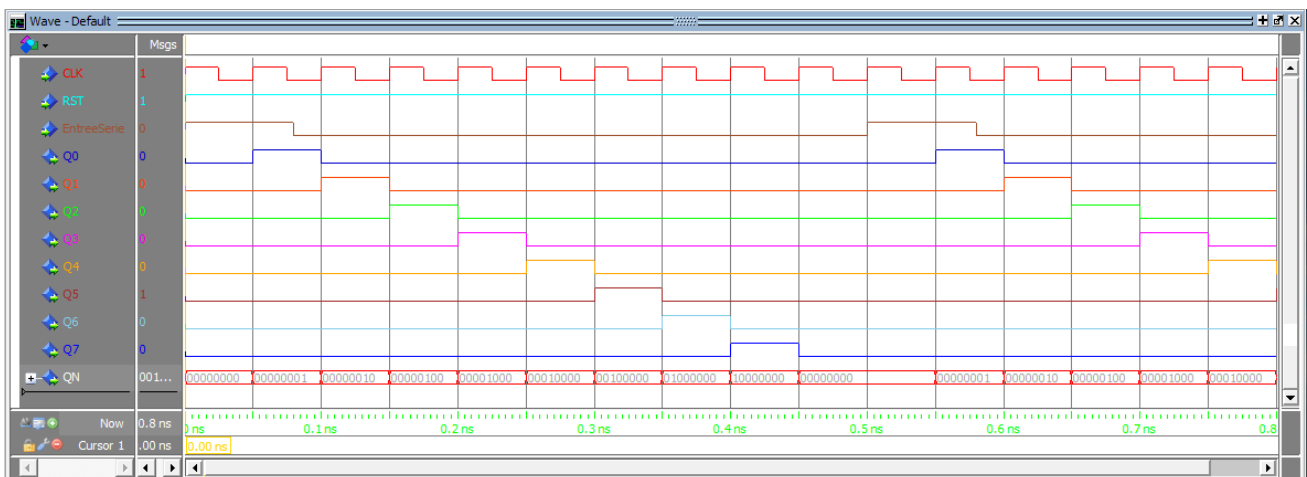
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.numeric_std.all;
4
5  ENTITY RegistreDecalage8bits IS
6  PORT ( CLK : IN std_logic; -- Signal d'horloge
7        EntreeSerie : IN std_logic; -- Entrée série de données
8        RST : IN std_logic; -- Mise à zéro du registre
9        QN : OUT std_logic_vector(7 downto 0) -- Sorties parallèles
10 );
11 END RegistreDecalage8bits;
12
13 ARCHITECTURE Registre OF RegistreDecalage8bits IS
14     signal aux : std_logic_vector(7 downto 0); -- Signaux intermédiaires
15 BEGIN
16     Reg : process (CLK)
17     BEGIN
18         IF (rising_edge (CLK)) THEN
19             IF (RST = '0') THEN -- Mise à zéro du registre
20                 aux <= (others => '0');
21             ELSE
22                 aux <= aux(6 downto 0) & EntreeSerie; -- Décalage à gauche
23             END IF;
24         END IF;
25     END process;
26     Charge : process (aux)
27     BEGIN
28         QN <= aux;
29     END process;
30 END Registre;

```

## ☞ Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default		16mA (default)
in EntreeSerie	Input	PIN_29	1	PIN_29	3.3-V LV...default		16mA (default)
out QN[7]	Output	PIN_49	1	PIN_49	3.3-V LV...default		16mA (default)
out QN[6]	Output	PIN_50	1	PIN_50	3.3-V LV...default		16mA (default)
out QN[5]	Output	PIN_51	1	PIN_51	3.3-V LV...default		16mA (default)
out QN[4]	Output	PIN_52	2	PIN_52	3.3-V LV...default		16mA (default)
out QN[3]	Output	PIN_53	2	PIN_53	3.3-V LV...default		16mA (default)
out QN[2]	Output	PIN_54	2	PIN_54	3.3-V LV...default		16mA (default)
out QN[1]	Output	PIN_57	2	PIN_57	3.3-V LV...default		16mA (default)
out QN[0]	Output	PIN_58	2	PIN_58	3.3-V LV...default		16mA (default)
in RST	Input			PIN_56	3.3-V LV...default		16mA (default)
<<new node>>							

## Résultat de la simulation.





## ☺ Programme VHDL. Entrée série – Sortie série.

Quartus II 64-Bit - RegistreDecalageSerSer8bits

File Edit View Project Assignments Processing Tools Window Help

Search altera.com

Project Navigator

Entity

MAX II: EPM570T100C5

RegistreDecalageSerSer8bits

Tasks

Flow: Compilation

Task Time

- Compile Design 00:00:12
- Analysis & Synthesis 00:00:03
- Fitter (Place & Route) 00:00:03
- Assembler (Generate programming files) 00:00:02
- TimeQuest Timing Analysis 00:00:02
- EDA Netlist Writer 00:00:02
- Program Device (Open Programmer)

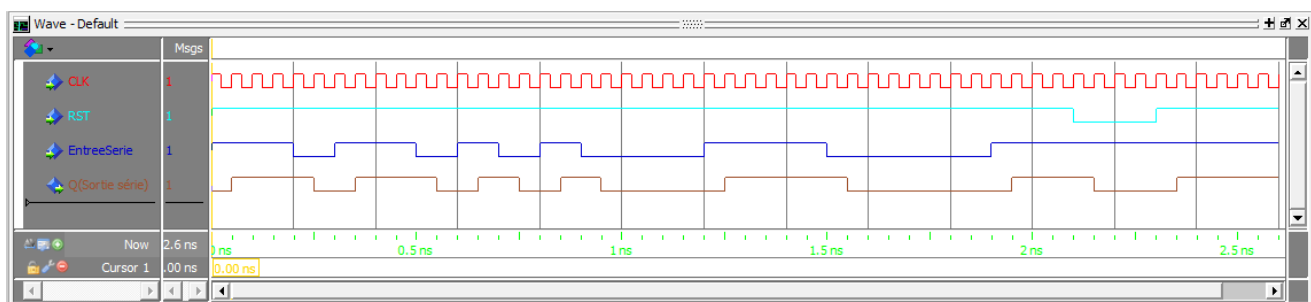
```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.numeric_std.all;
4
5  ENTITY RegistreDecalageSerSer8bits is
6  PORT ( CLK : IN      std_logic; -- Signal d'horloge
7        EntreeSerie : std_logic; -- Entrée série
8        RST : IN      std_logic; -- Mise à zéro du registre
9        Q : OUT      std_logic;  -- Sortie série
10 );
11 END RegistreDecalageSerSer8bits;
12
13 ARCHITECTURE Registre of RegistreDecalageSerSer8bits is
14 signal aux : std_logic; -- Signal intermédiaire
15 begin
16 process(CLK)
17 begin
18     if (rising_edge (CLK)) then
19         if (RST = '0') then -- Mise à zéro du registre
20             aux <= '0';
21         else
22             aux <= EntreeSerie; --Entrée des données
23         end if;
24     end if;
25 end process;
26 process(aux)
27 begin
28     Q <= aux;
29 end process;
30 end Registre;
  
```

## ☞ Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default		16mA (default)
in EntreeSerie	Input	PIN_29	1	PIN_29	3.3-V LV...default		16mA (default)
out Q	Output	PIN_58	2	PIN_58	3.3-V LV...default		16mA (default)
in RST	Input	PIN_38	1	PIN_38	3.3-V LV...default		16mA (default)

## Résultat de la simulation.



## Sources :

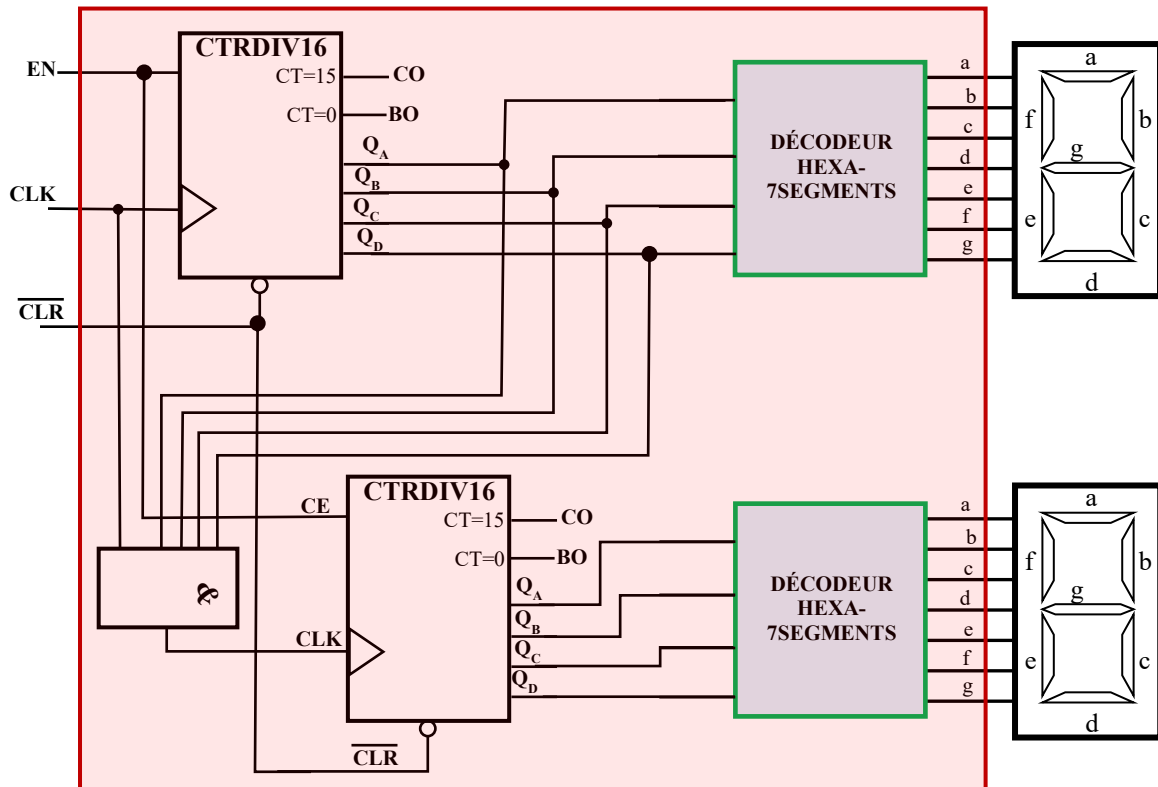
[https://fr.wikibooks.org/wiki/TD3\\_VHDL\\_Compteurs\\_et\\_registres](https://fr.wikibooks.org/wiki/TD3_VHDL_Compteurs_et_registres)

[https://hdl.telecom-paristech.fr/vhdl\\_comportemental.html](https://hdl.telecom-paristech.fr/vhdl_comportemental.html)

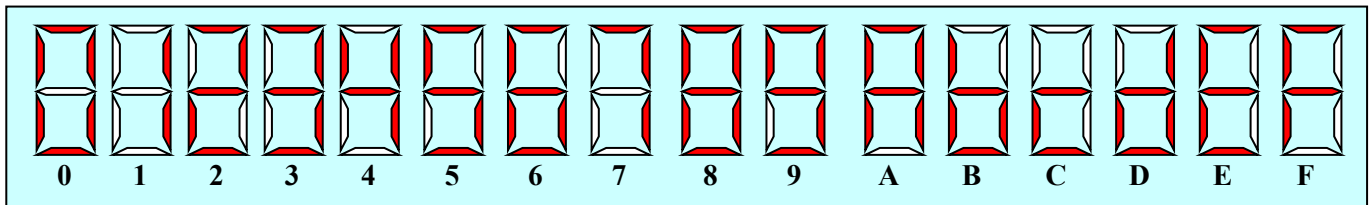
[https://www.enib.fr/~kerhoas/vhdl\\_cours\\_3.html](https://www.enib.fr/~kerhoas/vhdl_cours_3.html)

## APPLICATION N°1 : Comptage à cycle complet – Affichage 7-segments 2 digits.

- **CLK** : Signal d'horloge.
- **EN** : Entrée de validation du comptage.
- $\overline{\text{CLR}}$  : Entrée de mise à zéro du comptage.



## Affichage des chiffres décimaux et hexadécimaux.



## Programme VHDL : "CompteurAffichageHexa.vhd".

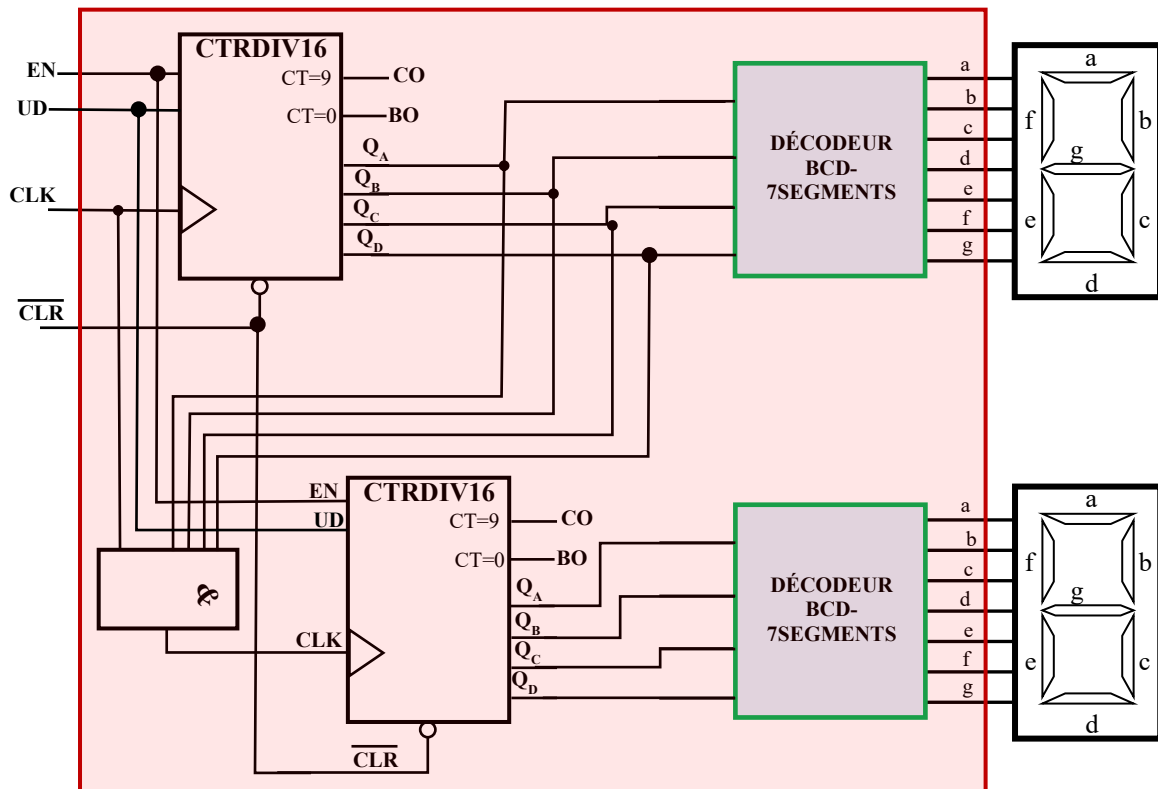
```

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 CompteurAffichageHexa IS
7  PORT (
8      CLK : IN    std_logic; -- Signal d'horloge du compteur
9      EN  : IN    std_logic; -- Entrée de validation du compteur
10     CLR : IN    std_logic; -- Mise à zéro du compteur
11     CO  : OUT   std_logic; -- Carry Out. Sortie fin de cycle du compteur 1
12     BO  : OUT   std_logic; -- Borrow Out. Sortie début de cycle du compteur 1
13     CO2 : OUT   std_logic; -- Carry Out. Sortie fin de cycle du compteur 2
14     BO2 : OUT   std_logic; -- Borrow Out. Sortie début de cycle du compteur 2
15     QN  : OUT   std_logic_vector(3 downto 0);
16     AF1 : OUT   std_logic_vector(6 downto 0); -- Les segments de l'afficheur 1
17     AF2 : OUT   std_logic_vector(6 downto 0); -- Les segments de l'afficheur 2
18 );
19 END CompteurAffichageHexa;
20
21 ARCHITECTURE LaStructure of CompteurAffichageHexa IS
22     signal aux : std_logic_vector(3 downto 0); -- Signaux intermédiaires pour le 1er digit de l'afficheur
23     signal aux2 : std_logic_vector(3 downto 0); -- Signaux intermédiaires pour le 2e digit de l'afficheur
24     begin
25         Comptage : process (CLK, CLR, EN)
26         begin
27             if ((CLR = '0') and (EN = '1')) then
28                 aux <= (others => '0');
29                 aux2 <= (others => '0');
30             elsif ((CLR = '1') and (EN = '0')) then
31                 aux <= aux;
32             elsif (rising_edge (CLK)) then -- Front montant du signal d'horloge
33                 if (aux = "1111") then -- Si fin de cycle compteur 1er digit
34                     aux <= "0000";
35                 else
36                     aux2 <= "1111";
37                     aux2 <= aux2 + 1; -- Incréméntation du compteur 2e digit
38                 end if;
39             end if;
40             aux <= aux + 1; -- Incréméntation du compteur 1er digit
41         end if;
42     end process;
43     QN <= aux;
44
45     Retenue : process (aux, aux2) -- Affichage des deux digits de l'afficheur
46     begin
47         if (aux = "0000") then
48             CO <= '0';
49             BO <= '1';
50         elsif (aux = "1111") then
51             CO <= '1';
52             BO <= '0';
53         else
54             CO <= '0';
55             BO <= '0';
56         end if;
57         if (aux2 = "0000") then
58             CO2 <= '0';
59             BO2 <= '1';
60         elsif (aux2 = "1111") then
61             CO2 <= '1';
62             BO2 <= '0';
63         else
64             CO2 <= '0';
65             BO2 <= '0';
66         end if;
67
68         -- Equations des 7-segments de l'afficheur : Premier digit.
69         AF2(0) <= (NOT(aux(0) OR aux(2))) OR (aux(1) AND aux(2)) OR (aux(1) AND (NOT aux(3)))
70         OR (aux(0) AND aux(2) AND (NOT aux(3))) OR ((NOT aux(1)) AND (NOT aux(2)) AND aux(3));
71         AF2(1) <= (NOT(aux(2) OR aux(3))) OR (NOT(aux(0) OR aux(2))) OR (aux(0) AND (NOT aux(1)) AND aux(3))
72         OR (aux(0) AND aux(1) AND (NOT aux(3))) OR (NOT(aux(0) OR aux(1) OR aux(3)));
73         AF2(2) <= ((NOT aux(2) AND aux(3)) OR (aux(2) AND (NOT aux(3))) OR (aux(0) AND (NOT aux(1)))
74         OR (aux(0) AND (NOT aux(2))) OR (NOT (aux(1) OR aux(2))));
75         AF2(3) <= (NOT(aux(0) OR aux(1) OR aux(2))) OR (NOT(aux(0)) AND NOT(aux(1)) AND aux(3))
76         OR (aux(1) AND NOT(aux(2) AND NOT(aux(3))) OR (aux(0) AND NOT(aux(2)) AND aux(3))
77         OR (NOT(aux(0)) AND aux(1) AND aux(2)) OR (aux(0) AND NOT(aux(1)) AND aux(2));
78         AF2(4) <= (NOT(aux(0)) AND aux(1)) OR (aux(2) AND aux(3)) OR (NOT(aux(0) OR aux(2)))
79         OR (aux(1) AND aux(3));
80         AF2(5) <= (NOT(aux(0) OR aux(1) OR aux(2))) OR (NOT(aux(2)) AND aux(3)) OR (aux(1) AND aux(3))
81         OR (NOT(aux(0)) AND aux(2) AND NOT(aux(3))) OR (NOT(aux(1)) AND aux(2) AND NOT(aux(3)));
82         AF2(6) <= aux(3) OR (aux(1) AND NOT(aux(2))) OR (NOT(aux(0)) AND aux(2)) OR (NOT(aux(1)) AND aux(2));
83
84         -- Equations des 7-segments de l'afficheur : Deuxième digit.
85         AF1(0) <= (NOT(aux2(0) OR aux2(2))) OR (aux2(1) AND aux2(2)) OR (aux2(1) AND (NOT aux2(3)))
86         OR (aux2(0) AND aux2(2) AND (NOT aux2(3))) OR ((NOT aux2(1)) AND (NOT aux2(2)) AND aux2(3));
87         AF1(1) <= (NOT(aux2(2) OR aux2(3))) OR (NOT(aux2(0) OR aux2(2))) OR (aux2(0) AND (NOT aux2(1)) AND aux2(3))
88         OR (aux2(0) AND aux2(1) AND (NOT aux2(3))) OR (NOT(aux2(0) OR aux2(1) OR aux2(3)));
89         AF1(2) <= ((NOT aux2(2) AND aux2(3)) OR (aux2(2) AND (NOT aux2(3))) OR (aux2(0) AND (NOT aux2(1)))
90         OR (aux2(0) AND (NOT aux2(2))) OR (NOT (aux2(1) OR aux2(2))));
91         AF1(3) <= (NOT(aux2(0) OR aux2(1) OR aux2(2))) OR (NOT(aux2(0)) AND NOT(aux2(1)) AND aux2(3))
92         OR (aux2(1) AND NOT(aux2(2) AND NOT(aux2(3))) OR (aux2(0) AND NOT(aux2(2)) AND aux2(3))
93         OR (NOT(aux2(0)) AND aux2(1) AND aux2(2)) OR (aux2(0) AND NOT(aux2(1)) AND aux2(2));
94         AF1(4) <= (NOT(aux2(0)) AND aux2(1)) OR (aux2(2) AND aux2(3)) OR (NOT(aux2(0) OR aux2(2)))
95         OR (aux2(1) AND aux2(3));
96         AF1(5) <= (NOT(aux2(0) OR aux2(1) OR aux2(2))) OR (NOT(aux2(2)) AND aux2(3)) OR (aux2(1) AND aux2(3))
97         OR (NOT(aux2(0)) AND aux2(2) AND NOT(aux2(3))) OR (NOT(aux2(1)) AND aux2(2) AND NOT(aux2(3)));
98         AF1(6) <= aux2(3) OR (aux2(1) AND NOT(aux2(2))) OR (NOT(aux2(0)) AND aux2(2)) OR (NOT(aux2(1)) AND aux2(2));
99     end process;
100 end LaStructure;

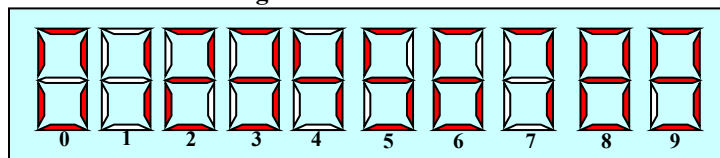
```

## APPLICATION N°2 : Comptage-Décomptage décimal – Affichage 7-segments 2 digits.

- **CLK** : Signal d'horloge.
- **EN** : Entrée de validation du comptage.
- $\overline{\text{CLR}}$  : Entrée de mise à zéro du comptage.
- **UD** : Comptage-Décomptage. **UD = 1** → Comptage ; **UD = 0** → Décomptage.



Affichage des chiffres décimaux.



## Programme VHDL : "ComptageAffichageDec2.vhd".

```

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 CompteurAffichageDec2 IS
7  PORT (
8      CLK : IN std_logic; --Signal d'horloge
9      CE : IN std_logic; --Entrée de validation du compteur
10     UD : IN std_logic; --UD=1 Comptage ; UD=0 Décomptage
11     RST : IN std_logic; --Mise à zéro du compteur
12     CO, CO2 : OUT std_logic; --Carry Out. Sortie fin de cycle
13     BO, BO2 : OUT std_logic; --Borrow Out. Sortie début de cycle
14     Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC; --Segments de l'afficheur 1
15     Ss1, Sb1, Sc1, Sd1, Se1, Sf1, Sg1 : OUT STD_LOGIC; --Segments de l'afficheur 2
16     QM : OUT std_logic_vector (3 downto 0); --Les sorties du compteur 2
17     QN : OUT std_logic_vector (3 downto 0); --Les sorties du compteur 1
18 );
19 END CompteurAffichageDec2;
20
21 ARCHITECTURE Comptage OF CompteurAffichageDec2 IS
22     signal aux : std_logic_vector(3 downto 0); --Signaux intermédiaires
23     signal aux2 : std_logic_vector(3 downto 0); --Signaux intermédiaires
24 BEGIN
25     process (CLK, CE, RST) --Processus de comptage-décomptage
26     BEGIN
27         IF (RST = '0') THEN
28             aux <= "0000";
29             aux2 <= "0000";
30         ELSEIF (CE = '1') THEN
31             IF (rising_edge(CLK)) THEN
32                 IF (UD = '1') THEN --Cycle de comptage
33                     IF (aux = "1001") THEN
34                         aux <= "0000";
35                     ELSE
36                         aux2 <= aux2 + 1;
37                     END IF;
38                 ELSE -- Cycle de décomptage
39                     IF (aux = "0000") THEN
40                         aux <= "1001";
41                     ELSEIF (aux2 = "0000") THEN
42                         aux2 <= "1001";
43                     ELSE
44                         aux2 <= aux2 - 1;
45                     END IF;
46                 END IF;
47             END IF;
48         END IF;
49         QM <= aux;
50         QN <= aux2;
51     END PROCESS; -- Fin du processus de comptage-décomptage
52
53     process (aux, aux2)
54     -- Processus qui gère l'affichage et qui génère les signaux de sortie CO et BO
55     BEGIN
56         IF (aux = "0000") THEN
57             BO <= '1';
58             CO <= '0';
59         ELSEIF (aux = "1001") THEN
60             CO <= '1';
61             BO <= '0';
62         ELSE
63             BO <= '0';
64             CO <= '0';
65         END IF;
66
67         IF (aux2 = "0000") THEN
68             BO2 <= '1';
69             CO2 <= '0';
70         ELSEIF (aux2 = "1001") THEN
71             CO2 <= '1';
72             BO2 <= '0';
73         ELSE
74             BO2 <= '0';
75             CO2 <= '0';
76         END IF;
77
78         -- Equations des 7-segments de l'afficheur : Premier digit.
79         Sa1 <= aux(1) OR aux(3) OR (aux(0) AND aux(2)) OR (NOT(aux(0) OR aux(2)));
80         Sb1 <= NOT(aux(2)) OR (aux(0) AND aux(1)) OR (NOT(aux(0) OR aux(1)));
81         Sc1 <= aux(0) OR NOT(aux(1)) OR aux(2);
82         Sd1 <= NOT(aux(0) OR aux(1)) OR (aux(1) AND NOT(aux(2))) OR (NOT(aux(0)) AND aux(1)) OR (aux(0) AND NOT(aux(1)) AND aux(2));
83         Se1 <= NOT(aux(0) OR aux(2)) OR (NOT(aux(0)) AND aux(1));
84         Sf1 <= NOT(aux(0) OR aux(1)) OR aux(3) OR (NOT(aux(0)) AND aux(2)) OR (NOT(aux(1)) AND aux(2));
85         Sg1 <= aux(3) OR (NOT(aux(0)) AND aux(2)) OR (aux(1) XOR aux(2));
86
87         -- Equations des 7-segments de l'afficheur : Deuxième digit.
88         Sa <= aux2(1) OR aux2(3) OR (aux2(0) AND aux2(2)) OR (NOT(aux2(0) OR aux2(2)));
89         Sb <= NOT(aux2(2)) OR (aux2(0) AND aux2(1)) OR (NOT(aux2(0) OR aux2(1)));
90         Sc <= aux2(0) OR NOT(aux2(1)) OR aux2(2);
91         Sd <= NOT(aux2(0) OR aux2(1)) OR (aux2(1) AND NOT(aux2(2))) OR (NOT(aux2(0)) AND aux2(1)) OR (aux2(0) AND NOT(aux2(1)) AND aux2(2));
92         Se <= NOT(aux2(0) OR aux2(2)) OR (NOT(aux2(0)) AND aux2(1));
93         Sf <= NOT(aux2(0) OR aux2(1)) OR aux2(3) OR (NOT(aux2(0)) AND aux2(2)) OR (NOT(aux2(1)) AND aux2(2));
94         Sg <= aux2(3) OR (NOT(aux2(0)) AND aux2(2)) OR (aux2(1) XOR aux2(2));
95     END PROCESS;
96 END Comptage;

```

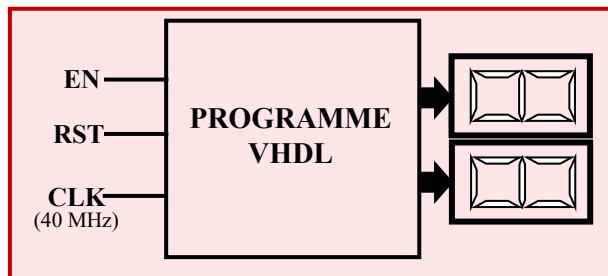


## Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
BO	Output	PIN_57	2	PIN_57	3.3-V LV..default		16mA (default)
BO2	Output	PIN_53	2	PIN_53	3.3-V LV..default		16mA (default)
CE	Input	PIN_29	1	PIN_29	3.3-V LV..default		16mA (default)
CLK	Input	PIN_12	1	PIN_12	3.3-V LV..default		16mA (default)
CO	Output	PIN_58	2	PIN_58	3.3-V LV..default		16mA (default)
CO2	Output	PIN_54	2	PIN_54	3.3-V LV..default		16mA (default)
RST	Input	PIN_38	1	PIN_38	3.3-V LV..default		16mA (default)
Sa	Output	PIN_68	2	PIN_68	3.3-V LV..default		16mA (default)
Sa1	Output	PIN_83	2	PIN_83	3.3-V LV..default		16mA (default)
Sb	Output	PIN_69	2	PIN_69	3.3-V LV..default		16mA (default)
Sb1	Output	PIN_85	2	PIN_85	3.3-V LV..default		16mA (default)
Sc	Output	PIN_70	2	PIN_70	3.3-V LV..default		16mA (default)
Sc1	Output	PIN_87	2	PIN_87	3.3-V LV..default		16mA (default)
Sd	Output	PIN_72	2	PIN_72	3.3-V LV..default		16mA (default)
Sd1	Output	PIN_91	2	PIN_91	3.3-V LV..default		16mA (default)
Se	Output	PIN_74	2	PIN_74	3.3-V LV..default		16mA (default)
Se1	Output	PIN_95	2	PIN_95	3.3-V LV..default		16mA (default)
Sf	Output	PIN_76	2	PIN_76	3.3-V LV..default		16mA (default)
Sf1	Output	PIN_97	2	PIN_97	3.3-V LV..default		16mA (default)
Sg	Output	PIN_81	2	PIN_81	3.3-V LV..default		16mA (default)
Sg1	Output	PIN_99	2	PIN_99	3.3-V LV..default		16mA (default)
UD	Input	PIN_28	1	PIN_28	3.3-V LV..default		16mA (default)

## EXERCICE N°1. Comptage décimal – Affichage 7-segments 2 digits : Chronomètre secondes.

**Objectif :** En utilisant le signal d'horloge **H = 40 MHz** de la carte CPLD, on souhaite réaliser un chronomètre qui affiche les secondes sur deux afficheurs 7-Segments.



Pour afficher les secondes, **il faut générer un signal de fréquence 1 Hz**. On divisera donc le signal d'horloge CLK par 40 000 000. D'où la réalisation d'un **compteur MODULO  $2^N = 40\,000\,000$** . N étant le nombre de bascules du compteur.

**Calcul de N.**  $N = \frac{\text{Ln}(40000000)}{\text{Ln}(2)} = 25,25$ . Nous utiliserons un compteur diviseur de fréquence à **26 bascules**.

En prenant l'exemple de l'application N°2 ci-dessus, on prendra **6 compteurs MODULO  $2^4$**  et un **compteur MODULO  $2^2$** . Ce qui nous donne un compteur MODULO  $2^{26} = 67\,108\,864$ . La remise à zéro de ce compteur se fera à la valeur 40 000 000 afin de répondre à l'objectif de réalisation d'un signal de fréquence 1Hz.

### Programmation.

- Les sorties des 6 compteurs MODULO  $2^4 = 16$  : **aux1, aux2, aux3, aux4, aux5 et aux6**.

**aux1** (aux1(3), aux1(2), aux1(1) et aux1(0)) ; **aux2** (aux2(3), aux2(2), aux2(1) et aux2(0)) :

**aux3** (aux3(3), aux3(2), aux3(1) et aux3(0)) ; **aux4** (aux4(3), aux4(2), aux4(1) et aux4(0))

**aux5** (aux5(3), aux5(2), aux5(1) et aux5(0)) ; **aux6** (aux6(3), aux6(2), aux6(1) et aux6(0))

- Les sorties du compteur MODULO  $2^2 = 4$  : **aux7** (aux7(1) et aux7(0))

➤ Arrêt du compteur à la valeur décimale **40 000 000** soit en hexadécimale **0x2625A00**. Ce qui donne pour les sorties des compteurs : **aux1(0000)** ; **aux2(0000)** ; **aux3(1010)** ; **aux4(0101)** ; **aux5(0010)** ; **aux6(0110)** et **aux7(10)**.

➤ La sortie de poids faible du compteur **aux7** (aux7(0)) donne un signal de fréquence 1Hz (ligne 58). Ce signal commande les deux décodeurs BCD 7-Segments **auxQN** et **auxQM** des afficheurs.

## Programme VHDL : "ComptageAffichageDec60s2.vhd"

```

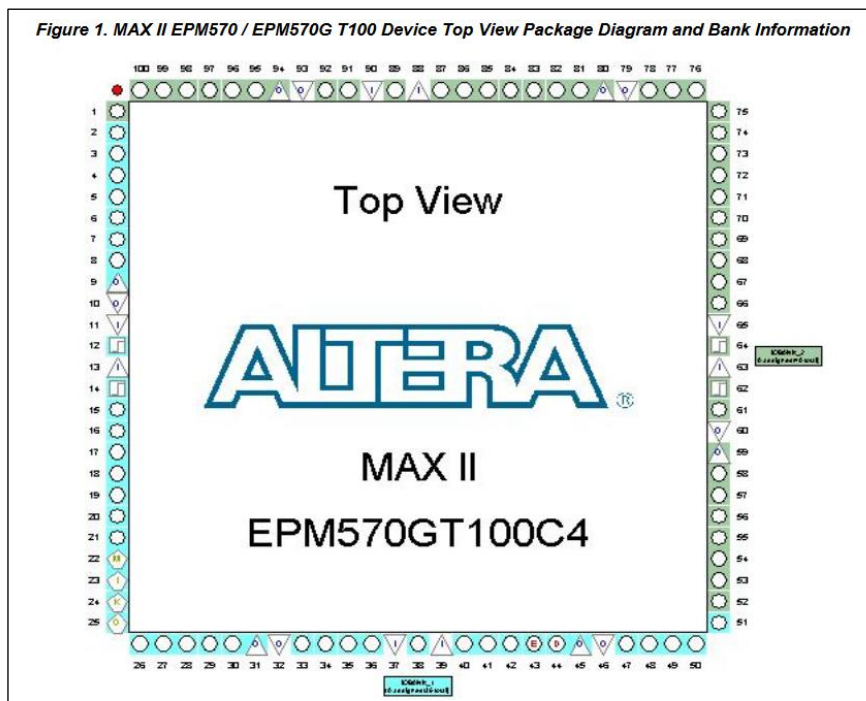
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_arith.all;
4  USE ieee.std_logic_unsigned.all;
5
6  ENTITY CompteurAffichageDec60s2 IS
7  PORT (
8      CLK : IN std_logic; --Signal d'horloge de fréquence F = 40MHz
9      CE : IN std_logic; --Entrée de validation du compteur
10     RST : IN std_logic; --Mise à zéro du compteur
11     Q : OUT std_logic; --Sortie 1Hz
12     Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC; --Segments de l'afficheur 1
13     Sal, Sbl, Scl, Sdl, Sel, Sfl, Sgl : OUT STD_LOGIC; --Segments de l'afficheur 2
14 );
15 END CompteurAffichageDec60s2;
16
17 Architecture Comptage of CompteurAffichageDec60s2 is
18     --On a besoin de 26 bascules pour un diviseur modulo 40000000
19     signal aux1 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 1
20     signal aux2 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 2
21     signal aux3 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 3
22     signal aux4 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 4
23     signal aux5 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 5
24     signal aux6 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 6
25     signal aux7 : std_logic_vector(1 downto 0); --Signaux intermédiaires pour les sorties du compteur 7
26     signal auxQN : std_logic_vector(3 downto 0); --Signaux intermédiaires pour le digit 2 de l'afficheur
27     signal auxQM : std_logic_vector(3 downto 0); --Signaux intermédiaires pour le digit 1 de l'afficheur
28
29     begin
30         process (CLK, CE, RST, aux1, aux2, aux3, aux4, aux5, aux6, aux7, auxQN, auxQM) --Processus de comptage
31         begin
32             if (RST = '0') then
33                 aux1 <= "0000";
34                 aux2 <= "0000";
35                 aux3 <= "0000";
36                 aux4 <= "0000";
37                 aux5 <= "0000";
38                 aux6 <= "0000";
39                 aux7 <= "00";
40                 auxQN <= "0000";
41                 auxQM <= "0000";
42             elsif (CE = '1') then
43                 if (rising_edge(CLK)) then --Incréméntation des 6 compteurs 4 bits et du 7e à 2 bits
44                     if (aux1 = "1111") then
45                         aux1 <= "0000";
46                     else
47                         if (aux2 = "1111") then
48                             aux2 <= "0000";
49                         else
50                             if (aux3 = "1111") then
51                                 aux3 <= "0000";
52                             else
53                                 if (aux4 = "1111") then
54                                     aux4 <= "0000";
55                                 else
56                                     if (aux5 = "1111") then
57                                         aux5 <= "0000";
58                                     else
59                                         if (aux6 = "1111") then
60                                             aux6 <= "0000";
61                                         else
62                                             if (aux7 = "11") then
63                                                 aux7 <= "00";
64                                             else
65                                                 aux7 <= aux7 + 1; --Incréméntation compteur 7
66                                                 if (auxQN = "1001") then
67                                                     auxQN <= "0000";
68                                                 else
69                                                     if (auxQM = "1001") then
70                                                         auxQM <= "0000";
71                                                     else
72                                                         auxQM <= auxQM + 1; --Incréméntation digit 1 afficheur
73                                                     end if;
74                                                 else
75                                                     auxQN <= auxQN + 1; --Incréméntation digit 2 afficheur
76                                                 end if;
77                                             end if;
78                                         end if;
79                                     else
80                                         aux6 <= aux6 + 1; --Incréméntation compteur 6
81                                     end if;
82                                 else
83                                     aux5 <= aux5 + 1; --Incréméntation compteur 5
84                                 end if;
85                             else
86                                 aux4 <= aux4 + 1; --Incréméntation compteur 4
87                             end if;
88                         else
89                             aux3 <= aux3 + 1; --Incréméntation compteur 3
90                         end if;
91                     else
92                         aux2 <= aux2 + 1; --Incréméntation compteur 2
93                     end if;
94                 else
95                     aux1 <= aux1 + 1; --Incréméntation compteur 1
96                 end if;
97             end if;
98             Q <= aux7(0);
99             -- Mise à zéro du compteur : 10 0110 0010 0101 1010 0000 0000 -> 0x2625A00 -> 40000000
100            if ((aux1="0000") and (aux2="0000") and (aux3="1010") and (aux4="0101")
101                and (aux5="0010") and (aux6="0110") and (aux7="10")) then
102                aux1 <= "0000";
103                aux2 <= "0000";
104                aux3 <= "0000";
105                aux4 <= "0000";
106                aux5 <= "0000";
107                aux6 <= "0000";
108                aux7 <= "00";
109            end if;
110        end process; -- Fin du processus de comptage
111
112        Affichage: process (aux7, auxQN, auxQM)
113        begin
114            -- Equations des 7-segments de l'afficheur : Premier digit.
115            Sa <= auxQM(1) OR auxQM(3) OR (auxQM(0) AND auxQM(2)) OR (NOT(auxQM(0) OR auxQM(2)));
116            Sb <= NOT(auxQM(2)) OR (auxQM(0) AND auxQM(1)) OR (NOT(auxQM(0) OR auxQM(1)));
117            Sc <= auxQM(0) OR NOT(auxQM(1)) OR auxQM(2);
118            Sd <= NOT(auxQM(0) OR auxQM(2)) OR (auxQM(1) AND NOT(auxQM(2))) OR (NOT(auxQM(0)) AND auxQM(1)) OR (auxQM(0) AND NOT(auxQM(1)) AND auxQM(2));
119            Se <= NOT(auxQM(0) OR auxQM(2)) OR (NOT(auxQM(0)) AND auxQM(1));
120            Sf <= NOT(auxQM(0) OR auxQM(1)) OR auxQM(3) OR (NOT(auxQM(0)) AND auxQM(2)) OR (NOT(auxQM(1)) AND auxQM(2));
121            Sg <= auxQM(3) OR (NOT(auxQM(0)) AND auxQM(2)) OR (auxQM(1) XOR auxQM(2));
122            -- Equations des 7-segments de l'afficheur : Deuxième digit.
123            Sal <= auxQN(1) OR auxQN(3) OR (auxQN(0) AND auxQN(2)) OR (NOT(auxQN(0) OR auxQN(2)));
124            Sbl <= NOT(auxQN(2)) OR (auxQN(0) AND auxQN(1)) OR (NOT(auxQN(0) OR auxQN(1)));
125            Scl <= auxQN(0) OR NOT(auxQN(1)) OR auxQN(2);
126            Sdl <= NOT(auxQN(0) OR auxQN(2)) OR (auxQN(1) AND NOT(auxQN(2))) OR (NOT(auxQN(0)) AND auxQN(1)) OR (auxQN(0) AND NOT(auxQN(1)) AND auxQN(2));
127            Sel <= NOT(auxQN(0) OR auxQN(2)) OR (NOT(auxQN(0)) AND auxQN(1));
128            Sfl <= NOT(auxQN(0) OR auxQN(1)) OR auxQN(3) OR (NOT(auxQN(0)) AND auxQN(2)) OR (NOT(auxQN(1)) AND auxQN(2));
129            Sgl <= auxQN(3) OR (NOT(auxQN(0)) AND auxQN(2)) OR (auxQN(1) XOR auxQN(2));
130        end process Affichage;
131    end Comptage;

```

## Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
CE	Input	PIN_29	1	PIN_29	3.3-V LV..default		16mA (default)
CLK	Input	PIN_62	2	PIN_62	3.3-V LV..default		16mA (default)
RST	Input	PIN_38	1	PIN_38	3.3-V LV..default		16mA (default)
Sa	Output	PIN_68	2	PIN_68	3.3-V LV..default		16mA (default)
Sa1	Output	PIN_83	2	PIN_83	3.3-V LV..default		16mA (default)
Sb	Output	PIN_69	2	PIN_69	3.3-V LV..default		16mA (default)
Sb1	Output	PIN_85	2	PIN_85	3.3-V LV..default		16mA (default)
Sc	Output	PIN_70	2	PIN_70	3.3-V LV..default		16mA (default)
Sc1	Output	PIN_87	2	PIN_87	3.3-V LV..default		16mA (default)
Sd	Output	PIN_72	2	PIN_72	3.3-V LV..default		16mA (default)
Sd1	Output	PIN_91	2	PIN_91	3.3-V LV..default		16mA (default)
Se	Output	PIN_74	2	PIN_74	3.3-V LV..default		16mA (default)
Se1	Output	PIN_95	2	PIN_95	3.3-V LV..default		16mA (default)
Sf	Output	PIN_76	2	PIN_76	3.3-V LV..default		16mA (default)
Sf1	Output	PIN_97	2	PIN_97	3.3-V LV..default		16mA (default)
Sg	Output	PIN_81	2	PIN_81	3.3-V LV..default		16mA (default)
Sg1	Output	PIN_99	2	PIN_99	3.3-V LV..default		16mA (default)
Q	Output	PIN_58	2	PIN_58	3.3-V LV..default		16mA (default)

✎ Programmer le circuit FPGA "ALTERA MAX II – EPM570GT100C4" de la carte CPLD Elektor.



Symbol	Pin Type	Symbol	Pin Type
	User I/O		CLK
	User Assigned I/O		TDI
	Filter Assigned I/O		TCK
	Unbonded Pad		TMS
	Reserved Pin		TDO
	DEV_OE		
	Other Dual Purpose		
	VCCINT		
	VCCIO		
	GNDINT		
	GNDIO		

✎ Vérifier le fonctionnement du compteur : faire valider le fonctionnement par le professeur.